
Goulib Documentation

Release 1.9.22

Ph. Guglielmetti, <https://github.com/goulu/Goulib>

Dec 08, 2017

Contents

1	Requirements	3
2	Modules	5
2.1	Goulib.colors module	6
2.2	Goulib.container module	11
2.3	Goulib.datetime2 module	14
2.4	Goulib.decorators module	24
2.5	Goulib.drawing module	24
2.6	Goulib.expr module	60
2.7	Goulib.geom module	63
2.8	Goulib.geom3d module	94
2.9	Goulib.graph module	103
2.10	Goulib.image module	145
2.11	Goulib.interval module	156
2.12	Goulib.itertools2 module	165
2.13	Goulib.markup module	174
2.14	Goulib.math2 module	190
2.15	Goulib.motion module	202
2.16	Goulib.optim module	214
2.17	Goulib.piecewise module	222
2.18	Goulib.plot module	224
2.19	Goulib.polynomial module	225
2.20	Goulib.statemachine module	228
2.21	Goulib.stats module	230
2.22	Goulib.table module	239
2.23	Goulib.tests module	246
2.24	Goulib.workdays module	253
3	Classes	257
4	Indices and tables	259
	Python Module Index	261

library of useful Python code for scientific + technical applications

see the [IPython notebook](#) for an overview of features

author Philippe Guglielmetti goulib@goulu.net

installation “pip install Goulib”

distribution <https://pypi.python.org/pypi/Goulib>

documentation <https://readthedocs.org/>

notebook <http://nbviewer.ipython.org/github/Goulu/Goulib/blob/master/notebook.ipynb>

source <https://github.com/goulu/Goulib>

CHAPTER 1

Requirements

Goulib uses “lazy” requirements. Many modules and functions do not require any other packages, packages listed in requirements.txt are needed only by some classes or functions

[Sphinx](#) is needed to generate this documentation, [Pythoscope](#) is used to generate nose unit tests

CHAPTER 2

Modules

<code>colors</code>	color conversion in various colorspace and palettes
<code>container</code>	advanced containers : Record (struct), SortedCollection, and INFINITE Sequence
<code>datetime2</code>	additions to <code>datetime</code> standard library
<code>decorators</code>	useful decorators
<code>drawing</code>	Read/Write and handle vector graphics in .dxf, .svg and .pdf formats
<code>expr</code>	simple symbolic math expressions
<code>geom</code>	2D geometry
<code>geom3d</code>	3D geometry
<code>graph</code>	efficient Euclidian Graphs for <code>networkx</code> and related algorithms
<code>image</code>	
<code>interval</code>	operations on [a..b] intervals
<code>itertools2</code>	additions to <code>itertools</code> standard library
<code>markup</code>	simple HTML/XML generation (forked from <code>markup</code>)
<code>math2</code>	more math than <code>math</code> standard library, without numpy
<code>motion</code>	motion simulation (kinematics)
<code>optim</code>	various optimization algorithms : knapsack, traveling salesman, simulated annealing, differential evolution
<code>piecewise</code>	piecewise-defined functions
<code>plot</code>	plotable rich object display on IPython/Jupyter notebooks
<code>polynomial</code>	simple manipulation of polynomials (without SimPy)
<code>statemachine</code>	state machines with graph representation
<code>stats</code>	very basic statistics functions
<code>table</code>	“mini pandas.DataFrame”
<code>tests</code>	utilities for unit tests (using nose)
<code>workdays</code>	WorkCalendar class with datetime operations on working hours, handling holidays

2.1 Goulib.colors module

color conversion in various colorspaces and palettes

`Goulib.colors.rgb2hex(c, illuminant='ignore')`

`Goulib.colors.hex2rgb(c, illuminant='ignore')`

`Goulib.colors.rgb2cmyk(rgb, **kwargs)`

Parameters `rgb` – 3-tuple of floats of red,green,blue in [0..1] range

Returns 4-tuple of floats (cyan, magenta, yellow, black) in [0..1] range

`Goulib.colors.cmyk2rgb(cmyk, **kwargs)`

Parameters `cmyk` – 4-tuple of floats (cyan, magenta, yellow, black) in [0..1] range

Result 3-tuple of floats (red,green,blue)

warning : rgb is out the [0..1] range for some cmyk

`Goulib.colors.xyz2xyy(xyz, **kwargs)`

Convert from XYZ to xyY

Based on formula from http://brucelindbloom.com/Equn_XYZ_to_xyY.html

Implementation Notes: 1. Watch out for black, where $X = Y = Z = 0$. In that case, x and y are set to the chromaticity coordinates of the reference whitepoint.

2. The output Y value is in the nominal range [0.0, Y[XYZ]].

`Goulib.colors.xyy2xyz(xyY, **kwargs)`

Convert from xyY to XYZ to

Based on formula from http://brucelindbloom.com/Equn_xyY_to_XYZ.html

Implementation Notes:

1. Watch out for the case where $y = 0$. In that case, you may want to set $X = Y = Z = 0$.
2. The output XYZ values are in the nominal range [0.0, 1.0].

`Goulib.colors.converter(c, illuminant='ignore')`

`Goulib.colors.convert(color, source, target)`

convert a color between colorspaces, eventually using intermediary steps

class `Goulib.colors.Color(value, space='RGB', name=None, illuminant='D65')`

Bases: `object`

A color with math operations and conversions Color is immutable (`._values` caches representations)

constructor :param value: string color name, hex string, or values tuple :param space: string defining the color space of value :param name: string for color name :param illuminant: string in {"A", "D50", "D55", "D65", "D75", "E"}

- D65 is used by default in skimage, see <http://scikit-image.org/docs/dev/api/skimage.color.html>
- D50 is used in Pantone and other graphic arts

— **__init__** (`value, space='RGB', name=None, illuminant='D65'`)

constructor :param value: string color name, hex string, or values tuple :param space: string defining the color space of value :param name: string for color name :param illuminant: string in {"A", "D50", "D55", "D65", "D75", "E"}

- D65 is used by default in skimage, see <http://scikit-image.org/docs/dev/api/skimimage.color.html>
- D50 is used in Pantone and other graphic arts

name

convert (*target*, ***kwargs*)

Parameters *target* – str of desired colorspace, or none for default

Returns color in target colorspace

str (*mode=None*)

native

rgb

hex

lab

luv

cmyk

hsv

xyz

xyY

__hash__ ()

__repr__ ()

compose (*other*, *f*, *mode='rgb'*)
compose colors in given mode

__add__ (*other*)

__radd__ (*other*)
only to allow sum(colors) easily

__sub__ (*other*)

__mul__ (*factor*)

__neg__ ()
complementary color

deltaE (*other*)
color difference according to CIEDE2000 https://en.wikipedia.org/wiki/Color_difference

isclose (*other*, *abs_tol=1*)
<http://zschuessler.github.io/DeltaE/learn/> <= 1.0 Not perceptible by human eyes. 1 - 2 Perceptible through close observation. 2 - 10 Perceptible at a glance. 11 - 49 Colors are more similar than opposite 100 Colors are exact opposite

__eq__ (*other*)

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class Goulib.colors.**Palette** (data=[], keys=256)

Bases: `collections.OrderedDict`

dict of Colors indexed by anything

__init__ (data=[], keys=256)

update (data, keys=256)

updates the dictionary with new colors :param data: colors to add :param keys: keys to use in dict, or int to discretize the Colormap

index (c, dE=5)

Returns key of c or nearest color, None if distance is larger than deltaE

__repr__ ()

patches (wide=64, size=(16, 16))

Image made of each palette color

pil

Returns a sequence of integers, or a string containing a binary

representation of the palette. In both cases, the palette contents should be ordered (r, g, b, r, g, b, ...). The palette can contain up to 768 entries (3*256). If a shorter palette is given, it is padded with zeros. [#http://effbot.org/zone/creating-palette-images.htm](http://effbot.org/zone/creating-palette-images.htm)

```
sorted (key=<function Palette.<lambda>>)

__class__
    alias of type

__contains__ ()
    True if D has a key k, else False.

__delattr__
    Implement delattr(self, name).

__delitem__
    Delete self[key].

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__getitem__ ()
    x.__getitem__(y) <==> x[y]

__gt__
    Return self>value.

__hash__ = None

__iter__
    Implement iter(self).

__le__
    Return self<=value.

__len__
    Return len(self).

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    Return state information for pickling

__reduce_ex__ ()
    helper for pickle
```

__reversed__ () $\leq$ $\Rightarrow$ *reversed(od)*

__setattr__

Implement setattr(self, name, value).

__setitem__

Set self[key] to value.

__sizeof__ ()

__str__

Return str(self).

clear () $\rightarrow$ None. Remove all items from od.

copy () $\rightarrow$ a shallow copy of od

fromkeys (S[, v]) $\rightarrow$ New ordered dictionary with keys from S.

If not specified, the value defaults to None.

get (k[, d]) $\rightarrow$ D[k] if k in D, else d. d defaults to None.

items ()

keys ()

move_to_end ()

Move an existing element to the end (or beginning if last==False).

Raises KeyError if the element does not exist. When last=True, acts like a fast version of self[key]=self.pop(key).

pop (k[, d]) $\rightarrow$ v, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised.

popitem () $\rightarrow$ (k, v), return and remove a (key, value) pair.

Pairs are returned in LIFO order if last is true or FIFO order if false.

setdefault (k[, d]) $\rightarrow$ od.get(k,d), also set od[k]=d if k not in od

values ()

Goulib.colors.**ColorTable** (colors, key=None, width=10)

Goulib.colors.**color_to_aci** (x, nearest=True)

Returns int Autocad Color Index of color x

Goulib.colors.**aci_to_color** (x, block_color=None, layer_color=None)

Goulib.colors.**deltaE** (c1, c2)

Goulib.colors.**nearest_color** (c, l=None, opt=<built-in function min>, comp=<function deltaE>)

Parameters

- **x** – Color
- **l** – list or dict of Color, color by default
- **opt** – with opt=max you can find the most different color ...

Returns nearest Color of x in l

Goulib.colors.**color_range** (n, start, end, space='hsv')

Parameters

- **n** – int number of colors to generate

- **start** – string hex color or color name
- **end** – string hex color or color name

Result list of n Color interpolated between start and end, included

2.2 Goulib.container module

advanced containers : Record (struct), SortedCollection, and INFINITE Sequence

class Goulib.container.**Record**(*args, **kwargs)

Bases: collections.OrderedDict

mimics a Pascal record or a C struct

__init__(*args, **kwargs)

__getattr__(name)

__setattr__(name, value)

__str__()

__class__

alias of type

__contains__()

True if D has a key k, else False.

__delattr__

Implement delattr(self, name).

__delitem__

Delete self[key].

__dir__() → list

default dir() implementation

__eq__

Return self==value.

__format__()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__getitem__()

x.__getitem__(y) <==> x[y]

__gt__

Return self>value.

__hash__ = None

__iter__

Implement iter(self).

__le__

Return self<=value.

`__len__`

Return len(self).

`__lt__`

Return self<value.

`__ne__`

Return self!=value.

`__new__` ()

Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()

Return state information for pickling

`__reduce_ex__` ()

helper for pickle

`__repr__`

Return repr(self).

`__reversed__` () <==> *reversed(od)*

`__setitem__`

Set self[key] to value.

`__sizeof__` ()

`clear` () → None. Remove all items from od.

`copy` () → a shallow copy of od

`fromkeys` (*S* [, *v*]) → New ordered dictionary with keys from *S*.

If not specified, the value defaults to None.

`get` (*k* [, *d*]) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to None.

`items` ()

`keys` ()

`move_to_end` ()

Move an existing element to the end (or beginning if last==False).

Raises `KeyError` if the element does not exist. When last=True, acts like a fast version of self[key]=self.pop(key).

`pop` (*k* [, *d*]) → *v*, remove specified key and return the corresponding value. If key is not found, *d* is returned if given, otherwise `KeyError` is raised.

`popitem` () → (*k*, *v*), return and remove a (key, value) pair.

Pairs are returned in LIFO order if last is true or FIFO order if false.

`setdefault` (*k* [, *d*]) → od.get(*k*,*d*), also set od[*k*]=*d* if *k* not in od

`update` ()

`values` ()

class Goulib.container.**Sequence** (*iterf=None*, *itemf=None*, *containf=None*, *desc=''*)

Bases: `object`

combines a generator and a read-only list used for INFINITE numeric (integer) sequences

Parameters

- **`iterf`** – optional iterator, or a function returning an iterator

- **itemf** – optional function(i) returning the i-th element
- **containf** – optional function(n) return bool True if n belongs to Sequence
- **desc** – string description

__init__ (*iterf=None, itemf=None, containf=None, desc=''*)

Parameters

- **iterf** – optional iterator, or a function returning an iterator
- **itemf** – optional function(i) returning the i-th element
- **containf** – optional function(n) return bool True if n belongs to Sequence
- **desc** – string description

__repr__ ()

save (*filename, comment=None, n=1000, maxtime=10*)

__iter__ ()

reset the generator

Returns a tee-ed copy of iterf

__getitem__ (*i*)

index (*v*)

__contains__ (*n*)

__add__ (*other*)

__sub__ (*other*)

__or__ (*other*)

Returns Sequence with items from both operands

__and__ (*other*)

Returns Sequence with items in both operands

__mod__ (*other*)

Returns Sequence with items from left operand not in right

apply (*f, containf=None, desc=''*)

filter (*f, desc=''*)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

accumulate (op=<built-in function add>, skip_first=False)

pairwise (op, skip_first=False)

sort (key=None, buffer=100)

unique (buffer=100)

Parameters **buffer** – int number of last elements found.

if two identical elements are separated by more than this number of elements in self, they might be generated twice in the resulting Sequence :return: Sequence made of unique elements of this one

2.3 Goulib.datetime2 module

additions to `datetime` standard library

```
class Goulib.datetime2.datetime2 (*args, **kwargs)
    Bases: datetime.datetime
    __init__ (*args, **kwargs)
    __sub__ (other)
```

__add__
Return self+value.

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
Formats self with strftime.

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__radd__
Return value+self.

__reduce__ () -> (cls, state)

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__rsub__
Return value-self.

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

astimezone()
tz -> convert to local time in new timezone tz

combine()
date, time -> datetime with same date and time fields

ctime()
Return ctime() style string.

date()
Return date object with same year, month and day.

day

dst()
Return self.tzinfo.dst(self).

fromordinal()
int -> date corresponding to a proleptic Gregorian ordinal.

fromtimestamp()
timestamp[, tz] -> tz's local time from POSIX timestamp.

hour

isocalendar()
Return a 3-tuple containing ISO year, week number, and weekday.

isoformat()
[sep] -> string in ISO 8601 format, YYYY-MM-DDTHH:MM:SS[.mmmmmm][+HH:MM].
sep is used to separate the year from the time, and defaults to 'T'.

isoweekday()
Return the day of the week represented by the date. Monday == 1 ... Sunday == 7

max = datetime.datetime(9999, 12, 31, 23, 59, 59, 999999)

microsecond

min = datetime.datetime(1, 1, 1, 0, 0)

minute

month

now()
Returns new datetime object representing current time local to tz.
tz Timezone object.
If no tz is specified, uses local timezone.

replace()
Return datetime with new specified fields.

resolution = datetime.timedelta(0, 0, 1)

second

strftime()
format -> strftime() style string.

strptime()
string, format -> new datetime parsed from a string (like time.strptime()).

time()
Return time object with same time but with tzinfo=None.

timestamp()
Return POSIX timestamp as float.

timetuple()
Return time tuple, compatible with time.localtime().

timetz()
Return time object with same time and tzinfo.

today()
Current date or datetime: same as self.__class__.fromtimestamp(time.time()).

toordinal()
Return proleptic Gregorian ordinal. January 1 of year 1 is day 1.

tzinfo

tzname()
Return self.tzinfo.tzname(self).

utcfromtimestamp()
Construct a naive UTC datetime from a POSIX timestamp.

utcnow()
Return a new datetime representing UTC day and time.

utcoffset()
Return self.tzinfo.utcoffset(self).

utctimetuple()
Return UTC time tuple, compatible with time.localtime().

weekday()
Return the day of the week represented by the date. Monday == 0 ... Sunday == 6

year

```
class Goulib.datetime2.date2
    Bases: datetime.date
    __init__ (*args, **kwargs)
    __add__
        Return self+value.
    __class__
        alias of type
    __delattr__
        Implement delattr(self, name).
    __dir__ () → list
        default dir() implementation
    __eq__
        Return self==value.
    __format__ ()
        Formats self with strftime.
    __ge__
        Return self>=value.
```

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__radd__
Return value+self.

__reduce__ () -> (cls, state)

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__rsub__
Return value-self.

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

__sub__
Return self-value.

ctime ()
Return ctime() style string.

day

fromordinal ()
int -> date corresponding to a proleptic Gregorian ordinal.

fromtimestamp ()
timestamp -> local date from a POSIX timestamp (like time.time()).

isocalendar ()
Return a 3-tuple containing ISO year, week number, and weekday.

isoformat()
Return string in ISO 8601 format, YYYY-MM-DD.

isoweekday()
Return the day of the week represented by the date. Monday == 1 ... Sunday == 7

max = datetime.date(9999, 12, 31)

min = datetime.date(1, 1, 1)

month

replace()
Return date with new specified fields.

resolution = datetime.timedelta(1)

strftime()
format -> strftime() style string.

timetuple()
Return time tuple, compatible with time.localtime().

today()
Current date or datetime: same as self.__class__.fromtimestamp(time.time()).

toordinal()
Return proleptic Gregorian ordinal. January 1 of year 1 is day 1.

weekday()
Return the day of the week represented by the date. Monday == 0 ... Sunday == 6

year

class Goulib.datetime2.time2(*args, **kwargs)

Bases: `datetime.time`

__init__(*args, **kwargs)

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

__dir__() → list
default dir() implementation

__eq__
Return self==value.

__format__()
Formats self with strftime.

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ () -> (cls, state)

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

dst ()
Return self.tzinfo.dst(self).

hour

isoformat ()
Return string in ISO 8601 format, HH:MM:SS[.mmmmmm][+HH:MM].

max = datetime.time(23, 59, 59, 999999)

microsecond

min = datetime.time(0, 0)

minute

replace ()
Return time with new specified fields.

resolution = datetime.timedelta(0, 0, 1)

second

strftime ()
format -> strftime() style string.

tzinfo

tzname ()
Return self.tzinfo.tzname(self).

utcoffset ()
Return self.tzinfo.utcoffset(self).

class Goulib.datetime2.**timedelta2** (*args, **kwargs)
Bases: datetime.timedelta

`__init__` (**args, **kwargs*)
`isoformat` ()
`__abs__`
abs(self)
`__add__`
Return self+value.
`__bool__`
self != 0
`__class__`
alias of `type`
`__delattr__`
Implement delattr(self, name).
`__dir__` () → list
default dir() implementation
`__divmod__`
Return divmod(self, value).
`__eq__`
Return self==value.
`__floordiv__`
Return self//value.
`__format__` ()
default object formatter
`__ge__`
Return self>=value.
`__getattr__`
Return getattr(self, name).
`__gt__`
Return self>value.
`__hash__`
Return hash(self).
`__le__`
Return self<=value.
`__lt__`
Return self<value.
`__mod__`
Return self%value.
`__mul__`
Return self*value.
`__ne__`
Return self!=value.
`__neg__`
-self

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__pos__
+self

__radd__
Return value+self.

__rdivmod__
Return divmod(value, self).

__reduce__ () -> (cls, state)

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__rfloordiv__
Return value//self.

__rmod__
Return value%self.

__rmul__
Return value*self.

__rsub__
Return value-self.

__rtruediv__
Return value/self.

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

__sub__
Return self-value.

__truediv__
Return self/value.

days
Number of days.

max = datetime.timedelta(999999999, 86399, 999999)

microseconds
Number of microseconds (>= 0 and less than 1 second).

min = datetime.timedelta(-999999999)

resolution = datetime.timedelta(0, 0, 1)

seconds
Number of seconds (>= 0 and less than 1 day).

total_seconds()

Total seconds in the duration.

`Goulib.datetime2.datetimeef(d, t=None, fmt='%Y-%m-%d')`

“converts something to a datetime :param d: can be:

- **datetime** : result is a copy of d with time optionally replaced
- **date** : result is date at time t, (00:00AM by default)
- **int or float** : if fmt is None, d is considered as Excel date numeric format (see <http://answers.oreilly.com/topic/1694-how-excel-stores-date-and-time-values/>)
- **string or specified format**: result is datetime parsed using specified format string

Parameters

- **fmt** – format string. See <http://docs.python.org/2/library/datetime.html#strftime-strptime-behavior>
- **t** – optional time. replaces the time of the datetime obtained from d. Allows `datetimeef(date,time)`

Returns datetime

`Goulib.datetime2.datef(d, fmt='%Y-%m-%d')`

converts something to a date. See `datetimeef`

`Goulib.datetime2.timef(t, fmt='%H:%M:%S')`

converts something to a time. See `datetimeef`

`Goulib.datetime2.fmt2regex(fmt)`

converts a date/time format string to the regex that comiles it

Parameters **fmt** – string

Returns compiled regex

`Goulib.datetime2.timedeltaf(t, fmt=None)`

converts something to a timedelta.

Parameters **t** – can be:

- already a timedelta, or a time, or a int/float giving a number of days (Excel)
- or a string in HH:MM:SS format by default
- or a string in timedelta `str()` output format

`Goulib.datetime2.strftimedelta(t, fmt='%H:%M:%S')`

Parameters **t** – float seconds or timedelta

`Goulib.datetime2.tdround(td, s=1)`

return timedelta rounded to s seconds

`Goulib.datetime2.minutes(td)`

Parameters **td** – timedelta

Returns float timedelta in minutes

`Goulib.datetime2.hours(td)`

Parameters **td** – timedelta

Returns float timedelta in hours

`Goulib.datetime2.daysgen (start, length, step=datetime.timedelta(1))`
returns a range of dates or datetimes

`Goulib.datetime2.days (start, length, step=datetime.timedelta(1))`

`Goulib.datetime2.timedelta_sum (timedeltas)`

`Goulib.datetime2.timedelta_div (t1, t2)`
divides a timedelta by a timedelta or a number. should be a method of timedelta...

`Goulib.datetime2.timedelta_mul (t1, t2)`
multiplies a timedelta. should be a method of timedelta...

`Goulib.datetime2.time_sub (t1, t2)`
subtracts 2 time. should be a method of time...

`Goulib.datetime2.time_add (t, d)`
adds delta to time. should be a method of time...

`Goulib.datetime2.add_months (date, months)`

`Goulib.datetime2.date_add (date, years=0, months=0, weeks=0, days=0)`

`Goulib.datetime2.equal (a, b, epsilon=datetime.timedelta(0, 0, 500000))`
approximately equal. Use this instead of `a==b` :return: True if a and b are less than seconds apart

`Goulib.datetime2.datetime_intersect (t1, t2)`
returns timedelta overlap between 2 intervals (tuples) of datetime

`Goulib.datetime2.time_intersect (t1, t2)`
returns timedelta overlap between 2 intervals (tuples) of time

2.4 Goulib.decorators module

useful decorators

`Goulib.decorators.memoize (obj)`

`Goulib.decorators.debug (func)`

`Goulib.decorators.noddebug (func)`

`Goulib.decorators.get_thread_pool ()`

`Goulib.decorators.timeout (timeout)`

`Goulib.decorators.itimeout (iterable, timeout)`
timeout for loops :param iterable: any iterable :param timeout: float max running time in seconds :yield: items in iterator until timeout occurs :raise: multiprocessing.TimeoutError if timeout occurred

2.5 Goulib.drawing module

Read/Write and handle vector graphics in .dxf, .svg and .pdf formats

requires

- `svg.path` for svg input
- `matplotlib` for bitmap + svg and pdf output

- [dxfwrite](#) for dxf output
- **optional**
- [dxfgrabber](#) for dxf input
- [pdfminer.six](#) for pdf input

Goulib.drawing.**Trans** (*scale=1, offset=None, rotation=None*)

Parameters

- **scale** – float or (scalex,scaley) tuple of scale factor
- **offset** – Vector3
- **rotation** – float angle in degrees

Returns Matrix3 of generalized scale+offset+rotation

class Goulib.drawing.**BBox** (*p1=None, p2=None*)

Bases: [Goulib.interval.Box](#)

bounding box

Parameters

- **pt1** – Point2 first corner (any)
- **pt2** – Point2 opposite corner (any)

__init__ (*p1=None, p2=None*)

Parameters

- **pt1** – Point2 first corner (any)
- **pt2** – Point2 opposite corner (any)

xmin

ymin

xmax

ymax

xmed

ymed

width

height

area

__contains__ (*other*)

Returns True if other lies in bounding box.

__iadd__ (*pt*)

enlarge box if required to contain specified point :param pt1: geom.Point2 point to add

__call__ ()

Returns list of flatten corners

size ()

Returns geom.Vector2 with xy sizes

center ()

Returns Pt center

trans (*trans*)

Parameters **trans** – Xform

Returns *BBox* = self transformed by trans

class Goulib.drawing.**Entity**

Bases: *Goulib.plot.Plot*

Base class for all drawing entities

color = 'black'

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

end

__repr__ ()

center

bbox ()

Returns *BBox* bounding box of Entity

isclosed ()

isline ()

isvertical (*tol=0.01*)

ishorizontal (*tol=0.01*)

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

static from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

static from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

patches (***kwargs*)

Returns list of (a single) *Patch* corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

static figure (*box*, ***kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

draw (*fig=None*, ***kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

render (*fmt*, ***kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

class `Goulib.drawing.Spline` (*points*)

Bases: `Goulib.drawing.Entity`, `Goulib.geom.Geometry`

cubic spline segment

Parameters **points** – list of (x,y) tuples

__init__ (*points*)

Parameters **points** – list of (x,y) tuples

start

end

xy

length

Returns float (very) approximate length

bbox ()

swap ()

swap start and end

__abstractmethods__ = frozenset()

class `Goulib.drawing.Group`

Bases: `list`, `Goulib.drawing._Group`

group of Entities but it is a Geometry since we can intersect, connect and compute distances between Groups

color

layer

append (*entity*, ***kwargs*)

append entity to group :param entity: Entity :param kwargs: dict of attributes copied to entity :return: Group (or Chain) to which the entity was added, or None if entity was None

extend (*entities*, ***kwargs*)

__copy__ ()

swap ()

swap start and end

chainify (*mergeable*)

merge all possible entities into chains

from_dxf (*dxf*, *layers=None*, *only=[]*, *ignore=['POINT']*, *trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0)*, *flatten=False*)

Parameters

- **dxf** – dxf.entity
- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read
- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

__abstractmethods__ = frozenset()

class Goulib.drawing.**Instance** (*group*, *trans*)

Bases: Goulib.drawing._Group

Parameters

- **group** – Group
- **trans** – optional mat3 of transformation

__init__ (*group*, *trans*)

Parameters

- **group** – Group
- **trans** – optional mat3 of transformation

static from_dxf (*e*, *blocks*, *mat3*)

Parameters

- **e** – dxf.entity
- **blocks** – dict of Groups indexed by name
- **mat3** – Matrix3 transform

__repr__ ()

__iter__ ()

__abstractmethods__ = frozenset()

class Goulib.drawing.**Chain** (*data=[]*)

Bases: *Goulib.drawing.Group*

group of contiguous Entities (Polyline or similar)

__init__ (*data=[]*)

start

end

`__repr__()`

contiguous (*edge, tol=1e-06, allow_swap=True*)

check if edge can be appended to the chain :param edge: *Entity* to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return: int,bool index where to append in chain, swap of edge required

append (*entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs*)

append entity to chain, ensuring contiguity :param entity: *Entity* to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param mergeable: function of the form f(e1,e2) returning True if entities e1,e2 can be merged :param attrs: attributes passed to Group.append :return: self, or None if edge is not contiguous

static from_pdf (*path, trans, color*)

Parameters *path* – pdf path

Returns Entity of correct subtype

See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf p. 132

static from_svg (*path, color*)

Parameters *path* – svg path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- *e* – dxf.entity
- *mat3* – Matrix3 transform

Returns Entity of correct subtype

to_dxf (*split=False, **attr*)

Parameters

- *split* – bool if True, each segment in Chain is saved separately
- *attr* – dict of graphic attributes

Returns polyline or list of entities along the chain

`__abstractmethods__ = frozenset()`

`Goulib.drawing.chains` (*group, tol=1e-06, mergeable=None*)

build chains from all possible segments in group :param mergeable: function(e1,e2) returning True if entities e1,e2 can be merged

class `Goulib.drawing.Rect` (**args*)

Bases: *Goulib.drawing.Chain*

a rectangle starting at low/left and going trigowise through top/right

`__init__` (**args*)

p1

p2

`__repr__` ()

`__abstractmethods__ = frozenset()`

class Goulib.drawing.**Text** (*text, point, size=12, rotation=0*)

Bases: *Goulib.drawing.Entity*

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

__init__ (*text, point, size=12, rotation=0*)

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

bbox ()

length

Returns float length of the text contour in mm

intersect (*other*)

to_dxf (***attr*)

patches (***kwargs*)

Returns list of (a single) *Patch* corresponding to entity

class Goulib.drawing.**Drawing** (*data=[], **kwargs*)

Bases: *Goulib.drawing.Group*

list of Entities representing a vector graphics drawing

__abstractmethods__ = frozenset()

__init__ (*data=[], **kwargs*)

load (*filename, **kwargs*)

read_pdf (*filename, **kwargs*)

reads a vector graphics on a .pdf file only the first page is parsed

read_svg (*content, **kwargs*)

appends svg content to drawing :param content: string, either filename or svg content

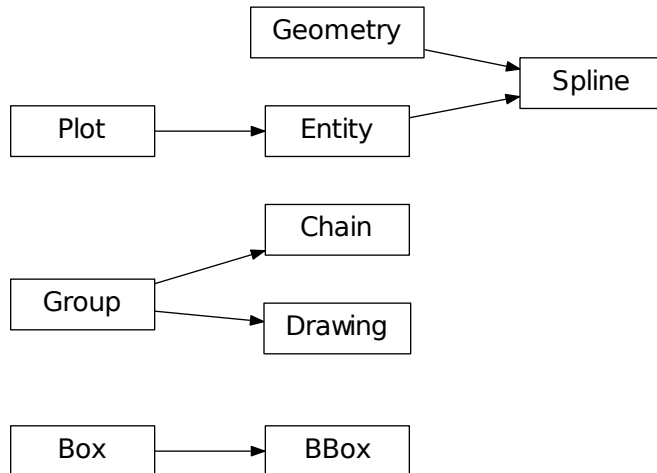
read_dxf (*filename, options=None, **kwargs*)

reads a .dxf file :param filename: string path to .dxf file to read :param options: passed to *from_dxf*

save (*filename, **kwargs*)

save graph in various formats

2.5.1 Classes



Read/Write and handle vector

graphics in .dxf, .svg and .pdf formats

requires

- [svg.path](#) for svg input
- [matplotlib](#) for bitmap + svg and pdf output
- [dxfwrite](#) for dxf output

optional

- [dxfgrabber](#) for dxf input
- [pdfminer.six](#) for pdf input

Goulib.drawing.**Trans** (*scale=1, offset=None, rotation=None*)

Parameters

- **scale** – float or (scalex,scaley) tuple of scale factor
- **offset** – `Vector3`
- **rotation** – float angle in degrees

Returns `Matrix3` of generalized scale+offset+rotation

class Goulib.drawing.**BBox** (*p1=None, p2=None*)

Bases: [Goulib.interval.Box](#)

bounding box

Parameters

- **pt1** – `Point2` first corner (any)
- **pt2** – `Point2` opposite corner (any)

`__init__` (*p1=None, p2=None*)

Parameters

- **pt1** – `Point2` first corner (any)
- **pt2** – `Point2` opposite corner (any)

xmin

ymin

xmax

ymax

xmed

ymed

width

height

area

`__contains__` (*other*)

Returns True if other lies in bounding box.

`__iadd__` (*pt*)

enlarge box if required to contain specified point :param pt1: `geom.Point2` point to add

`__call__` ()

Returns list of flatten corners

size ()

Returns `geom.Vector2` with xy sizes

center ()

Returns Pt center

trans (*trans*)

Parameters **trans** – Xform

Returns `BBox` = self transformed by trans

`__add__` (*other*)

enlarge box if required to contain specified point :param other: `Box` or (list of) N-tuple point(s) :return: new `Box` containing both

`__class__`

alias of `type`

`__delattr__`

Implement `delattr`(self, name).

`__delitem__`

Delete `self[key]`.

`__dir__` () → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__()`
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__getitem__()`
x.__getitem__(y) <==> x[y]

`__gt__`
Return self>value.

`__hash__ = None`

`__imul__`
Implement self*=value.

`__iter__`
Implement iter(self).

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__lt__`
Return self<value.

`__mul__`
Return self*value.n

`__ne__`
Return self!=value.

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__nonzero__()`

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__reversed__()`
L.__reversed__() – return a reverse iterator over the list

`__rmul__`
Return self*value.

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

__sizeof__ ()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

append (object) → None – append object to end

clear () → None – remove all items from L

copy () → list – a shallow copy of L

corner (n)
return n-th corner of box 0-th corner is “start” made of all minimal values of intervals -1.th corner is “end”, made of all maximal values of intervals

count (value) → integer – return number of occurrences of value

empty ()
Returns True iff Box is empty.

end

extend (iterable) → None – extend list by appending elements from the iterable

index (value[, start[, stop]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

max

min

pop ([index]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (value) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

sort (key=None, reverse=False) → None – stable sort **IN PLACE**

start

class Goulib.drawing.**Entity**
Bases: *Goulib.plot.Plot*
Base class for all drawing entities

color = ‘black’

setattr (**kwargs)
set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

end

__repr__ ()

center

bbox ()

Returns *BBox* bounding box of Entity

isclosed()

isline()

isvertical (*tol=0.01*)

ishorizontal (*tol=0.01*)

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

static from_svg (*path, color*)

Parameters *path* – svg path

Returns Entity of correct subtype

static from_pdf (*path, trans, color*)

Parameters *path* – pdf path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- *e* – dxf.entity
- *mat3* – Matrix3 transform

Returns Entity of correct subtype

patches (***kwargs*)

Returns list of (a single) *Patch* corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

static figure (*box, **kwargs*)

Parameters

- *box* – `drawing.BBox` bounds and clipping box
- *kwargs* – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

draw (*fig=None, **kwargs*)

draw entities :param *fig*: matplotlib figure where to draw. `figure(g)` is called if missing :return: *fig,patch*

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

__class__

alias of *type*

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

html (**kwargs)

plot (**kwargs)
renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

save (filename, **kwargs)

svg (**kwargs)

class Goulib.drawing.**Spline** (points)
Bases: *Goulib.drawing.Entity, Goulib.geom.Geometry*
cubic spline segment

Parameters **points** – list of (x,y) tuples

```

__init__(points)
    Parameters points – list of (x,y) tuples

start
end
xy
length
    Returns float (very) approximate length

bbox()
swap()
    swap start and end

__abstractmethods__ = frozenset()

__class__
    alias of ABCMeta

__contains__(pt)

__delattr__
    Implement delattr(self, name).

__dir__() → list
    default dir() implementation

__eq__
    Return self==value.

__format__()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__()
    Create and return a new object. See help(type) for accurate signature.

__reduce__()
    helper for pickle

```

__reduce_ex__ ()
helper for pickle

__repr__ ()

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

center

color = 'black'

connect (*other*)

Returns Geometry shortest (Segment2 or Segment3) that connects self to other

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

figure (*box, **kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to *~matplotlib.pyplot.figure*

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class `Goulib.drawing.Group`

Bases: `list`, `Goulib.drawing._Group`

group of Entities but it is a Geometry since we can intersect, connect and compute distances between Groups

color

layer

append (*entity, **kwargs*)

append entity to group :param *entity*: Entity :param *kwargs*: dict of attributes copied to entity :return: Group (or Chain) to which the entity was added, or None if entity was None

extend (*entities, **kwargs*)

__copy__ ()

swap ()

swap start and end

chainify (*mergeable*)

merge all possible entities into chains

from_dxf (*dxf, layers=None, only=[], ignore=['POINT'], trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0), flatten=False*)

Parameters

- **dxf** – dxf.entity
- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read

- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

```
__abstractmethods__ = frozenset()

__add__
    Return self+value.

__class__
    alias of ABCMeta

__contains__
    Return key in self.

__delattr__
    Implement delattr(self, name).

__delitem__
    Delete self[key].

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__getitem__ ()
    x.__getitem__(y) <==> x[y]

__gt__
    Return self>value.

__hash__ = None

__iadd__
    Implement self+=value.

__imul__
    Implement self*=value.

__init__
    Initialize self. See help(type(self)) for accurate signature.

__iter__
    Implement iter(self).

__le__
    Return self<=value.
```

__len__
Return len(self).

__lt__
Return self<value.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__reversed__ ()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__ ()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBBox* bounding box of Entity

center

clear () → None – remove all items from L

connect (*other*)

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box*, ***kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_pdf (*path*, *trans*, *color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path*, *color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises `ValueError` if the value is not present.

insert ()

`L.insert(index, object)` – insert object before index

intersect (*other*)

Parameters **other** – *geom.Entity*

Result generate tuples (`Point2`, `Entity_self`) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of `Patch` corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns `Point2` or `Point3` at parameter *u*

pop ([*index*]) → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises `ValueError` if the value is not present.

render (*fmt*, ***kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

```

reverse ()
    L.reverse() – reverse IN PLACE

save (filename, **kwargs)

setattr (**kwargs)
    set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

sort (key=None, reverse=False) → None – stable sort *IN PLACE*

start

svg (**kwargs)

tangent (u)
    Returns Vector2 or Vector3 tangent at parameter u

to_dxf (**kwargs)
    Returns flatten list of dxf entities

class Goulib.drawing.Instance (group, trans)
    Bases: Goulib.drawing._Group

    Parameters
        • group – Group
        • trans – optional mat3 of transformation

    __init__ (group, trans)

    Parameters
        • group – Group
        • trans – optional mat3 of transformation

static from_dxf (e, blocks, mat3)

    Parameters
        • e – dxf.entity
        • blocks – dict of Groups indexed by name
        • mat3 – Matrix3 transform

__repr__ ()

__iter__ ()

__abstractmethods__ = frozenset()

__class__
    alias of ABCMeta

__contains__ (pt)

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

```

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBBox* bounding box of Entity

center

color = 'black'

connect (*other*)

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box, **kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

Parameters **other** – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of *Patch* corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***kwargs*)

Returns flatten list of dxf entities

```
class Goulib.drawing.Chain(data=[])
    Bases: Goulib.drawing.Group

    group of contiguous Entities (Polyline or similar)

    __init__(data=[])

    start
    end

    __repr__()

    contiguous(edge, tol=1e-06, allow_swap=True)
        check if edge can be appended to the chain :param edge: Entity to append :param tol: float tolerance
        on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return:
        int,bool index where to append in chain, swap of edge required

    append(entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs)
        append entity to chain, ensuring contiguity :param entity: Entity to append :param tol: float tolerance
        on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param
        mergeable: function of the form f(e1,e2) returning True if entities e1,e2 can be merged :param attrs:
        attributes passed to Group.append :return: self, or None if edge is not contiguous

    static from_pdf(path, trans, color)

        Parameters path – pdf path

        Returns Entity of correct subtype

        See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000\_2008.pdf p.
        132

    static from_svg(path, color)

        Parameters path – svg path

        Returns Entity of correct subtype

    static from_dxf(e, mat3)

        Parameters

        • e – dxf.entity

        • mat3 – Matrix3 transform

        Returns Entity of correct subtype

    to_dxf(split=False, **attr)

        Parameters

        • split – bool if True, each segment in Chain is saved separately

        • attr – dict of graphic attributes

        Returns polyline or list of entities along the chain

    __abstractmethods__ = frozenset()

    __add__
        Return self+value.

    __class__
        alias of ABCMeta
```

`__contains__`
Return key in self.

`__copy__` ()

`__delattr__`
Implement delattr(self, name).

`__delitem__`
Delete self[key].

`__dir__` () → list
default dir() implementation

`__eq__`
Return self==value.

`__format__` ()
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__getitem__` ()
x.__getitem__(y) <==> x[y]

`__gt__`
Return self>value.

`__hash__` = None

`__iadd__`
Implement self+=value.

`__imul__`
Implement self*=value.

`__iter__`
Implement iter(self).

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__lt__`
Return self<value.

`__mul__`
Return self*value.n

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

__reduce_ex__()
helper for pickle

__reversed__()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBox* bounding box of Entity

center

chainify (*mergeable*)
merge all possible entities into chains

clear () → None – remove all items from L

color

connect (*other*)

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

extend (*entities, **kwargs*)

figure (*box, **kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to *~matplotlib.pyplot.figure*

Returns matplotlib axis suitable for drawing

html (***kwargs*)

index (*value[, start[, stop]]*) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

intersect (*other*)

Parameters *other* – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

layer

length

patches (***kwargs*)

Returns list of *Patch* corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

pop ([*index*]) → item – remove and return item at index (default last).

Raises *IndexError* if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.

Raises *ValueError* if the value is not present.

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()

L.reverse() – reverse *IN PLACE*

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

svg (***kwargs*)

swap ()

swap start and end

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

Goulib.drawing.Chains (*group, tol=1e-06, mergeable=None*)

build chains from all possible segments in group :param *mergeable*: function(*e1,e2*) returning True if entities *e1,e2* can be merged

class *Goulib.drawing.Rect* (**args*)

Bases: *Goulib.drawing.Chain*

a rectangle starting at low/left and going trigowise through top/right

```
__init__ (*args)
p1
p2
__repr__ ()
__abstractmethods__ = frozenset()
__add__
    Return self+value.
__class__
    alias of ABCMeta
__contains__
    Return key in self.
__copy__ ()
__delattr__
    Implement delattr(self, name).
__delitem__
    Delete self[key].
__dir__ () → list
    default dir() implementation
__eq__
    Return self==value.
__format__ ()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__getitem__ ()
    x.__getitem__(y) <==> x[y]
__gt__
    Return self>value.
__hash__ = None
__iadd__
    Implement self+=value.
__imul__
    Implement self*=value.
__iter__
    Implement iter(self).
__le__
    Return self<=value.
__len__
    Return len(self).
```

__lt__
Return self<value.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__reversed__()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

append (entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs)
append entity to chain, ensuring contiguity :param entity: *Entity* to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param mergeable: function of the form f(e1,e2) returning True if entities e1,e2 can be merged :param attrs: attributes passed to Group.append :return: self, or None if edge is not contiguous

bbox (filter=None)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBox* bounding box of Entity

center

chainify (mergeable)
merge all possible entities into chains

clear () → None – remove all items from L

color

connect (other)

contiguous (edge, tol=1e-06, allow_swap=True)
check if edge can be appended to the chain :param edge: *Entity* to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return: int,bool index where to append in chain, swap of edge required

copy () → list – a shallow copy of L

count (value) → integer – return number of occurrences of value

distance (other)

draw (fig=None, **kwargs)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

extend (entities, **kwargs)

figure (box, **kwargs)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (e, mat3)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (path, trans, color)

Parameters **path** – pdf path

Returns Entity of correct subtype

See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf p. 132

from_svg (path, color)

Parameters **path** – svg path

Returns Entity of correct subtype

html (**kwargs)

index (value[, start[, stop]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (other)

Parameters **other** – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (tol=0.01)

isline ()

isvertical (tol=0.01)

layer

length

patches (***kwargs*)

Returns list of `Patch` corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

pop (*[index]*) → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.

Raises `ValueError` if the value is not present.

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()

`L.reverse()` – reverse *IN PLACE*

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

start

svg (***kwargs*)

swap ()

swap start and end

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (*split=False, **attr*)

Parameters

- **split** – bool if True, each segment in Chain is saved separately
- **attr** – dict of graphic attributes

Returns polyline or list of entities along the chain

class `Goulib.drawing.Text` (*text, point, size=12, rotation=0*)

Bases: `Goulib.drawing.Entity`

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

`__init__` (*text*, *point*, *size*=12, *rotation*=0)

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

bbox ()

length

Returns float length of the text contour in mm

intersect (*other*)

to_dxf (***attr*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

helper for pickle

```

__reduce_ex__ ()
    helper for pickle

__repr__ ()

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

center

color = 'black'

draw (fig=None, **kwargs)
    draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (box, **kwargs)

    Parameters
        • box – drawing.BBox bounds and clipping box
        • kwargs – parameters passed to ~matplotlib.pyplot.figure

    Returns matplotlib axis suitable for drawing

from_dxf (e, mat3)

    Parameters
        • e – dxf.entity
        • mat3 – Matrix3 transform

    Returns Entity of correct subtype

from_pdf (path, trans, color)

    Parameters path – pdf path

    Returns Entity of correct subtype

from_svg (path, color)

    Parameters path – svg path

    Returns Entity of correct subtype

html (**kwargs)

isclosed ()

ishorizontal (tol=0.01)

isline ()

isvertical (tol=0.01)

plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

```

render (*fmt*, ***kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename*, ***kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

class Goulib.drawing.**Drawing** (*data=[]*, ***kwargs*)

Bases: *Goulib.drawing.Group*

list of Entities representing a vector graphics drawing

__abstractmethods__ = frozenset()

__add__

Return self+value.

__class__

alias of ABCMeta

__contains__

Return key in self.

__copy__ ()

__delattr__

Implement delattr(self, name).

__delitem__

Delete self[key].

__dir__ () → list

default dir() implementation

__eq__

Return self==value.

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__getitem__ ()

x.__getitem__(y) <==> x[y]

__gt__

Return self>value.

__hash__ = None

__iadd__

Implement self+=value.

__imul__

Implement self*=value.

__init__ (*data=[]*, ***kwargs*)

```

__iter__
    Implement iter(self).

__le__
    Return self<=value.

__len__
    Return len(self).

__lt__
    Return self<value.

__mul__
    Return self*value.n

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__reversed__ ()
    L.__reversed__() – return a reverse iterator over the list

__rmul__
    Return self*value.

__setattr__
    Implement setattr(self, name, value).

__setitem__
    Set self[key] to value.

__sizeof__ ()
    L.__sizeof__() – size of L in memory, in bytes

__str__
    Return str(self).

append (entity, **kwargs)
    append entity to group :param entity: Entity :param kwargs: dict of attributes copied to entity :return:
    Group (or Chain) to which the entity was added, or None if entity was None

bbox (filter=None)

    Parameters filter – optional function(entity):bool returning True if entity should be consid-
        ered in box

    Returns BBBox bounding box of Entity

center

chainify (mergeable)
    merge all possible entities into chains

clear () → None – remove all items from L

```

color

connect (*other*)

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None, **kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

extend (*entities, **kwargs*)

figure (*box, **kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*dxf, layers=None, only=[], ignore=['POINT'], trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0), flatten=False*)

Parameters

- **dxf** – dxf.entity
- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read
- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

index (*value[, start[, stop]]*) → integer – return first index of value.
Raises `ValueError` if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (*other*)

Parameters *other* – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

layer

length

patches (***kwargs*)

Returns list of *Patch* corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

pop (*[index]*) → item – remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.

Raises ValueError if the value is not present.

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()

L.reverse() – reverse *IN PLACE*

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

start

svg (***kwargs*)

swap ()

swap start and end

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***kwargs*)

Returns flatten list of dxf entities

load (*filename, **kwargs*)

read_pdf (*filename, **kwargs*)

reads a vector graphics on a .pdf file only the first page is parsed

read_svg (*content, **kwargs*)

appends svg content to drawing :param content: string, either filename or svg content

read_dxf (*filename, options=None, **kwargs*)
reads a .dxf file :param filename: string path to .dxf file to read :param options: passed to `from_dxf`

save (*filename, **kwargs*)
save graph in various formats

2.6 Goulib.expr module

simple symbolic math expressions

`Goulib.expr.add_function` (*f, s=None, r=None, l=None*)
add a function to those allowed in Expr.

Parameters

- **f** – function
- **s** – string representation, should be formula-like
- **r** – repr representation, should be cut&pastable in a calculator, or in python ...
- **l** – LaTeX representation

`Goulib.expr.add_module` (*module*)

`Goulib.expr.eval` (*node, ctx={}*)
safe eval of ast node : only `_functions` and `_operators` listed above can be used

Parameters

- **node** – ast.AST to evaluate
- **ctx** – dict of varname : value to substitute in node

Returns number or expression string

`Goulib.expr.get_function_source` (*f*)
returns cleaned code of a function or lambda currently only supports: - `lambda x:formula_of_(x)` - `def anything(x): return formula_of_(x)`

`Goulib.expr.plouffe` (*f, epsilon=1e-06*)

class `Goulib.expr.Expr` (*f*)
Bases: `Goulib.plot.Plot`

Math expressions that can be evaluated like standard functions combined using standard operators and plotted in IPython/Jupyter notebooks

Parameters **f** – function or operator, Expr to copy construct, or formula string

`__init__` (*f*)

Parameters **f** – function or operator, Expr to copy construct, or formula string

isNum

isconstant

Returns True if Expr evaluates to a constant number or bool

`__call__` (*x=None, **kwargs*)
evaluate the Expr at x OR compose self(x())

`__float__` ()

```

__repr__()
__str__()
latex()
    Returns string LaTeX formula
apply(f, right=None)
    function composition self o f = f(self(x))
applx(f, var='x')
    function composition f o self = self(f(x))
__eq__(other)
__ne__(other)
__lt__(other)
__le__(other)
__ge__(other)
__gt__(other)
__add__(right)
__sub__(right)
__neg__()
__mul__(right)
__rmul__(right)
__truediv__(right)
__pow__(right)
__div__(right)
__invert__()
__and__(right)
__or__(right)
__xor__(right)
__lshift__(dx)
__rshift__(dx)
complexity()
    measures the complexity of Expr :return: int, sum of the precedence of used ops
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__() → list
    default dir() implementation
__format__()
    default object formatter

```

__getattr__
Return `getattr(self, name)`.

__hash__ = None

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

html (***kwargs*)

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

class `Goulib.expr.TextVisitor` (*dialect*)

Bases: `ast.NodeVisitor`

Parameters **dialect** – int index in `_operators` of symbols to use

__init__ (*dialect*)

Parameters **dialect** – int index in `_operators` of symbols to use

prec (*op*)
calculate the precedence of *op*

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

visit (*node*)
Visit a node.

visit_Call (*n*)

visit_Name (*n*)

visit_NameConstant (*node*)

visit_UnaryOp (*n*)

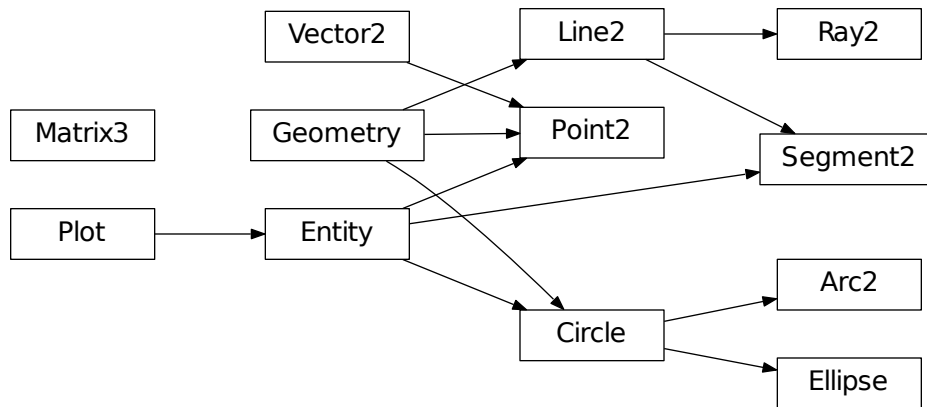
visit_BinOp (*n*)

visit_Compare (*n*)

visit_Num (*n*)

generic_visit (*n*)

2.7 Goulib.geom module



2D ge-

ometry

class Goulib.geom.**Geometry** (*args)
 Bases: `object`

The following classes are available for dealing with simple 2D geometry. The interface to each shape is similar; in particular, the `connect` and `distance` methods are defined identically for each.

For example, to find the closest point on a line to a circle:

```
>>> circ = Circle(Point2(3., 2.), 2.)
>>> line = Line2(Point2(0., 0.), Point2(-1., 1.))
>>> line.connect(circ).p1
Point2(0.50, -0.50)
```

To find the corresponding closest point on the circle to the line:

```
>>> line.connect(circ).p2
Point2(1.59, 0.59)
```

this constructor is called by descendant classes at copy it is replaced to copy some graphics attributes in module drawings

__init__ (*args)

this constructor is called by descendant classes at copy it is replaced to copy some graphics attributes in module drawings

point (u)

Returns Point2 or Point3 at parameter u

tangent (u)

Returns Vector2 or Vector3 tangent at parameter u

intersect (other)

connect (other)

Returns Geometry shortest (Segment2 or Segment3) that connects self to other

```

distance (other)
__contains__ (pt)
__abstractmethods__ = frozenset()
__class__
    alias of ABCMeta
__delattr__
    Implement delattr(self, name).
__dir__ () → list
    default dir() implementation
__eq__
    Return self==value.
__format__ ()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__
    Return hash(self).
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__ ()
    Create and return a new object. See help(type) for accurate signature.
__reduce__ ()
    helper for pickle
__reduce_ex__ ()
    helper for pickle
__repr__
    Return repr(self).
__setattr__
    Implement setattr(self, name, value).
__sizeof__ () → int
    size of object in memory, in bytes
__str__
    Return str(self).

```

Goulib.geom.**argPair** (*x*, *y=None*)
 Process a pair of values passed in various ways.

class Goulib.geom.**Vector2**(*args)

Bases: `object`

Mutable 2D vector:

Construct a vector in the obvious way:

```
>>> Vector2(1.5, 2.0)
Vector2(1.50, 2.00)

>>> Vector3(1.0, 2.0, 3.0)
Vector3(1.00, 2.00, 3.00)
```

Element access

Components may be accessed as attributes (examples that follow use *Vector3*, but all results are similar for *Vector2*, using only the *x* and *y* components):

```
>>> v = Vector3(1, 2, 3)
>>> v.x
1
>>> v.y
2
>>> v.z
3
```

Vectors support the list interface via slicing:

```
>>> v = Vector3(1, 2, 3)
>>> len(v)
3
>>> v[0]
1
>>> v[:]
(1, 2, 3)
```

You can also “swizzle” the components (*a la* GLSL or Cg):

```
>>> v = Vector3(1, 2, 3)
>>> v.xyz
(1, 2, 3)
>>> v.zx
(3, 1)
>>> v.zzzz
(3, 3, 3, 3)
```

Operators

Addition and subtraction are supported via operator overloading (note that in-place operators perform faster than those that create a new object):

```
>>> v1 = Vector3(1, 2, 3)
>>> v2 = Vector3(4, 5, 6)
>>> v1 + v2
Vector3(5.00, 7.00, 9.00)
>>> v1 -= v2
>>> v1
Vector3(-3.00, -3.00, -3.00)
```

Multiplication and division can be performed with a scalar only:

```
>>> Vector3(1, 2, 3) * 2
Vector3(2.00, 4.00, 6.00)
>>> v1 = Vector3(1., 2., 3.)
>>> v1 /= 2
>>> v1
Vector3(0.50, 1.00, 1.50)
```

The magnitude of a vector can be found with `abs`:

```
>>> v = Vector3(1., 2., 3.)
>>> abs(v)
3.7416573867739413
```

A vector can be normalized in-place (note that the in-place method also returns `self`, so you can chain it with further operators):

```
>>> v = Vector3(1., 2., 3.)
>>> v.normalize()
Vector3(0.27, 0.53, 0.80)
>>> v
Vector3(0.27, 0.53, 0.80)
```

The following methods do *not* alter the original vector or their arguments:

magnitude() Returns the magnitude of the vector; equivalent to `abs(v)`. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.magnitude()
3.7416573867739413
```

magnitude_squared() Returns the sum of the squares of each component. Useful for comparing the length of two vectors without the expensive square root operation. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.magnitude_squared()
14.0
```

normalized() Return a unit length vector in the same direction. Note that this method differs from `normalize` in that it does not modify the vector in-place. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.normalized()
Vector3(0.27, 0.53, 0.80)
>>> v
Vector3(1.00, 2.00, 3.00)
```

dot(other) Return the scalar “dot” product of two vectors. Example:

```
>>> v1 = Vector3(1., 2., 3.)
>>> v2 = Vector3(4., 5., 6.)
>>> v1.dot(v2)
32.0
```

cross() and **cross(other)** Return the cross product of a vector (for **Vector2**), or the cross product of two vectors (for **Vector3**). The return type is a vector. Example:

```
>>> v1 = Vector3(1., 2., 3.)
>>> v2 = Vector3(4., 5., 6.)
```

```
>>> v1.cross(v2)
Vector3(-3.00, 6.00, -3.00)
```

In two dimensions there can be no argument to `cross`:

```
>>> v1 = Vector2(1., 2.)
>>> v1.cross()
Vector2(2.00, -1.00)
```

reflect(normal) Return the vector reflected about the given normal. In two dimensions, *normal* is the normal to a line, in three dimensions it is the normal to a plane. The normal must have unit length. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.reflect(Vector3(0, 1, 0))
Vector3(1.00, -2.00, 3.00)
>>> v = Vector2(1., 2.)
>>> v.reflect(Vector2(1, 0))
Vector2(-1.00, 2.00)
```

rotate_around(axes, theta) For 3D vectors, return the vector rotated around axis by the angle theta.

```
>>> v = Vector3(1., 2., 3.)
>>> axes = Vector3(1., 1., 0)
>>> v.rotate_around(axes, math.pi/4)
Vector3(2.65, 0.35, 2.62)
```

Constructor. :param **args*: x,y values

`__init__` (**args*)

Constructor. :param **args*: x,y values

xy

Returns tuple (x,y)

`__repr__` ()

`__hash__` ()

`__eq__` (*other*)

Tests for equality include comparing against other sequences:

```
>>> v2 = Vector2(1, 2)
>>> v2 == Vector2(3, 4)
```

False >>> v2 != Vector2(1, 2) False >>> v2 == (1, 2) True

```
>>> v3 = Vector3(1, 2, 3)
>>> v3 == Vector3(3, 4, 5)
False
>>> v3 != Vector3(1, 2, 3)
False
>>> v3 == (1, 2, 3)
True
```

`__len__` ()

`__iter__` ()

```

__add__(other)
__radd__(other)
__iadd__(other)
__sub__(other)
__rsub__(other)
    Point2 - Vector 2 subtraction :param other: Point2 or (x,y) tuple :return: Vector2
__mul__(other)
__rmul__(other)
__imul__(other)
__div__(other)
__rdiv__(other)
__floordiv__(other)
__rfloordiv__(other)
__truediv__(other)
__rtruediv__(other)
__neg__()
__pos__()
__abs__()
mag()
length
mag2()
normalize()
normalized()
dot(other)
cross()
reflect(normal)
angle(other=None, unit=False)
    angle between two vectors. :param unit: bool True if vectors are unit vectors. False increases computations
    :return: float angle in radians to the other vector, or self direction if other=None
project(other)
    Return the projection (the component) of the vector on other.
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__() → list
    default dir() implementation
__format__()
    default object formatter

```

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class Goulib.geom.**Point2** (*args)

Bases: Goulib.geom.Vector2, *Goulib.geom.Geometry*, *Goulib.drawing.Entity*

A point on a 2D plane. Construct in the obvious way:

```
>>> p = Point2(1.0, 2.0)
>>> p
```

Point2(1.00, 2.00)

Point2 subclasses **Vector2**, so all of **Vector2** operators and methods apply. In particular, subtracting two points gives a vector:

```
>>> Point2(2.0, 3.0) - Point2(1.0, 0.0)
```

Vector2(1.00, 3.00)

connect (other) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

Constructor. :param *args: x,y values

distance (other)

absolute minimum distance to other object :param other: Point2, Line2 or Circle :return: float positive distance between self and other

__contains__ (pt)

Returns True if self and pt are the same point, False otherwise

needed for coherency

intersect (*other*)

Point2/object intersection :return: Point2 copy of self if on other object, None if not

connect (*other*)

__abs__ ()

__abstractmethods__ = frozenset()

__add__ (*other*)

__class__

alias of ABCMeta

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__div__ (*other*)

__eq__ (*other*)

Tests for equality include comparing against other sequences:

```
>>> v2 = Vector2(1, 2)
>>> v2 == Vector2(3, 4)
```

```
False >>> v2 != Vector2(1, 2) False >>> v2 == (1, 2) True
```

```
>>> v3 = Vector3(1, 2, 3)
>>> v3 == Vector3(3, 4, 5)
False
>>> v3 != Vector3(1, 2, 3)
False
>>> v3 == (1, 2, 3)
True
```

__floordiv__ (*other*)

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__ ()

__iadd__ (*other*)

__imul__ (*other*)

__init__ (*args)

Constructor. :param *args: x,y values

`__iter__()`

`__le__`
Return self<=value.

`__len__()`

`__lt__`
Return self<value.

`__mul__(other)`

`__ne__`
Return self!=value.

`__neg__()`

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__pos__()`

`__radd__(other)`

`__rdiv__(other)`

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__()`

`__rfloordiv__(other)`

`__rmul__(other)`

`__rsub__(other)`
Point2 - Vector 2 substraction :param other: Point2 or (x,y) tuple :return: Vector2

`__rtruediv__(other)`

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

`__sub__(other)`

`__truediv__(other)`

angle (other=None, unit=False)
angle between two vectors. :param unit: bool True if vectors are unit vectors. False increases computations
:return: float angle in radians to the other vector, or self direction if other=None

bbox ()

Returns BB_{OX} bounding box of Entity

center

color = 'black'

```

cross ()
dot (other)
draw (fig=None, **kwargs)
    draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch
end
figure (box, **kwargs)
    Parameters
        • box – drawing.BBox bounds and clipping box
        • kwargs – parameters passed to ~matplotlib.pyplot.figure
    Returns matplotlib axis suitable for drawing
from_dxf (e, mat3)
    Parameters
        • e – dxf.entity
        • mat3 – Matrix3 transform
    Returns Entity of correct subtype
from_pdf (path, trans, color)
    Parameters path – pdf path
    Returns Entity of correct subtype
from_svg (path, color)
    Parameters path – svg path
    Returns Entity of correct subtype
html (**kwargs)
isclosed ()
ishorizontal (tol=0.01)
isline ()
isvertical (tol=0.01)
length
mag ()
mag2 ()
normalize ()
normalized ()
patches (**kwargs)
    Returns list of (a single) Patch corresponding to entity
    Note this is the only method that needs to be overridden in descendants for draw, render and
        IPython _repr_xxx_ to work
plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)

```

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

project (*other*)

Return the projection (the component) of the vector on *other*.

reflect (*normal*)

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

xy

Returns tuple (x,y)

Goulib.geom.**Polar** (*mag, angle*)

class Goulib.geom.**Line2** (**args*)

Bases: *Goulib.geom.Geometry*

A **Line2** is a line on a 2D plane extending to infinity in both directions; a **Ray2** has a finite end-point and extends to infinity in a single direction; a **Segment2** joins two points.

All three classes support the same constructors, operators and methods, but may behave differently when calculating intersections etc.

You may construct a line, ray or line segment using any of:

- another line, ray or line segment
- two points
- a point and a vector
- a point, a vector and a length

For example:

```
>>> Line2(Point2(1.0, 1.0), Point2(2.0, 3.0))
Line2(<1.00, 1.00> + u<1.00, 2.00>)
>>> Line2(Point2(1.0, 1.0), Vector2(1.0, 2.0))
Line2(<1.00, 1.00> + u<1.00, 2.00>)
>>> Ray2(Point2(1.0, 1.0), Vector2(1.0, 2.0), 1.0)
Ray2(<1.00, 1.00> + u<0.45, 0.89>)
```

Internally, lines, rays and line segments store a `Point2` p and a `Vector2` v . You can also access (but not set) the two endpoints $p1$ and $p2$. These may or may not be meaningful for all types of lines.

The following methods are supported by all three classes:

intersect (*other*) If *other* is a **Line2**, **Ray2** or **Segment2**, returns a **Point2** of intersection, or `None` if the lines are parallel.

If *other* is a **Circle**, returns a **Segment2** or **Point2** giving the part of the line that intersects the circle, or `None` if there is no intersection.

connect (*other*) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. For two parallel lines, this line segment may be in an arbitrary position. *other* may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

distance (*other*) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

Segment2 also has a *length* property which is read-only.

`__init__` (*args)

`__eq__` (*other*)

lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use `line1.distance(line2)==0`

`__repr__` ()

point (*u*)

Returns `Point2` at parameter *u*

tangent (*u*)

Returns `Vector2` tangent at parameter *u*. Warning : tangent is generally not a unit vector

intersect (*other*)

connect (*other*)

`__abstractmethods__` = frozenset()

`__class__`

alias of `ABCMeta`

`__contains__` (*pt*)

`__delattr__`

Implement `delattr`(self, name).

`__dir__` () → list

default `dir()` implementation

`__format__` ()

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr`(self, name).

`__gt__`

Return `self>value`.

`__hash__` = `None`

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

distance (other)

class Goulib.geom.**Ray2** (*args)
Bases: *Goulib.geom.Line2*

__abstractmethods__ = frozenset()

__class__
alias of ABCMeta

__contains__ (pt)

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__ (other)
lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use
line1.distance(line2)==0

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__ = None

__init__ (*args)

```

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__ ()

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

connect (other)

distance (other)

intersect (other)

point (u)

    Returns Point2 at parameter u

tangent (u)

    Returns Vector2 tangent at parameter u. Warning : tangent is generally not a unit vector

class Goulib.geom.Segment2 (*args)
    Bases: Goulib.geom.Line2, Goulib.drawing.Entity

    p1
    p2

    __repr__ ()

    __abs__ ()

    mag2 ()

    length

    swap ()

    midpoint ()

    bisect ()

    __abstractmethods__ = frozenset()

    __class__
        alias of ABCMeta

```

__contains__ (*pt*)

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__ (*other*)
lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use
line1.distance(line2)==0

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__ = None

__init__ (*args)

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

bbox ()

Returns BB_{OX} bounding box of Entity

center

color = ‘black’

connect (*other*)

distance (*other*)

draw (*fig=None, **kwargs*)
 draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box, **kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)
 renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 at parameter u

render (*fmt, **kwargs*)
 render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 tangent at parameter u. Warning : tangent is generally not a unit vector

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class Goulib.geom.**Circle** (**args*)

Bases: *Goulib.geom.Geometry, Goulib.drawing.Entity*

Circles are constructed with a center **Point2** and a radius:

```
>>> c = Circle(Point2(1.0, 1.0), 0.5)
>>> c
```

Circle(<1.00, 1.00>, radius=0.50)

Internally there are two attributes: *c*, giving the center point and *r*, giving the radius.

The following methods are supported:

connect (*other*) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

distance (*other*) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

Parameters *args* – can be

- Circle
- center, point on circle
- center, radius

__init__ (**args*)

Parameters *args* – can be

- Circle
- center, point on circle
- center, radius

__eq__ (*other*)

__repr__ ()

__abs__ ()

Returns float perimeter

length

point (*u*)

Returns Point2 at angle u radians

tangent (u)

Returns Vector2 tangent at angle u . Warning : tangent has magnitude $r \neq 1$

__contains__ (pt)

Returns True if pt is ON or IN the circle

intersect ($other$)

Parameters **other** – Line2, Ray2 or Segment2**, **Ray2** or **Segment2**, returns

a **Segment2** giving the part of the line that intersects the circle, or None if there is no intersection.

connect ($other$)

swap ()

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__ = None

__le__

Return self<=value.

__lt__

Return self<value.

__ne__

Return self!=value.

__new__ ()

Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

helper for pickle

__reduce_ex__ ()

helper for pickle

__setattr__

Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

bbox ()
Returns BBox bounding box of Entity

center

color = 'black'

distance (other)

draw (fig=None, **kwargs)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (box, **kwargs)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to *~matplotlib.pyplot.figure*

Returns matplotlib axis suitable for drawing

from_dxf (e, mat3)
Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (path, trans, color)
Parameters **path** – pdf path
Returns Entity of correct subtype

from_svg (path, color)
Parameters **path** – svg path
Returns Entity of correct subtype

html (**kwargs)

isclosed ()

ishorizontal (tol=0.01)

isline ()

isvertical (tol=0.01)

patches (**kwargs)
Returns list of (a single) Patch corresponding to entity
Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt, **kwargs*)
render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)
set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

`Goulib.geom.circle_from_3_points(a, b, c)`
constructs Circle passing through 3 distinct points :param a,b,c: Point2 :return: the unique Circle through the three points a, b, c

`Goulib.geom.arc_from_3_points(a, b, c)`
constructs Arc2 starting in a, going through b and ending in c :param a,b,c: Point2 :return: the unique Arc2 starting in a, going through b and ending in c

class `Goulib.geom.Arc2` (*center, p1=0, p2=6.283185307179586, r=None, dir=1*)

Bases: `Goulib.geom.Circle`

Parameters

- **center** – Point2 or (x,y) tuple
- **p1** – starting Point2 or angle in radians
- **p2** – ending Point2 or angle in radians
- **r** – float radius, needed only if p1 or p2 is an angle
- **dir** – arc direction. +1 is trig positive (CCW) and -1 is Clockwise

__init__ (*center, p1=0, p2=6.283185307179586, r=None, dir=1*)

Parameters

- **center** – Point2 or (x,y) tuple
- **p1** – starting Point2 or angle in radians
- **p2** – ending Point2 or angle in radians
- **r** – float radius, needed only if p1 or p2 is an angle
- **dir** – arc direction. +1 is trig positive (CCW) and -1 is Clockwise

angle (*b=None*)

Returns float signed arc angle

__abs__ ()

Returns float arc length

point (*u*)

Returns Point2 at parameter u

tangent (*u*)

Returns Vector2 tangent at parameter u

__eq__ (*other*)

__repr__ ()

swap ()

__contains__ (*pt*)

Returns True if pt is ON the Arc

intersect (*other*)

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__ = None

__le__

Return self<=value.

__lt__

Return self<value.

__ne__

Return self!=value.

__new__ ()

Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

helper for pickle

__reduce_ex__ ()

helper for pickle

__setattr__

Implement setattr(self, name, value).

__sizeof__ () → int

size of object in memory, in bytes

__str__
Return str(self).

bbox()
Returns BBox bounding box of Entity

center

color = 'black'

connect(other)

distance(other)

draw (fig=None, **kwargs)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (box, **kwargs)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (e, mat3)
Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (path, trans, color)
Parameters **path** – pdf path
Returns Entity of correct subtype

from_svg (path, color)
Parameters **path** – svg path
Returns Entity of correct subtype

html (**kwargs)

isclosed()

ishorizontal (tol=0.01)

isline()

isvertical (tol=0.01)

length

patches (**kwargs)
Returns list of (a single) Patch corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_xxx_` to work

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt, **kwargs*)
render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)
set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class Goulib.geom.**Ellipse** (**args*)
Bases: *Goulib.geom.Circle*

Parameters **args** – can be

- Ellipse
- center, corner point
- center, r1,r2,angle

__init__ (**args*)

Parameters **args** – can be

- Ellipse
- center, corner point
- center, r1,r2,angle

__repr__ ()

__eq__ (*other*)

__abs__ ()

Returns float perimeter

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__contains__ (*pt*)

Returns True if pt is ON or IN the circle

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

```

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__ = None

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

bbox ()
    Returns BBox bounding box of Entity

center

color = 'black'

connect (other)

distance (other)

draw (fig=None, **kwargs)
    draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (box, **kwargs)
    Parameters
    • box – drawing.BBox bounds and clipping box
    • kwargs – parameters passed to ~matplotlib.pyplot.figure
    Returns matplotlib axis suitable for drawing

```

from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

Parameters **other** – Line2, Ray2 or Segment2**, **Ray2** or **Segment2**, returns

a **Segment2** giving the part of the line that intersects the circle, or None if there is no intersection.

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of (a single) **Patch** corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 at angle u radians

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

swap ()

tangent (*u*)

Returns Vector2 tangent at angle *u*. Warning : tangent has magnitude $r \neq 1$

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class Goulib.geom.**Matrix3** (**args*)

Bases: `object`

Two matrix classes are supplied, *Matrix3*, a 3x3 matrix for working with 2D affine transformations, and *Matrix4*, a 4x4 matrix for working with 3D affine transformations.

The default constructor initializes the matrix to the identity:

```
>>> Matrix3()
Matrix3([ 1.00  0.00  0.00
          0.00  1.00  0.00
          0.00  0.00  1.00])
>>> Matrix4()
Matrix4([ 1.00  0.00  0.00  0.00
          0.00  1.00  0.00  0.00
          0.00  0.00  1.00  0.00
          0.00  0.00  0.00  1.00])
```

Element access

Internally each matrix is stored as a set of attributes named *a* to *p*. The layout for *Matrix3* is:

```
# a b c
# e f g
# i j k
```

and for *Matrix4*:

```
# a b c d
# e f g h
# i j k l
# m n o p
```

If you wish to set or retrieve a number of elements at once, you can do so with a slice:

```
>>> m = Matrix4()
>>> m[:]
[1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0]
>>> m[12:15] = (5, 5, 5)
>>> m
Matrix4([ 1.00  0.00  0.00  5.00
          0.00  1.00  0.00  5.00
          0.00  0.00  1.00  5.00
          0.00  0.00  0.00  1.00])
```

Note that slices operate in column-major order, which makes them suitable for working directly with OpenGL's `glLoadMatrix` and `glGetFloatv` functions.

Class constructors

There are class constructors for the most common types of transform.

new_identity Equivalent to the default constructor. Example:

```
>>> m = Matrix4.new_identity()
>>> m
Matrix4([ [ 1.00    0.00    0.00    0.00
            0.00    1.00    0.00    0.00
            0.00    0.00    1.00    0.00
            0.00    0.00    0.00    1.00]])
```

new_scale(x, y) and **new_scale(x, y, z)** The former is defined on **Matrix3**, the latter on **Matrix4**. Equivalent to the OpenGL call `glScalef`. Example:

```
>>> m = Matrix4.new_scale(2.0, 3.0, 4.0)
>>> m
Matrix4([ [ 2.00    0.00    0.00    0.00
            0.00    3.00    0.00    0.00
            0.00    0.00    4.00    0.00
            0.00    0.00    0.00    1.00]])
```

new_translate(x, y) and **new_translate(x, y, z)** The former is defined on **Matrix3**, the latter on **Matrix4**. Equivalent to the OpenGL call `glTranslatef`. Example:

```
>>> m = Matrix4.new_translate(3.0, 4.0, 5.0)
>>> m
Matrix4([ [ 1.00    0.00    0.00    3.00
            0.00    1.00    0.00    4.00
            0.00    0.00    1.00    5.00
            0.00    0.00    0.00    1.00]])
```

new_rotate(angle) Create a **Matrix3** for a rotation around the origin. *angle* is specified in radians, anti-clockwise. This is not implemented in **Matrix4** (see below for equivalent methods). Example:

```
>>> import math
>>> m = Matrix3.new_rotate(math.pi / 2)
>>> m
Matrix3([ [ 0.00   -1.00    0.00
            1.00    0.00    0.00
            0.00    0.00    1.00]])
```

The following constructors are defined for **Matrix4** only.

new_rotatex(angle), **new_rotatey(angle)**, **new_rotatez(angle)** Create a **Matrix4** for a rotation around the X, Y or Z axis, respectively. *angle* is specified in radians. Example:

```
>>> m = Matrix4.new_rotatex(math.pi / 2)
>>> m
Matrix4([ [ 1.00    0.00    0.00    0.00
            0.00    0.00   -1.00    0.00
            0.00    1.00    0.00    0.00
            0.00    0.00    0.00    1.00]])
```

new_rotate_axis(angle, axis) Create a **Matrix4** for a rotation around the given axis. *angle* is specified in radians, and *axis* must be an instance of **Vector3**. It is not necessary to normalize the axis. Example:

```
>>> m = Matrix4.new_rotate_axis(math.pi / 2, Vector3(1.0, 0.0, 0.0))
>>> m
Matrix4([ [ 1.00    0.00    0.00    0.00
            0.00    0.00   -1.00    0.00
```

```

0.00    1.00    0.00    0.00
0.00    0.00    0.00    1.00])

```

new_rotate_euler(heading, attitude, bank) Create a **Matrix4** for the given Euler rotation. *heading* is a rotation around the Y axis, *attitude* around the X axis and *bank* around the Z axis. All rotations are performed simultaneously, so this method avoids “gimbal lock” and is the usual method for implemented 3D rotations in a game. Example:

```

>>> m = Matrix4.new_rotate_euler(math.pi / 2, math.pi / 2, 0.0)
>>> m
Matrix4([
    0.00    -0.00    1.00    0.00
    1.00     0.00   -0.00    0.00
   -0.00     1.00     0.00    0.00
    0.00     0.00     0.00    1.00])

```

new_perspective(fov_y, aspect, near, far) Create a **Matrix4** for projection onto the 2D viewing plane. This method is equivalent to the OpenGL call `gluPerspective`. *fov_y* is the view angle in the Y direction, in radians. *aspect* is the aspect ration *width / height* of the viewing plane. *near* and *far* are the distance to the near and far clipping planes. They must be positive and non-zero. Example:

```

>>> m = Matrix4.new_perspective(math.pi / 2, 1024.0 / 768, 1.0, 100.0)
>>> m
Matrix4([
    0.75     0.00     0.00     0.00
    0.00     1.00     0.00     0.00
    0.00     0.00    -1.02    -2.02
    0.00     0.00    -1.00     0.00])

```

Operators

Matrices of the same dimension may be multiplied to give a new matrix. For example, to create a transform which translates and scales:

```

>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m2 = Matrix3.new_scale(1.0, 2.0)
>>> m1 * m2
Matrix3([
    1.00     0.00     5.00
    0.00     2.00     6.00
    0.00     0.00     1.00])

```

Note that multiplication is not commutative (the order that you apply transforms matters):

```

>>> m2 * m1
Matrix3([
    1.00     0.00     5.00
    0.00     2.00    12.00
    0.00     0.00     1.00])

```

In-place multiplication is also permitted (and optimised):

```

>>> m1 *= m2
>>> m1
Matrix3([
    1.00     0.00     5.00
    0.00     2.00     6.00
    0.00     0.00     1.00])

```

Multiplying a matrix by a vector returns a vector, and is used to transform a vector:

```
>>> m1 = Matrix3.new_rotate(math.pi / 2)
>>> m1 * Vector2(1.0, 1.0)
Vector2(-1.00, 1.00)
```

Note that translations have no effect on vectors. They do affect points, however:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m1 * Vector2(1.0, 2.0)
Vector2(1.00, 2.00)
>>> m1 * Point2(1.0, 2.0)
Point2(6.00, 8.00)
```

Multiplication is currently incorrect between matrices and vectors – the projection component is ignored. Use the **Matrix4.transform** method instead.

Matrix4 also defines **transpose** (in-place), **transposed** (functional), **determinant** and **inverse** (functional) methods.

A **Matrix3** can be multiplied with a **Vector2** or any of the 2D geometry objects (**Point2**, **Line2**, **Circle**, etc).

A **Matrix4** can be multiplied with a **Vector3** or any of the 3D geometry objects (**Point3**, **Line3**, **Sphere**, etc).

For convenience, each of the matrix constructors are also available as in-place operators. For example, instead of writing:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m2 = Matrix3.new_scale(1.0, 2.0)
>>> m1 *= m2
```

you can apply the scale directly to *m1*:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m1.scale(1.0, 2.0)
Matrix3([ 1.00  0.00  5.00
          0.00  2.00  6.00
          0.00  0.00  1.00])

>>> m1
Matrix3([ 1.00  0.00  5.00
          0.00  2.00  6.00
          0.00  0.00  1.00])
```

Note that these methods operate in-place (they modify the original matrix), and they also return themselves as a result. This allows you to chain transforms together directly:

```
>>> Matrix3().translate(1.0, 2.0).rotate(math.pi / 2).scale(4.0, 4.0)
Matrix3([ 0.00 -4.00  1.00
          4.00  0.00  2.00
          0.00  0.00  1.00])
```

All constructors have an equivalent in-place method. For **Matrix3**, they are `identity`, `translate`, `scale` and `rotate`. For **Matrix4**, they are `identity`, `translate`, `scale`, `rotatex`, `rotatey`, `rotatez`, `rotate_axis` and `rotate_euler`. Both **Matrix3** and **Matrix4** also have an in-place `transpose` method.

The `copy` method is also implemented in both matrix classes and behaves in the obvious way.

```
__init__(*args)
__repr__()
```

```

__iter__()
__getitem__(key)
__setitem__(key, value)
__eq__(other)
__sub__(other)
__imul__(other)
__mul__(other)
__call__(other)
identity()
scale(x, y=None)
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__() → list
    default dir() implementation
__format__()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__ = None
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__()
    Create and return a new object. See help(type) for accurate signature.
__reduce__()
    helper for pickle
__reduce_ex__()
    helper for pickle
__setattr__
    Implement setattr(self, name, value).
__sizeof__() → int
    size of object in memory, in bytes

```

`__slotnames__ = []`

`__str__`

Return str(self).

`offset()`

`angle (angle=0)`

Parameters `angle` – angle in radians of a unit vector starting at origin

Returns float bearing in radians of the transformed vector

`mag (v=None)`

Return the net (uniform) scaling of this transform.

`translate (*args)`

Parameters `*args` – x,y values

`rotate (angle)`

`classmethod new_identity()`

`classmethod new_scale (x, y)`

`classmethod new_translate (x, y)`

`classmethod new_rotate (angle)`

`mag2()`

`__abs__()`

`transpose()`

`transposed()`

`determinant()`

`inverse()`

`orientation()`

Returns 1 if matrix is right handed, -1 if left handed

2.8 Goulib.geom3d module

3D geometry

`class Goulib.geom3d.Vector3 (*args)`

Bases: `object`

Mutable 3D Vector. See ‘Vector2’ documentation

Constructor. :param `*args`: x,y,z values

`__init__ (*args)`

Constructor. :param `*args`: x,y,z values

`xyz`

Returns tuple (x,y,z)

`__repr__()`

`__eq__ (other)`

```
__ne__(other)
__bool__()
__nonzero__()
__len__()
__iter__()
__add__(other)
__radd__(other)
__iadd__(other)
__sub__(other)
__rsub__(other)
__mul__(other)
__rmul__(other)
__imul__(other)
__div__(other)
__rdiv__(other)
__floordiv__(other)
__rfloordiv__(other)
__truediv__(other)
__rtruediv__(other)
__neg__()
__pos__()
__abs__()
mag()
mag2()
normalize()
normalized()
dot(other)
cross(other)
reflect(normal)
rotate_around(axis, theta)
    Return the vector rotated around axis through angle theta. Right hand rule applies
angle(other)
    angle between two vectors. :param other: Vector3 :return: float angle in radians to the other vector, or self
    direction if other=None
project(other)
    Return one vector projected on the vector other
__hash__ = None
```

class Goulib.geom3d.**Point3**(*args)
Bases: *Goulib.geom3d.Vector3, Goulib.geom.Geometry*

A point on a 3D plane. Construct in the obvious way:

```
>>> p = Point3(1.0, 2.0, 3.0)
>>> p
Point3(1.00, 2.00, 3.00)
```

Point3 subclasses **Vector3**, so all of **Vector3** operators and methods apply. In particular, subtracting two points gives a vector:

```
>>> Point3(1.0, 2.0, 3.0) - Point3(1.0, 0.0, -2.0)
Vector3(0.00, 2.00, 5.00)
```

The following methods are also defined:

intersect (*other*) If *other* is a **Sphere**, returns True iff the point lies within the sphere.

connect (*other*) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (*other*) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of **connect**.

Constructor. :param *args: x,y,z values

intersect (*other*)

Point3/object intersection :return: self Point3 if on other object, None if not

connect (*other*)

__abstractmethods__ = frozenset()

class Goulib.geom3d.**Line3**(*args)
Bases: *Goulib.geom.Geometry*

A **Line3** is a line on a 3D plane extending to infinity in both directions; a **Ray3** has a finite end-point and extends to infinity in a single direction; a **LineSegment3** joins two points.

All three classes support the same constructors, operators and methods, but may behave differently when calculating intersections etc.

You may construct a line, ray or line segment using any of:

- another line, ray or line segment
- two points
- a point and a vector
- a point, a vector and a length

For example:

```
>>> Line3(Point3(1.0, 1.0, 1.0), Point3(1.0, 2.0, 3.0))
Line3(<1.00, 1.00, 1.00> + u<0.00, 1.00, 2.00>)
>>> Line3(Point3(0.0, 1.0, 1.0), Vector3(1.0, 1.0, 2.0))
Line3(<0.00, 1.00, 1.00> + u<1.00, 1.00, 2.00>)
>>> Ray3(Point3(1.0, 1.0, 1.0), Vector3(1.0, 1.0, 2.0), 1.0)
Ray3(<1.00, 1.00, 1.00> + u<0.41, 0.41, 0.82>)
```

Internally, lines, rays and line segments store a **Point3** *p* and a **Vector3** *v*. You can also access (but not set) the two endpoints *p1* and *p2*. These may or may not be meaningful for all types of lines.

The following methods are supported by all three classes:

intersect (other) If *other* is a **Sphere**, returns a **LineSegment3** which is the intersection of the sphere and line, or **None** if there is no intersection.

If *other* is a **Plane**, returns a **Point3** of intersection, or **None**.

connect (other) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. For two parallel lines, this line segment may be in an arbitrary position. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of **connect**.

LineSegment3 also has a *length* property which is read-only.

```
__init__ (*args)
```

```
__repr__ ()
```

```
p1
```

```
p2
```

```
point (u)
```

Returns Point3 at parameter u

```
intersect (other)
```

```
connect (other)
```

```
__abstractmethods__ = frozenset()
```

```
class Goulib.geom3d.Ray3 (*args)
```

 Bases: *Goulib.geom3d.Line3*

```
__abstractmethods__ = frozenset()
```

```
class Goulib.geom3d.Segment3 (*args)
```

 Bases: *Goulib.geom3d.Line3*

```
__repr__ ()
```

```
__abs__ ()
```

```
mag2 ()
```

```
swap ()
```

```
length
```

```
__abstractmethods__ = frozenset()
```

```
Goulib.geom3d.Spherical (r, theta, phi)
```

```
class Goulib.geom3d.Sphere (*args)
```

 Bases: *Goulib.geom.Geometry*

Spheres are constructed with a center **Point3** and a radius:

```
>>> s = Sphere(Point3(1.0, 1.0, 1.0), 0.5)
>>> s
```

Sphere(<1.00, 1.00, 1.00>, radius=0.50)

Internally there are two attributes: *c*, giving the center point and *r*, giving the radius.

The following methods are supported:

intersect (other): If *other* is a **Point3**, returns `True` iff the point lies within the sphere.

If *other* is a **Line3**, **Ray3** or **LineSegment3**, returns a **LineSegment3** giving the intersection, or `None` if the line does not intersect the sphere.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

Parameters *args* – can be

- Sphere
- center, point on sphere
- center, radius

__init__ (**args*)

Parameters *args* – can be

- Sphere
- center, point on sphere
- center, radius

__repr__ ()

__contains__ (*pt*)

Returns `True` if *pt* is ON or IN the sphere

point (*u*, *v*)

Parameters

- *u* – float angle from “north pole” (=radians(90-lat) in radians
- *v* – float angle from 0 meridian

Returns **Point3** on sphere at specified coordinates

intersect (*other*)

connect (*other*)

minimal joining segment between Sphere and other 3D Object :param other: **Point3**, **Line3**, **Sphere**, **Plane**
:return: **LineSegment3** of minimal length

distance_on_sphere (*phi1*, *theta1*, *phi2*, *theta2*)

Parameters

- *phi1* – float angle from “north pole” (=radians(90-lat) in radians
- *theta1* – float angle from 0 meridian
- *phi2* – float angle from “north pole” (=radians(90-lat) in radians
- *theta2* – float angle from 0 meridian

__abstractmethods__ = **frozenset**()

```
class Goulib.geom3d.Plane(*args)
    Bases: Goulib.geom.Geometry
```

Planes can be constructed with any of:

- three **Point3**'s lying on the plane
- a **Point3** on the plane and the **Vector3** normal
- a **Vector3** normal and k , described below.

Internally, planes are stored with the normal n and constant k such that $n.p = k$ for any point on the plane p .

The following methods are supported:

intersect (other) If *other* is a **Line3**, **Ray3** or **LineSegment3**, returns a **Point3** of intersection, or **None** if there is no intersection.

If *other* is a **Plane**, returns the **Line3** of intersection.

connect (other) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of **connect**.

```
__init__ (*args)
__repr__ ()
intersect (other)
connect (other)
__abstractmethods__ = frozenset()
```

```
class Goulib.geom3d.Matrix4(*args)
    Bases: object
```

```
__init__ (*args)
__repr__ ()
__iter__ ()
__getitem__ (key)
__setitem__ (key, value)
__mul__ (other)
__call__ (other)
__imul__ (other)
transform (other)
identity ()
scale (x, y, z)
translate (x, y, z)
rotatex (angle)
rotatey (angle)
rotatez (angle)
rotate_axis (angle, axis)
```

```
rotate_euler (heading, attitude, bank)  
rotate_triple_axis (x, y, z)  
transpose ()  
transposed ()  
classmethod new (*values)  
classmethod new_identity ()  
classmethod new_scale (x, y, z)  
classmethod new_translate (x, y, z)  
classmethod new_rotatex (angle)  
classmethod new_rotatey (angle)  
classmethod new_rotatez (angle)  
classmethod new_rotate_axis (angle, axis)  
classmethod new_rotate_euler (heading, attitude, bank)  
classmethod new_rotate_triple_axis (x, y, z)  
classmethod new_look_at (eye, at, up)  
classmethod new_perspective (fov_y, aspect, near, far)  
determinant ()  
inverse ()
```

class Goulib.geom3d.**Quaternion** (*w=1, x=0, y=0, z=0*)

Bases: `object`

A quaternion represents a three-dimensional rotation or reflection transformation. They are the preferred way to store and manipulate rotations in 3D applications, as they do not suffer the same numerical degradation that matrices do.

The quaternion constructor initializes to the identity transform:

```
>>> q = Quaternion()  
>>> q  
Quaternion(real=1.00, imag=<0.00, 0.00, 0.00>)
```

Element access

Internally, the quaternion is stored as four attributes: `x`, `y` and `z` forming the imaginary vector, and `w` the real component.

Constructors

Rotations can be formed using the constructors:

new_identity() Equivalent to the default constructor.

new_rotate_axis(*angle, axis*) Equivalent to the `Matrix4` constructor of the same name. *angle* is specified in radians, *axis* is an instance of **Vector3**. It is not necessary to normalize the axis. Example:

```
>>> q = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))  
>>> q  
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
```

new_rotate_euler(heading, attitude, bank) Equivalent to the `Matrix4` constructor of the same name. *heading* is a rotation around the Y axis, *attitude* around the X axis and *bank* around the Z axis. All angles are given in radians. Example:

```
>>> q = Quaternion.new_rotate_euler(math.pi / 2, math.pi / 2, 0)
>>> q
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
```

new_interpolate(q1, q2, t) Create a quaternion which gives a (SLERP) interpolated rotation between *q1* and *q2*. *q1* and *q2* are instances of **Quaternion**, and *t* is a value between 0.0 and 1.0. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q2 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> for i in range(11):
...     print Quaternion.new_interpolate(q1, q2, i / 10.0)
...
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
Quaternion(real=0.75, imag=<0.66, 0.09, 0.00>)
Quaternion(real=0.78, imag=<0.61, 0.17, 0.00>)
Quaternion(real=0.80, imag=<0.55, 0.25, 0.00>)
Quaternion(real=0.81, imag=<0.48, 0.33, 0.00>)
Quaternion(real=0.82, imag=<0.41, 0.41, 0.00>)
Quaternion(real=0.81, imag=<0.33, 0.48, 0.00>)
Quaternion(real=0.80, imag=<0.25, 0.55, 0.00>)
Quaternion(real=0.78, imag=<0.17, 0.61, 0.00>)
Quaternion(real=0.75, imag=<0.09, 0.66, 0.00>)
Quaternion(real=0.71, imag=<0.00, 0.71, 0.00>)
```

Operators

Quaternions may be multiplied to compound rotations. For example, to rotate 90 degrees around the X axis and then 90 degrees around the Y axis:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q2 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> q1 * q2
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
```

Multiplying a quaternion by a vector gives a vector, transformed appropriately:

```
>>> q = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> q * Vector3(1.0, 0, 0)
Vector3(0.00, 0.00, -1.00)
```

Similarly, any 3D object can be multiplied (e.g., **Point3**, **Line3**, **Sphere**, etc):

```
>>> q * Ray3(Point3(1., 1., 1.), Vector3(1., 1., 1.))
Ray3(<1.00, 1.00, -1.00> + u<1.00, 1.00, -1.00>)
```

As with the matrix classes, the constructors are also available as in-place operators. These are named `identity`, `rotate_euler` and `rotate_axis`. For example:

```
>>> q1 = Quaternion()
>>> q1.rotate_euler(math.pi / 2, math.pi / 2, 0)
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
>>> q1
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
```

Quaternions are usually unit length, but you may wish to use sized quaternions. In this case, you can find the magnitude using `abs`, `magnitude` and `magnitude_squared`, as with the vector classes. Example:

```
>>> q1 = Quaternion()
>>> abs(q1)
1.0
>>> q1.magnitude()
1.0
```

Similarly, the class implements `normalize` and `normalized` in the same way as the vectors.

The following methods do not alter the quaternion:

conjugated() Returns a quaternion that is the conjugate of the instance. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.conjugated()
Quaternion(real=0.71, imag=<-0.71, -0.00, -0.00>)
>>> q1
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
```

get_angle_axis() Returns a tuple (angle, axis), giving the angle to rotate around an axis equivalent to the quaternion. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.get_angle_axis()
(1.5707963267948966, Vector3(1.00, 0.00, 0.00))
```

get_matrix() Returns a **Matrix4** implementing the transformation of the quaternion. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.get_matrix()
Matrix4([
    1.00    0.00    0.00    0.00
    0.00    0.00   -1.00    0.00
    0.00    1.00    0.00    0.00
    0.00    0.00    0.00    1.00])
```

__init__(*w=1, x=0, y=0, z=0*)

__repr__()

__mul__(*other*)

__imul__(*other*)

mag2()

__abs__()

mag()

identity()

rotate_axis(*angle, axis*)

rotate_euler(*heading, attitude, bank*)

rotate_matrix(*m*)

conjugated()

normalize()

normalized()

```

get_angle_axis()
get_euler()
get_matrix()
classmethod new_identity()
classmethod new_rotate_axis (angle, axis)
classmethod new_rotate_euler (heading, attitude, bank)
classmethod new_rotate_matrix (m)
classmethod new_interpolate (q1, q2, t)

```

2.9 Goulib.graph module

efficient Euclidian Graphs for `networkx` and related algorithms

requires

- `networkx`
- `matplotlib`

optional

- `scipy` for delauney triangulation
- `rtree` for faster GeoGraph algorithms

class `Goulib.graph.index`

Bases: `object`

class `Property`

Bases: `object`

set_dimension (*n*)

class `Index` (*properties*)

Bases: `dict`

fallback for `rtree.index`

__init__ (*properties*)

count (*ignored*)

insert (*k, p, _*)

delete (*k, _*)

nearest (*p, num_results, objects='raw'*)

very inefficient, but remember it's a fallback...

class `Goulib.graph.AGraph`

Bases: `object`

`Goulib.graph.to_networkx_graph` (*data, create_using=None, multigraph_input=False*)

Make a NetworkX graph from a known data structure. enhances `networkx.convert.to_networkx_graph` :param data: any type handled by `convert.to_networkx_graph`, plus: * `scipy.spatial.qhull.Delaunay` to enable building a graph from a delauney triangulation

If `create_using` is a :class:`GeoGraph` and `data` is a Graph where nodes have a 'pos' attribute, then this attribute will be used to rename nodes as (x,y,...) tuples suitable for GeoGraph.

class `Goulib.graph.GeoGraph` (`data=None, nodes=None, **kwargs`)

Bases: `Goulib.graph._Geo, networkx.classes.multigraph.MultiGraph`

Undirected graph with nodes positions can be set to non multiedges anytime with attribute `multi=False`

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`__init__` (`data=None, nodes=None, **kwargs`)

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

class `Goulib.graph.DiGraph` (`data=None, nodes=None, **kwargs`)

Bases: `Goulib.graph._Geo, networkx.classes.multidigraph.MultiDiGraph`

directed graph with nodes positions can be set to non multiedges anytime with attribute `multi=False`

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`__init__` (`data=None, nodes=None, **kwargs`)

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`Goulib.graph.figure` (`g, box=None, **kwargs`)

Parameters

- **g** – _Geo derived Graph
- **box** – optional interval.Box if g has no box

Returns matplotlib axis suitable for drawing graph g

`Goulib.graph.draw_networkx` (`g, pos=None, **kwargs`)

improves `nx.draw_networkx` :param g: NetworkX Graph :param pos: can be either :

- optional dictionary of (x,y) node positions
- function of the form `lambda node:(x,y)` that maps node positions.
- None. in this case, nodes are directly used as positions if graph is a GeoGraph, otherwise `nx.draw_shell` is used

Parameters ****kwargs** – passed to `nx.draw` method as described in http://networkx.lanl.gov/reference/generated/networkx.drawing.nx_pylib.draw_networkx.html with one tweak:

- if `edge_color` is a function of the form `lambda data:color` string, it is mapped over all edges

`Goulib.graph.to_drawing(g, d=None, edges=[])`
 draws *Graph* to a *Drawing* :param g: *Graph* :param d: existing *Drawing* to draw onto, or None to create a new *Drawing* :param edges: iterable of edges (with data) that will be added, in the same order. By default all edges are drawn :return: *Drawing*

Graph edges with an 'entity' property

`Goulib.graph.write_dxf(g, filename)`
 writes *networkx.Graph* in .dxf format

`Goulib.graph.write_dot(g, filename)`

`Goulib.graph.to_json(g, **kwargs)`
Returns string JSON representation of a graph

`Goulib.graph.write_json(g, filename, **kwargs)`
 write a JSON file, suitable for D*.js representation

`Goulib.graph.read_json(filename, directed=False, multigraph=True, attrs=None)`

`Goulib.graph.delauney_triangulation(nodes, qhull_options='', incremental=False, **kwargs)`
https://en.wikipedia.org/wiki/Delaunay_triangulation :param nodes: _Geo graph or list of (x,y) or (x,y,z) node positions :param qhull_options: string passed to `scipy.spatial.Delaunay()`, which passes it to Qhull (<http://www.qhull.org/>) *'Qt' ensures all points are connected *'Qz' required when nodes lie on a sphere *'QJ' solves some singularity situations

Parameters *kwargs* – passed to the *GeoGraph* constructor

Returns *GeoGraph* with delauney triangulation between nodes

`Goulib.graph.euclidean_minimum_spanning_tree(nodes, **kwargs)`
Parameters *nodes* – list of (x,y) nodes positions

Returns *GeoGraph* with minimum spanning tree between nodes

see https://en.wikipedia.org/wiki/Euclidean_minimum_spanning_tree

`Goulib.graph.points_on_sphere(N)`

2.9.1 Classes

efficient Euclidian Graphs for *networkx* and related algorithms

requires

- *networkx*
- *matplotlib*

optional

- *scipy* for delauney triangulation
- *rtree* for faster *GeoGraph* algorithms

class `Goulib.graph.index`

Bases: *object*

class `Property`

Bases: *object*

set_dimension(*n*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__init__
Initialize self. See `help(type(self))` for accurate signature.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return `repr(self)`.

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

class Index (*properties*)
Bases: `dict`
fallback for `rtree.index`

```

__init__(properties)
count(ignored)
insert(k, p, _)
delete(k, _)
nearest(p, num_results, objects='raw')
    very inefficient, but remember it's a fallback...
__class__
    alias of type
__contains__()
    True if D has a key k, else False.
__delattr__
    Implement delattr(self, name).
__delitem__
    Delete self[key].
__dir__() → list
    default dir() implementation
__eq__
    Return self==value.
__format__()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__getitem__()
    x.__getitem__(y) <==> x[y]
__gt__
    Return self>value.
__hash__ = None
__iter__
    Implement iter(self).
__le__
    Return self<=value.
__len__
    Return len(self).
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__()
    Create and return a new object. See help(type) for accurate signature.

```

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

`__sizeof__()` → size of D in memory, in bytes

`__str__`
Return str(self).

`clear()` → None. Remove all items from D.

`copy()` → a shallow copy of D

`fromkeys()`
Returns a new dict with keys from iterable and values equal to value.

`get(k[, d])` → D[k] if k in D, else d. d defaults to None.

`items()` → a set-like object providing a view on D's items

`keys()` → a set-like object providing a view on D's keys

`pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

`popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

`setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

`update([E], **F)` → None. Update D from dict/iterable E and F.
If E is present and has a `.keys()` method, then does: for k in E: D[k] = E[k] If E is present and lacks a `.keys()` method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

`values()` → an object providing a view on D's values

`__class__`
alias of `type`

`__delattr__`
Implement delattr(self, name).

`__dir__()` → list
default dir() implementation

`__eq__`
Return self==value.

`__format__()`
default object formatter

`__ge__`
Return self>=value.

```

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__init__
    Initialize self. See help(type(self)) for accurate signature.

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

class Goulib.graph.AGraph

Bases: `object`

```

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

```

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

Goulib.graph.**to_networkx_graph** (data, create_using=None, multigraph_input=False)

Make a NetworkX graph from a known data structure. enhances `networkx.convert.to_networkx_graph` :param data: any type handled by `convert.to_networkx_graph`, plus: * `scipy.spatial.qhull.Delaunay` to enable building a graph from a delauney triangulation

If create_using is a :class:‘GeoGraph’ and data is a Graph where nodes have a ‘pos’ attribute, then this attribute will be used to rename nodes as (x,y,...) tuples suitable for GeoGraph.

class Goulib.graph.**GeoGraph** (data=None, nodes=None, **kwargs)

Bases: Goulib.graph._Geo, networkx.classes.multigraph.MultiGraph

Undirected graph with nodes positions can be set to non multiedges anytime with attribute multi=False

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__init__ (data=None, nodes=None, **kwargs)

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`__bool__()`

Returns True if graph has at least one node

`__class__`

alias of `type`

`__contains__(n)`

Return True if `n` is a node, False otherwise. Use: ‘`n in G`’.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> 1 in G
True
```

`__delattr__`

Implement `delattr(self, name)`.

`__dir__()` → list

default `dir()` implementation

`__eq__(other)`

Returns True if `self` and `other` are equal

`__format__()`

default object formatter

`__ge__`

Return `self >= value`.

`__getattr__`

Return `getattr(self, name)`.

`__getitem__(n)`

Return a dict of neighbors of node `n`. Use: ‘`G[n]`’.

n [node] A node in the graph.

adj_dict [dictionary] The adjacency dictionary for nodes connected to `n`.

`G[n]` is the same as `G.adj[n]` and similar to `G.neighbors(n)` (which is an iterator over `G.adj[n]`)

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G[0]
AtlasView({1: {}})
```

`__getstate__()`

`__gt__`

Return `self > value`.

`__hash__ = None`

`__iter__()`

Iterate over the nodes. Use: ‘for `n in G`’.

niter [iterator] An iterator over all nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> [n for n in G]
[0, 1, 2, 3]
>>> list(G)
[0, 1, 2, 3]
```

__le__

Return self<=value.

__len__()

Return the number of nodes. Use: 'len(G)'.

nnodes [int] The number of nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> len(G)
4
```

__lt__

Return self<value.

__ne__

Return self!=value.

__new__()

Create and return a new object. See help(type) for accurate signature.

__nonzero__()

Returns True if graph has at least one node

__reduce__()

helper for pickle

__reduce_ex__()

helper for pickle

__repr__

Return repr(self).

__setattr__

Implement setattr(self, name, value).

__sizeof__() → int

size of object in memory, in bytes

__str__()

Returns string representation, used mainly for logging and debugging

add_cycle (nodes, **attr)

add_edge (u, v, key=None, **attr)

add an edge to graph

Returns edge key

add_edge2 (u, v, key=None, **attrs)

add an edge to graph :return: edge data from created or existing edge

add_edges_from (ebunch, **attr)

Add all the edges in ebunch.

ebunch [container of edges] Each edge given in the container will be added to the graph. The edges can be:

- 2-tuples (u, v) or
- 3-tuples (u, v, d) for an edge data dict d, or
- 3-tuples (u, v, k) for not iterable key k, or
- 4-tuples (u, v, k, d) for an edge with data and key k

attr [keyword arguments, optional] Edge data (or labels or objects) can be assigned using keyword arguments.

A list of edge keys assigned to the edges in *ebunch*.

`add_edge` : add a single edge `add_weighted_edges_from` : convenient way to add weighted edges

Adding the same edge twice has no effect but any edge data will be updated when each duplicate edge is added.

Edge attributes specified in an *ebunch* take precedence over attributes specified via keyword arguments.

Default keys are generated using the method `new_edge_key()`. This method can be overridden by subclassing the base class and providing a custom `new_edge_key()` method.

```
>>> G = nx.Graph()      # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edges_from([(0, 1), (1, 2)]) # using a list of edge tuples
>>> e = zip(range(0, 3), range(1, 4))
>>> G.add_edges_from(e) # Add the path graph 0-1-2-3
```

Associate data to edges

```
>>> G.add_edges_from([(1, 2), (2, 3)], weight=3)
>>> G.add_edges_from([(3, 4), (1, 4)], label='WN2898')
```

add_node (*p*, ****attr**)

add a node or return one already very close :return (x,y,...) node id

add_nodes_from (*nodes*, ****attr**)

add_path (*nodes*, ****attr**)

add_star (*nodes*, ****attr**)

add_weighted_edges_from (*ebunch*, *weight='weight'*, ****attr**)

Add all the weighted edges in *ebunch* with specified weights.

ebunch [container of edges] Each edge in the container is added to the graph. The edges must be given as 3-tuples (u, v, w) where w is a number.

weight [string, optional (default= 'weight')] The attribute name for the edge weights to be added.

attr [keyword arguments, optional (default= no attributes)] Edge attributes to add/update for all edges.

`add_edge` : add a single edge `add_edges_from` : add multiple edges

Adding the same edge twice for Graph/DiGraph simply updates the edge data. For Multi-Graph/MultiDiGraph, duplicate edges are stored.

```
>>> G = nx.Graph()      # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_weighted_edges_from([(0, 1, 3.0), (1, 2, 7.5)])
```

adj

Graph adjacency object holding the neighbors of each node.

This object is a read-only dict-like structure with node keys and neighbor-dict values. The neighbor-dict is keyed by neighbor to the edgekey-data-dict. So `G.adj[3][2][0]['color'] = 'blue'` sets the color of the edge (3, 2, 0) to “blue”.

Iterating over `G.adj` behaves like a dict. Useful idioms include `for nbr, nbrdict in G.adj[n].items():`.

The neighbor information is also provided by subscripting the graph. So `for nbr, foovalue in G[node].data('foo', default=1):` works.

For directed graphs, `G.adj` holds outgoing (successor) info.

adjacency()

Return an iterator over (node, adjacency dict) tuples for all nodes.

For directed graphs, only outgoing neighbors/adjacencies are included.

adj_iter [iterator] An iterator over (node, adjacency dictionary) for all nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> [(n, nbrdict) for n, nbrdict in G.adjacency()]
[(0, {1: {}}), (1, {0: {}, 2: {}}), (2, {1: {}, 3: {}}), (3, {2: {}})]
```

adjlist_inner_dict_factory

alias of `dict`

adjlist_outer_dict_factory

alias of `dict`

box()

Returns nodes bounding box as (xmin,ymin,...),(xmax,ymax,...)

box_size()

Returns (x,y) size

clear()**closest_edges(p, data=False)**

Returns container of edges close to p and distance

closest_nodes(p, n=1, skip=False)

nodes closest to a given position :param p: (x,y) position tuple :param skip: optional bool to skip n itself
:return: list of nodes, minimal distance

contiguity(pts)

Returns int number of points from pts already in graph

copy()

Returns copy of self graph

degree

A DegreeView for the Graph as `G.degree` or `G.degree()`.

The node degree is the number of edges adjacent to the node. The weighted node degree is the sum of the edge weights for edges incident to that node.

This object provides an iterator for (node, degree) as well as lookup for the degree for a single node.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

weight [string or None, optional (default=None)] The name of an edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

If a single node is requested `deg : int`

Degree of the node, if a single node is passed as argument.

OR if multiple nodes are requested `nd_iter : iterator`

The iterator returns two-tuples of (node, degree).

```
>>> G = nx.Graph()      # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.degree(0) # node 0 with degree 1
1
>>> list(G.degree([0, 1]))
[(0, 1), (1, 2)]
```

dist (*u*, *v*)

Returns float distance between nodes *u* and *v*

draw (***kwargs*)

draw graph with default params

edge_attr_dict_factory

alias of `dict`

edge_key_dict_factory

alias of `dict`

edge_subgraph (*edges*)

Returns the subgraph induced by the specified edges.

The induced subgraph contains each edge in *edges* and each node incident to any one of those edges.

edges [iterable] An iterable of edges in this graph.

G [Graph] An edge-induced subgraph of this graph with the same edge attributes.

The graph, edge, and node attributes in the returned subgraph view are references to the corresponding attributes in the original graph. The view is read-only.

To create a full graph version of the subgraph with its own copy of the edge or node attributes, use:

```
>>> G.edge_subgraph(edges).copy()
```

```
>>> G = nx.path_graph(5)
>>> H = G.edge_subgraph([(0, 1), (3, 4)])
>>> list(H.nodes)
[0, 1, 3, 4]
>>> list(H.edges)
[(0, 1), (3, 4)]
```

edges

Return an iterator over the edges.

`edges(self, nbunch=None, data=False, keys=False, default=None)`

The EdgeView provides set-like operations on the edge-tuples as well as edge attribute lookup. When called, it also provides an EdgeDataView object which allows control of access to edge attributes (but does not provide set-like operations). Hence, `G.edges[u, v]['color']` provides the value of the color attribute for edge (u, v) while `for (u, v, c) in G.edges(data='color', default='red')`: iterates through all the edges yielding the color attribute.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple $(u, v, \text{ddict}[\text{data}])$. If True, return edge attribute dict in 3-tuple (u, v, ddict) . If False, return 2-tuple (u, v) .

keys [bool, optional (default=False)] If True, return edge keys with each edge.

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

edges [MultiEdgeView] A view of edge attributes, usually it iterates over (u, v) (u, v, k) or (u, v, k, d) tuples of edges, but can also be used for attribute lookup as `edges[u, v, k]['foo']`.

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiGraph()      # or MultiDiGraph
>>> nx.add_path(G, [0, 1, 2])
>>> key = G.add_edge(2, 3, weight=5)
>>> [e for e in G.edges()]
[(0, 1), (1, 2), (2, 3)]
>>> G.edges.data() # default data is {} (empty dict)
MultiEdgeDataView([(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})])
>>> G.edges.data('weight', default=1)
MultiEdgeDataView([(0, 1, 1), (1, 2, 1), (2, 3, 5)])
>>> G.edges(keys=True) # default keys are integers
MultiEdgeView([(0, 1, 0), (1, 2, 0), (2, 3, 0)])
>>> G.edges.data(keys=True)
MultiEdgeDataView([(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})])
>>> G.edges.data('weight', default=1, keys=True)
MultiEdgeDataView([(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)])
>>> G.edges([0, 3])
MultiEdgeDataView([(0, 1), (3, 2)])
>>> G.edges(0)
MultiEdgeDataView([(0, 1)])
```

fresh_copy()

Return a fresh copy graph with the same data structure.

A fresh copy has no nodes, edges or graph attributes. It is the same data structure as the current graph. This method is typically used to create an empty version of the graph.

If you subclass the base class you should overwrite this method to return your class of graph.

get_edge_data $(u, v, \text{key}=\text{None}, \text{default}=\text{None})$

Return the attribute dictionary associated with edge (u, v) .

This is identical to `G[u][v][key]` except the default is returned instead of an exception is the edge doesn't exist.

u, v : nodes

default [any Python object (default=None)] Value to return if the edge (u, v) is not found.

key [hashable identifier, optional (default=None)] Return data only for the edge with specified key.

edge_dict [dictionary] The edge attribute dictionary.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> key = G.add_edge(0, 1, key='a', weight=7)
>>> G[0][1]['a'] # key='a'
{'weight': 7}
>>> G.edges[0, 1, 'a'] # key='a'
{'weight': 7}
```

Warning: we protect the graph data structure by making `G.edges[1, 2, key]` and `G[1][2][key]` read-only dict-like structures. You need to specify all edge info to assign to the edge data associated with that edge.

```
>>> G[0][1]['a']['weight'] = 10
>>> G.edges[0, 1, 'a']['weight'] = 10
>>> G[0][1]['a']['weight']
10
>>> G.edges[1, 0, 'a']['weight']
10
```

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.get_edge_data(0, 1)
{0: {}}
>>> e = (0, 1)
>>> G.get_edge_data(*e) # tuple form
{0: {}}
>>> G.get_edge_data('a', 'b', default=0) # edge not in graph, return 0
0
```

has_edge (u, v, key=None)

Return True if the graph has an edge between nodes u and v.

This is the same as `v in G[u]` or `key in G[u][v]` without `KeyError` exceptions.

u, v [nodes] Nodes can be, for example, strings or numbers.

key [hashable identifier, optional (default=None)] If specified return True only if the edge with key is found.

edge_ind [bool] True if edge is in the graph, False otherwise.

Can be called either using two nodes u, v, an edge tuple (u, v), or an edge tuple (u, v, key).

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.has_edge(0, 1) # using two nodes
True
>>> e = (0, 1)
>>> G.has_edge(*e) # e is a 2-tuple (u, v)
True
>>> G.add_edge(0, 1, key='a')
'a'
>>> G.has_edge(0, 1, key='a') # specify key
True
```

```
>>> e=(0, 1, 'a')
>>> G.has_edge(*e) # e is a 3-tuple (u, v, 'a')
True
```

The following syntax are equivalent:

```
>>> G.has_edge(0, 1)
True
>>> 1 in G[0] # though this gives :exc:`KeyError` if 0 not in G
True
```

has_node(n)

Return True if the graph contains the node n.

Identical to *n in G*

n : node

```
>>> G = nx.path_graph(3) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.has_node(0)
True
```

It is more readable and simpler to use

```
>>> 0 in G
True
```

html(**kwargs)

is_directed()

Return True if graph is directed, False otherwise.

is_multigraph()

used internally in constructor

length(edges=None)

Parameters **edges** – iterator over edges either as (u,v,data) or (u,v,key,data). If None, all edges are taken

Returns sum of ‘length’ attributes of edges

multi

name

String identifier of the graph.

This graph attribute appears in the attribute dict G.graph keyed by the string “name”. as well as an attribute (technically a property) *G.name*. This is entirely user controlled.

nbunch_iter(nbunch=None)

Return an iterator over nodes contained in nbunch that are also in the graph.

The nodes in nbunch are checked for membership in the graph and if not are silently ignored.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

niter [iterator] An iterator over nodes in nbunch that are also in the graph. If nbunch is None, iterate over all nodes in the graph.

NetworkXError If nbunch is not a node or or sequence of nodes. If a node in nbunch is not hashable.

Graph.__iter__

When nbunch is an iterator, the returned iterator yields values directly from nbunch, becoming exhausted when nbunch is exhausted.

To test whether nbunch is a single node, one can use “if nbunch in self:”, even after processing with this routine.

If nbunch is not a node or a (possibly empty) sequence/iterator or None, a `NetworkXError` is raised. Also, if any object in nbunch is not hashable, a `NetworkXError` is raised.

neighbors (*n*)

Return an iterator over all neighbors of node *n*.

This is identical to `iter(G[n])`

n [node] A node in the graph

neighbors [iterator] An iterator over all neighbors of node *n*

NetworkXError If the node *n* is not in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> [n for n in G.neighbors(0)]
[1]
```

It is usually more convenient (and faster) to access the adjacency dictionary as `G[n]`:

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a', 'b', weight=7)
>>> G['a']
AtlasView({'b': {'weight': 7}})
>>> G = nx.path_graph(4)
>>> [n for n in G[0]]
[1]
```

new_edge_key (*u*, *v*)

Return an unused key for edges between nodes *u* and *v*.

The nodes *u* and *v* do not need to be already in the graph.

In the standard `MultiGraph` class the new key is the number of existing edges between *u* and *v* (increased if necessary to ensure unused). The first edge will have key 0, then 1, etc. If an edge is removed further `new_edge_keys` may not be in this order.

u, *v* : nodes

key : int

node

A `NodeView` of the Graph as `G.nodes` or `G.nodes()`.

Can be used as `G.nodes` for data lookup and for set-like operations. Can also be used as `G.nodes(data='color', default=None)` to return a `NodeDataView` which reports specific node data but no set operations. It presents a dict-like interface as well with `G.nodes.items()` iterating over (*node*, *nodedata*) 2-tuples and `G.nodes[3]['foo']` providing the value of the *foo* attribute for node 3. In addition, a view `G.nodes.data('foo')` provides a dict-like interface to the *foo* attribute of each node. `G.nodes.data('foo', default=1)` provides a default for nodes that do not have attribute *foo*.

data [string or bool, optional (default=False)] The node attribute returned in 2-tuple (n, ddict[data]). If True, return entire node attribute dict as (n, ddict). If False, return just the nodes n.

default [value, optional (default=None)] Value used for nodes that dont have the requested attribute. Only relevant if data is not True or False.

NodeView Allows set-like operations over the nodes as well as node attribute dict lookup and calling to get a NodeDataView. A NodeDataView iterates over (*n*, *data*) and has no set operations. A NodeView iterates over *n* and includes set operations.

When called, if data is False, an iterator over nodes. Otherwise an iterator of 2-tuples (node, attribute value) where the attribute is specified in *data*. If data is True then the attribute becomes the entire data dictionary.

If your node data is not needed, it is simpler and equivalent to use the expression `for n in G`, or `list(G)`.

There are two simple ways of getting a list of all nodes in the graph:

```
>>> G = nx.path_graph(3)
>>> list(G.nodes)
[0, 1, 2]
>>> list(G)
[0, 1, 2]
```

To get the node data along with the nodes:

```
>>> G.add_node(1, time='5pm')
>>> G.nodes[0]['foo'] = 'bar'
>>> list(G.nodes(data=True))
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
>>> list(G.nodes.data())
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
```

```
>>> list(G.nodes(data='foo'))
[(0, 'bar'), (1, None), (2, None)]
>>> list(G.nodes.data('foo'))
[(0, 'bar'), (1, None), (2, None)]
```

```
>>> list(G.nodes(data='time'))
[(0, None), (1, '5pm'), (2, None)]
>>> list(G.nodes.data('time'))
[(0, None), (1, '5pm'), (2, None)]
```

```
>>> list(G.nodes(data='time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
>>> list(G.nodes.data('time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
```

If some of your nodes have an attribute and the rest are assumed to have a default attribute value you can create a dictionary from node/attribute pairs using the *default* keyword argument to guarantee the value is never None:

```
>>> G = nx.Graph()
>>> G.add_node(0)
>>> G.add_node(1, weight=2)
>>> G.add_node(2, weight=3)
```

```
>>> dict(G.nodes(data='weight', default=1))
{0: 1, 1: 2, 2: 3}
```

node_dict_factory

alias of `dict`

nodes

A NodeView of the Graph as `G.nodes` or `G.nodes()`.

Can be used as `G.nodes` for data lookup and for set-like operations. Can also be used as `G.nodes(data='color', default=None)` to return a NodeDataView which reports specific node data but no set operations. It presents a dict-like interface as well with `G.nodes.items()` iterating over *(node, nodedata)* 2-tuples and `G.nodes[3]['foo']` providing the value of the *foo* attribute for node 3. In addition, a view `G.nodes.data('foo')` provides a dict-like interface to the *foo* attribute of each node. `G.nodes.data('foo', default=1)` provides a default for nodes that do not have attribute *foo*.

data [string or bool, optional (default=False)] The node attribute returned in 2-tuple (n, ddict[data]). If True, return entire node attribute dict as (n, ddict). If False, return just the nodes n.

default [value, optional (default=None)] Value used for nodes that dont have the requested attribute. Only relevant if data is not True or False.

NodeView Allows set-like operations over the nodes as well as node attribute dict lookup and calling to get a NodeDataView. A NodeDataView iterates over *(n, data)* and has no set operations. A NodeView iterates over *n* and includes set operations.

When called, if data is False, an iterator over nodes. Otherwise an iterator of 2-tuples (node, attribute value) where the attribute is specified in *data*. If data is True then the attribute becomes the entire data dictionary.

If your node data is not needed, it is simpler and equivalent to use the expression `for n in G`, or `list(G)`.

There are two simple ways of getting a list of all nodes in the graph:

```
>>> G = nx.path_graph(3)
>>> list(G.nodes)
[0, 1, 2]
>>> list(G)
[0, 1, 2]
```

To get the node data along with the nodes:

```
>>> G.add_node(1, time='5pm')
>>> G.nodes[0]['foo'] = 'bar'
>>> list(G.nodes(data=True))
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
>>> list(G.nodes.data())
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
```

```
>>> list(G.nodes(data='foo'))
[(0, 'bar'), (1, None), (2, None)]
>>> list(G.nodes.data('foo'))
[(0, 'bar'), (1, None), (2, None)]
```

```
>>> list(G.nodes(data='time'))
[(0, None), (1, '5pm'), (2, None)]
```

```
>>> list(G.nodes.data('time'))
[(0, None), (1, '5pm'), (2, None)]
```

```
>>> list(G.nodes(data='time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
>>> list(G.nodes.data('time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
```

If some of your nodes have an attribute and the rest are assumed to have a default attribute value you can create a dictionary from node/attribute pairs using the *default* keyword argument to guarantee the value is never None:

```
>>> G = nx.Graph()
>>> G.add_node(0)
>>> G.add_node(1, weight=2)
>>> G.add_node(2, weight=3)
>>> dict(G.nodes(data='weight', default=1))
{0: 1, 1: 2, 2: 3}
```

nodes_with_selfloops()

number_of_edges (*u=None, v=None*)

Return the number of edges between two nodes.

u, v [nodes, optional (Gefault=all edges)] If *u* and *v* are specified, return the number of edges between *u* and *v*. Otherwise return the total number of all edges.

nedges [int] The number of edges in the graph. If nodes *u* and *v* are specified return the number of edges between those nodes. If the graph is directed, this only returns the number of edges from *u* to *v*.

size

For undirected multigraphs, this method counts the total number of edges in the graph:

```
>>> G = nx.MultiGraph()
>>> G.add_edges_from([(0, 1), (0, 1), (1, 2)])
[0, 1, 0]
>>> G.number_of_edges()
3
```

If you specify two nodes, this counts the total number of edges joining the two nodes:

```
>>> G.number_of_edges(0, 1)
2
```

For directed multigraphs, this method can count the total number of directed edges from *u* to *v*:

```
>>> G = nx.MultiDiGraph()
>>> G.add_edges_from([(0, 1), (0, 1), (1, 0)])
[0, 1, 0]
>>> G.number_of_edges(0, 1)
2
>>> G.number_of_edges(1, 0)
1
```

number_of_nodes (*doublecheck=False*)

number_of_selfloops()

order ()

Return the number of nodes in the graph.

nnodes [int] The number of nodes in the graph.

number_of_nodes, __len__ which are identical

plot (**kwargs)

renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

pos (nodes=None)

Parameters **nodes** – a single node, an iterator of all nodes if None

Returns the position of node(s)

remove_edge (u, v=None, key=None, clean=False)

Parameters

- **u** – Node or Edge (Nodes tuple)
- **v** – Node if u is a single Node
- **clean** – bool removes disconnected nodes. must be False for certain nx algos to work

Result return attributes of removed edge

remove edge from graph. NetworkX graphs do not remove unused nodes

remove_edges_from (ebunch)

Remove all edges specified in ebunch.

ebunch: list or container of edge tuples Each edge given in the list or container will be removed from the graph. The edges can be:

- 2-tuples (u, v) All edges between u and v are removed.
- 3-tuples (u, v, key) The edge identified by key is removed.
- 4-tuples (u, v, key, data) where data is ignored.

remove_edge : remove a single edge

Will fail silently if an edge in ebunch is not in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> ebunch=[(1, 2), (2, 3)]
>>> G.remove_edges_from(ebunch)
```

Removing multiple copies of edges

```
>>> G = nx.MultiGraph()
>>> keys = G.add_edges_from([(1, 2), (1, 2), (1, 2)])
>>> G.remove_edges_from([(1, 2), (1, 2)])
>>> list(G.edges())
[(1, 2)]
>>> G.remove_edges_from([(1, 2), (1, 2)]) # silently ignore extra copy
>>> list(G.edges) # now empty graph
[]
```

remove_node (n)

Parameters **n** – node tuple

remove node from graph and rtree

remove_nodes_from (*nodes*)

Remove multiple nodes.

nodes [iterable container] A container of nodes (list, dict, set, etc.). If a node in the container is not in the graph it is silently ignored.

remove_node

```
>>> G = nx.path_graph(3) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> e = list(G.nodes)
>>> e
[0, 1, 2]
>>> G.remove_nodes_from(e)
>>> list(G.nodes)
[]
```

render (*fnt='svg', **kwargs*)

render graph to bitmap stream :param fnt: string defining the format. 'svg' by default for INotepads
:return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

save graph in various formats

selfloop_edges (*data=False, keys=False, default=None*)

shortest_path (*source=None, target=None*)

size (*weight=None*)

Return the number of edges or total of all edge weights.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1.

size [numeric] The number of edges or (if weight keyword is provided) the total weight sum.

If weight is None, returns an int. Otherwise a float (or more general numeric if the weights are more general).

number_of_edges

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.size()
3
```

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a', 'b', weight=2)
>>> G.add_edge('b', 'c', weight=4)
>>> G.size()
2
>>> G.size(weight='weight')
6.0
```

stats ()

Returns dict of graph data to use in `__repr__` or usable otherwise

subgraph (*nodes*)

Return a SubGraph view of the subgraph induced on nodes in *nodes*.

The induced subgraph of the graph contains the nodes in *nodes* and the edges between those nodes.

nodes [list, iterable] A container of nodes which will be iterated through once.

G [SubGraph View] A subgraph view of the graph. The graph structure cannot be changed but node/edge attributes can and are shared with the original graph.

The graph, edge and node attributes are shared with the original graph. Changes to the graph structure is ruled out by the view, but changes to attributes are reflected in the original graph.

To create a subgraph with its own copy of the edge/node attributes use: `G.subgraph(nodes).copy()`

For an inplace reduction of a graph to a subgraph you can remove nodes: `G.remove_nodes_from([n for n in G if n not in set(nodes)])`

```
>>> G = nx.Graph()    # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> nx.add_path(G, [0, 1, 2, 3])
>>> H = G.subgraph([0, 1, 2])
>>> list(H.edges)
[(0, 1), (1, 2)]
```

svg (***kwargs*)

to_directed (*as_view=False*)

Return a directed representation of the graph.

G [MultiDiGraph] A directed graph with the same name, same nodes, and with each edge (u, v, data) replaced by two directed edges (u, v, data) and (v, u, data).

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `D=DiGraph(G)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <https://docs.python.org/2/library/copy.html>.

Warning: If you have subclassed MultiGraph to use dict-like objects in the data structure, those changes do not transfer to the MultiDiGraph created by this method.

```
>>> G = nx.Graph()    # or MultiGraph, etc
>>> G.add_edge(0, 1)
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1), (1, 0)]
```

If already directed, return a (deep) copy

```
>>> G = nx.DiGraph()  # or MultiDiGraph, etc
>>> G.add_edge(0, 1)
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1)]
```

to_undirected (*as_view=False*)

Return an undirected copy of the graph.

G [Graph/MultiGraph] A deepcopy of the graph.

`copy`, `add_edge`, `add_edges_from`

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar $G = nx.MultiGraph(D)$ which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <https://docs.python.org/2/library/copy.html>.

Warning: If you have subclassed MultiGraph to use dict-like objects in the data structure, those changes do not transfer to the MultiGraph created by this method.

```
>>> G = nx.path_graph(2)    # or MultiGraph, etc
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1), (1, 0)]
>>> G2 = H.to_undirected()
>>> list(G2.edges)
[(0, 1)]
```

tol

class Goulib.graph.DiGraph(*data=None, nodes=None, **kwargs*)

Bases: Goulib.graph._Geo, networkx.classes.multidigraph.MultiDiGraph

directed graph with nodes positions can be set to non multiedges anytime with attribute multi=False

Parameters

- **data** – see *to_networkx_graph()* for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__init__(*data=None, nodes=None, **kwargs*)

Parameters

- **data** – see *to_networkx_graph()* for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__bool__()

Returns True if graph has at least one node

__class__

alias of *type*

__contains__(*n*)

Return True if *n* is a node, False otherwise. Use: ‘*n* in *G*’.

```
>>> G = nx.path_graph(4)    # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> 1 in G
True
```

__delattr__

Implement *delattr*(*self, name*).

__dir__() → list

default *dir*() implementation

__eq__(*other*)

Returns True if *self* and *other* are equal

__format__()

default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__getitem__(n)
Return a dict of neighbors of node n. Use: 'G[n]'.
n [node] A node in the graph.

adj_dict [dictionary] The adjacency dictionary for nodes connected to n.

G[n] is the same as G.adj[n] and similar to G.neighbors(n) (which is an iterator over G.adj[n])

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G[0]
AtlasView({1: {}})
```

__getstate__()

__gt__
Return self>value.

__hash__ = None

__iter__()
Iterate over the nodes. Use: 'for n in G'.

niter [iterator] An iterator over all nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> [n for n in G]
[0, 1, 2, 3]
>>> list(G)
[0, 1, 2, 3]
```

__le__
Return self<=value.

__len__()
Return the number of nodes. Use: 'len(G)'.

nnodes [int] The number of nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> len(G)
4
```

__lt__
Return self<value.

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__nonzero__()
Returns True if graph has at least one node

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__() → int
size of object in memory, in bytes

__str__()

Returns string representation, used mainly for logging and debugging

add_cycle(nodes, **attr)

add_edge(u, v, key=None, **attr)
add an edge to graph

Returns edge key

add_edge2(u, v, key=None, **attrs)
add an edge to graph :return: edge data from created or existing edge

add_edges_from(ebunch, **attr)
Add all the edges in ebunch.

ebunch [container of edges] Each edge given in the container will be added to the graph. The edges can be:

- 2-tuples (u, v) or
- 3-tuples (u, v, d) for an edge data dict d, or
- 3-tuples (u, v, k) for not iterable key k, or
- 4-tuples (u, v, k, d) for an edge with data and key k

attr [keyword arguments, optional] Edge data (or labels or objects) can be assigned using keyword arguments.

A list of edge keys assigned to the edges in *ebunch*.

add_edge : add a single edge **add_weighted_edges_from** : convenient way to add weighted edges

Adding the same edge twice has no effect but any edge data will be updated when each duplicate edge is added.

Edge attributes specified in an ebunch take precedence over attributes specified via keyword arguments.

Default keys are generated using the method `new_edge_key()`. This method can be overridden by subclassing the base class and providing a custom `new_edge_key()` method.

```
>>> G = nx.Graph()      # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edges_from([(0, 1), (1, 2)]) # using a list of edge tuples
>>> e = zip(range(0, 3), range(1, 4))
>>> G.add_edges_from(e) # Add the path graph 0-1-2-3
```

Associate data to edges

```
>>> G.add_edges_from([(1, 2), (2, 3)], weight=3)
>>> G.add_edges_from([(3, 4), (1, 4)], label='WN2898')
```

add_node (*p*, ****attr**)

add a node or return one already very close :return (x,y,...) node id

add_nodes_from (*nodes*, ****attr**)

add_path (*nodes*, ****attr**)

add_star (*nodes*, ****attr**)

add_weighted_edges_from (*ebunch*, *weight='weight'*, ****attr**)

Add all the weighted edges in ebunch with specified weights.

ebunch [container of edges] Each edge in the container is added to the graph. The edges must be given as 3-tuples (u, v, w) where w is a number.

weight [string, optional (default= 'weight')] The attribute name for the edge weights to be added.

attr [keyword arguments, optional (default= no attributes)] Edge attributes to add/update for all edges.

add_edge : add a single edge add_edges_from : add multiple edges

Adding the same edge twice for Graph/DiGraph simply updates the edge data. For Multi-Graph/MultiDiGraph, duplicate edges are stored.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_weighted_edges_from([(0, 1, 3.0), (1, 2, 7.5)])
```

adj

Graph adjacency object holding the neighbors of each node.

This object is a read-only dict-like structure with node keys and neighbor-dict values. The neighbor-dict is keyed by neighbor to the edgekey-dict. So *G.adj[3][2][0]['color'] = 'blue'* sets the color of the edge (3, 2, 0) to “blue”.

Iterating over *G.adj* behaves like a dict. Useful idioms include *for nbr, datadict in G.adj[n].items():*.

The neighbor information is also provided by subscripting the graph. So *for nbr, foovalue in G[node].data('foo', default=1):* works.

For directed graphs, *G.adj* holds outgoing (successor) info.

adjacency()

Return an iterator over (node, adjacency dict) tuples for all nodes.

For directed graphs, only outgoing neighbors/adjacencies are included.

adj_iter [iterator] An iterator over (node, adjacency dictionary) for all nodes in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> [(n, nbrdict) for n, nbrdict in G.adjacency()]
[(0, {1: {}}), (1, {0: {}, 2: {}}), (2, {1: {}, 3: {}}), (3, {2: {}})]
```

adjlist_inner_dict_factory

alias of *dict*

adjlist_outer_dict_factory

alias of *dict*

box()

Returns nodes bounding box as (xmin,ymin,...),(xmax,ymax,...)

box_size()

Returns (x,y) size

clear()

closest_edges (*p*, *data=False*)

Returns container of edges close to p and distance

closest_nodes (*p*, *n=1*, *skip=False*)

nodes closest to a given position :param p: (x,y) position tuple :param skip: optional bool to skip n itself
:return: list of nodes, minimal distance

contiguity (*pts*)

Returns int number of points from pts already in graph

copy()

Returns copy of self graph

degree

A DegreeView for the Graph as G.degree or G.degree().

The node degree is the number of edges adjacent to the node. The weighted node degree is the sum of the edge weights for edges incident to that node.

This object provides an iterator for (node, degree) as well as lookup for the degree for a single node.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

weight [string or None, optional (default=None)] The name of an edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

If a single nodes is requested deg : int

Degree of the node

OR if multiple nodes are requested nd_iter : iterator

The iterator returns two-tuples of (node, degree).

out_degree, in_degree

```
>>> G = nx.MultiDiGraph()
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.degree(0) # node 0 with degree 1
1
>>> list(G.degree([0, 1, 2]))
[(0, 1), (1, 2), (2, 2)]
```

dist (*u*, *v*)

Returns float distance between nodes u and v

draw (***kwargs*)

draw graph with default params

edge_attr_dict_factory

alias of `dict`

edge_key_dict_factory

alias of `dict`

edge_subgraph (*edges*)

Returns the subgraph induced by the specified edges.

The induced subgraph contains each edge in *edges* and each node incident to any one of those edges.

edges [iterable] An iterable of edges in this graph.

G [Graph] An edge-induced subgraph of this graph with the same edge attributes.

The graph, edge, and node attributes in the returned subgraph view are references to the corresponding attributes in the original graph. The view is read-only.

To create a full graph version of the subgraph with its own copy of the edge or node attributes, use:

```
>>> G.edge_subgraph(edges).copy()
```

```
>>> G = nx.path_graph(5)
>>> H = G.edge_subgraph([(0, 1), (3, 4)])
>>> list(H.nodes)
[0, 1, 3, 4]
>>> list(H.edges)
[(0, 1), (3, 4)]
```

edges

An OutMultiEdgeView of the Graph as `G.edges` or `G.edges()`.

`edges(self, nbunch=None, data=False, keys=False, default=None)`

The OutMultiEdgeView provides set-like operations on the edge-tuples as well as edge attribute lookup. When called, it also provides an EdgeDataView object which allows control of access to edge attributes (but does not provide set-like operations). Hence, `G.edges[u, v]['color']` provides the value of the color attribute for edge (u, v) while `for (u, v, c) in G.edges(data='color', default='red')`: iterates through all the edges yielding the color attribute with default 'red' if no color attribute exists.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple $(u, v, \text{ddict}[\text{data}])$. If True, return edge attribute dict in 3-tuple (u, v, ddict) . If False, return 2-tuple (u, v) .

keys [bool, optional (default=False)] If True, return edge keys with each edge.

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

edges [EdgeView] A view of edge attributes, usually it iterates over (u, v) (u, v, k) or (u, v, k, d) tuples of edges, but can also be used for attribute lookup as `edges[u, v, k]['foo']`.

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiDiGraph()
>>> nx.add_path(G, [0, 1, 2])
>>> key = G.add_edge(2, 3, weight=5)
>>> [e for e in G.edges()]
[(0, 1), (1, 2), (2, 3)]
>>> list(G.edges(data=True)) # default data is {} (empty dict)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
```

```

>>> list(G.edges(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> list(G.edges(keys=True)) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> list(G.edges(data=True, keys=True))
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight', default=1, keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> list(G.edges([0, 2]))
[(0, 1), (2, 3)]
>>> list(G.edges(0))
[(0, 1)]

```

in_edges, out_edges

fresh_copy()

Return a fresh copy graph with the same data structure.

A fresh copy has no nodes, edges or graph attributes. It is the same data structure as the current graph. This method is typically used to create an empty version of the graph.

If you subclass the base class you should overwrite this method to return your class of graph.

get_edge_data(u, v, key=None, default=None)

Return the attribute dictionary associated with edge (u, v).

This is identical to `G[u][v][key]` except the default is returned instead of an exception if the edge doesn't exist.

u, v : nodes

default [any Python object (default=None)] Value to return if the edge (u, v) is not found.

key [hashable identifier, optional (default=None)] Return data only for the edge with specified key.

edge_dict [dictionary] The edge attribute dictionary.

```

>>> G = nx.MultiGraph() # or MultiDiGraph
>>> key = G.add_edge(0, 1, key='a', weight=7)
>>> G[0][1]['a'] # key='a'
{'weight': 7}
>>> G.edges[0, 1, 'a'] # key='a'
{'weight': 7}

```

Warning: we protect the graph data structure by making `G.edges[1, 2, key]` and `G[1][2][key]` read-only dict-like structures. You need to specify all edge info to assign to the edge data associated with that edge.

```

>>> G[0][1]['a']['weight'] = 10
>>> G.edges[0, 1, 'a']['weight'] = 10
>>> G[0][1]['a']['weight']
10
>>> G.edges[1, 0, 'a']['weight']
10

```

```

>>> G = nx.MultiGraph() # or MultiDiGraph
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.get_edge_data(0, 1)
{0: {}}
>>> e = (0, 1)

```

```
>>> G.get_edge_data(*e) # tuple form
{0: {}}
>>> G.get_edge_data('a', 'b', default=0) # edge not in graph, return 0
0
```

has_edge (*u*, *v*, *key=None*)

Return True if the graph has an edge between nodes *u* and *v*.

This is the same as *v* in *G[u]* or *key* in *G[u][v]* without *KeyError* exceptions.

u, v [nodes] Nodes can be, for example, strings or numbers.

key [hashable identifier, optional (default=None)] If specified return True only if the edge with *key* is found.

edge_ind [bool] True if edge is in the graph, False otherwise.

Can be called either using two nodes *u*, *v*, an edge tuple (*u*, *v*), or an edge tuple (*u*, *v*, *key*).

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.has_edge(0, 1) # using two nodes
True
>>> e = (0, 1)
>>> G.has_edge(*e) # e is a 2-tuple (u, v)
True
>>> G.add_edge(0, 1, key='a')
'a'
>>> G.has_edge(0, 1, key='a') # specify key
True
>>> e=(0, 1, 'a')
>>> G.has_edge(*e) # e is a 3-tuple (u, v, 'a')
True
```

The following syntax are equivalent:

```
>>> G.has_edge(0, 1)
True
>>> 1 in G[0] # though this gives :exc:`KeyError` if 0 not in G
True
```

has_node (*n*)

Return True if the graph contains the node *n*.

Identical to *n* in *G*

n : node

```
>>> G = nx.path_graph(3) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.has_node(0)
True
```

It is more readable and simpler to use

```
>>> 0 in G
True
```

has_predecessor (*u*, *v*)

Return True if node *u* has predecessor *v*.

This is true if graph has the edge $u \leftarrow v$.

has_successor (u, v)

Return True if node u has successor v .

This is true if graph has the edge $u \rightarrow v$.

html (***kwargs*)

in_degree

A DegreeView for (node, in_degree) or in_degree for single node.

The node in-degree is the number of edges pointing in to the node. The weighted node degree is the sum of the edge weights for edges incident to that node.

This object provides an iterator for (node, degree) as well as lookup for the degree for a single node.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

If a single node is requested $\text{deg} : \text{int}$

Degree of the node

OR if multiple nodes are requested $\text{nd_iter} : \text{iterator}$

The iterator returns two-tuples of (node, in-degree).

degree, out_degree

```
>>> G = nx.MultiDiGraph()
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.in_degree(0) # node 0 with degree 0
0
>>> list(G.in_degree([0, 1, 2]))
[(0, 0), (1, 1), (2, 1)]
```

in_edges

An InMultiEdgeView of the Graph as $G.in_edges$ or $G.in_edges()$.

$\text{in_edges}(\text{self}, \text{nbunch}=\text{None}, \text{data}=\text{False}, \text{keys}=\text{False}, \text{default}=\text{None})$

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple ($u, v, \text{ddict}[\text{data}]$). If True, return edge attribute dict in 3-tuple (u, v, ddict). If False, return 2-tuple (u, v).

keys [bool, optional (default=False)] If True, return edge keys with each edge.

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

in_edges [InMultiEdgeView] A view of edge attributes, usually it iterates over (u, v) or (u, v, k) or (u, v, k, d) tuples of edges, but can also be used for attribute lookup as $\text{edges}[u, v, k][\text{'foo'}]$.

edges

is_directed ()

Return True if graph is directed, False otherwise.

is_multigraph()

used internally in constructor

length (*edges=None*)

Parameters **edges** – iterator over edges either as (u,v,data) or (u,v,key,data). If None, all edges are taken

Returns sum of ‘length’ attributes of edges

multi

name

String identifier of the graph.

This graph attribute appears in the attribute dict `G.graph` keyed by the string “*name*”. as well as an attribute (technically a property) *G.name*. This is entirely user controlled.

nbunch_iter (*nbunch=None*)

Return an iterator over nodes contained in nbunch that are also in the graph.

The nodes in nbunch are checked for membership in the graph and if not are silently ignored.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

niter [iterator] An iterator over nodes in nbunch that are also in the graph. If nbunch is None, iterate over all nodes in the graph.

NetworkXError If nbunch is not a node or or sequence of nodes. If a node in nbunch is not hashable.

`Graph.__iter__`

When nbunch is an iterator, the returned iterator yields values directly from nbunch, becoming exhausted when nbunch is exhausted.

To test whether nbunch is a single node, one can use “if nbunch in self:”, even after processing with this routine.

If nbunch is not a node or a (possibly empty) sequence/iterator or None, a `NetworkXError` is raised. Also, if any object in nbunch is not hashable, a `NetworkXError` is raised.

neighbors (*n*)

Return an iterator over successor nodes of n.

`neighbors()` and `successors()` are the same.

new_edge_key (*u, v*)

Return an unused key for edges between nodes *u* and *v*.

The nodes *u* and *v* do not need to be already in the graph.

In the standard `MultiGraph` class the new key is the number of existing edges between *u* and *v* (increased if necessary to ensure unused). The first edge will have key 0, then 1, etc. If an edge is removed further `new_edge_keys` may not be in this order.

u, v : nodes

key : int

node

A `NodeView` of the Graph as `G.nodes` or `G.nodes()`.

Can be used as `G.nodes` for data lookup and for set-like operations. Can also be used as `G.nodes(data='color', default=None)` to return a `NodeDataView` which reports specific node data but no set operations. It presents a dict-like interface as well with `G.nodes.items()` iterating over `(node, nodedata)` 2-tuples and `G.nodes[3]['foo']` providing the value of the `foo` attribute for node 3. In addition, a view `G.nodes.data('foo')` provides a dict-like interface to the `foo` attribute of each node. `G.nodes.data('foo', default=1)` provides a default for nodes that do not have attribute `foo`.

data [string or bool, optional (default=False)] The node attribute returned in 2-tuple `(n, ddict[data])`. If True, return entire node attribute dict as `(n, ddict)`. If False, return just the nodes `n`.

default [value, optional (default=None)] Value used for nodes that dont have the requested attribute. Only relevant if data is not True or False.

NodeView Allows set-like operations over the nodes as well as node attribute dict lookup and calling to get a `NodeDataView`. A `NodeDataView` iterates over `(n, data)` and has no set operations. A `NodeView` iterates over `n` and includes set operations.

When called, if data is False, an iterator over nodes. Otherwise an iterator of 2-tuples (node, attribute value) where the attribute is specified in `data`. If data is True then the attribute becomes the entire data dictionary.

If your node data is not needed, it is simpler and equivalent to use the expression `for n in G`, or `list(G)`.

There are two simple ways of getting a list of all nodes in the graph:

```
>>> G = nx.path_graph(3)
>>> list(G.nodes)
[0, 1, 2]
>>> list(G)
[0, 1, 2]
```

To get the node data along with the nodes:

```
>>> G.add_node(1, time='5pm')
>>> G.nodes[0]['foo'] = 'bar'
>>> list(G.nodes(data=True))
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
>>> list(G.nodes.data())
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
```

```
>>> list(G.nodes(data='foo'))
[(0, 'bar'), (1, None), (2, None)]
>>> list(G.nodes.data('foo'))
[(0, 'bar'), (1, None), (2, None)]
```

```
>>> list(G.nodes(data='time'))
[(0, None), (1, '5pm'), (2, None)]
>>> list(G.nodes.data('time'))
[(0, None), (1, '5pm'), (2, None)]
```

```
>>> list(G.nodes(data='time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
>>> list(G.nodes.data('time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
```

If some of your nodes have an attribute and the rest are assumed to have a default attribute value you can create a dictionary from node/attribute pairs using the *default* keyword argument to guarantee the value is never None:

```
>>> G = nx.Graph()
>>> G.add_node(0)
>>> G.add_node(1, weight=2)
>>> G.add_node(2, weight=3)
>>> dict(G.nodes(data='weight', default=1))
{0: 1, 1: 2, 2: 3}
```

node_dict_factory

alias of `dict`

nodes

A NodeView of the Graph as `G.nodes` or `G.nodes()`.

Can be used as `G.nodes` for data lookup and for set-like operations. Can also be used as `G.nodes(data='color', default=None)` to return a NodeDataView which reports specific node data but no set operations. It presents a dict-like interface as well with `G.nodes.items()` iterating over *(node, nodedata)* 2-tuples and `G.nodes[3]['foo']` providing the value of the *foo* attribute for node 3. In addition, a view `G.nodes.data('foo')` provides a dict-like interface to the *foo* attribute of each node. `G.nodes.data('foo', default=1)` provides a default for nodes that do not have attribute *foo*.

data [string or bool, optional (default=False)] The node attribute returned in 2-tuple *(n, ddict[data])*. If True, return entire node attribute dict as *(n, ddict)*. If False, return just the nodes *n*.

default [value, optional (default=None)] Value used for nodes that dont have the requested attribute. Only relevant if *data* is not True or False.

NodeView Allows set-like operations over the nodes as well as node attribute dict lookup and calling to get a NodeDataView. A NodeDataView iterates over *(n, data)* and has no set operations. A NodeView iterates over *n* and includes set operations.

When called, if *data* is False, an iterator over nodes. Otherwise an iterator of 2-tuples (node, attribute value) where the attribute is specified in *data*. If *data* is True then the attribute becomes the entire data dictionary.

If your node data is not needed, it is simpler and equivalent to use the expression `for n in G`, or `list(G)`.

There are two simple ways of getting a list of all nodes in the graph:

```
>>> G = nx.path_graph(3)
>>> list(G.nodes)
[0, 1, 2]
>>> list(G)
[0, 1, 2]
```

To get the node data along with the nodes:

```
>>> G.add_node(1, time='5pm')
>>> G.nodes[0]['foo'] = 'bar'
>>> list(G.nodes(data=True))
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
>>> list(G.nodes.data())
[(0, {'foo': 'bar'}), (1, {'time': '5pm'}), (2, {})]
```

```
>>> list(G.nodes(data='foo'))
[(0, 'bar'), (1, None), (2, None)]
>>> list(G.nodes.data('foo'))
[(0, 'bar'), (1, None), (2, None)]
```

```
>>> list(G.nodes(data='time'))
[(0, None), (1, '5pm'), (2, None)]
>>> list(G.nodes.data('time'))
[(0, None), (1, '5pm'), (2, None)]
```

```
>>> list(G.nodes(data='time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
>>> list(G.nodes.data('time', default='Not Available'))
[(0, 'Not Available'), (1, '5pm'), (2, 'Not Available')]
```

If some of your nodes have an attribute and the rest are assumed to have a default attribute value you can create a dictionary from node/attribute pairs using the *default* keyword argument to guarantee the value is never None:

```
>>> G = nx.Graph()
>>> G.add_node(0)
>>> G.add_node(1, weight=2)
>>> G.add_node(2, weight=3)
>>> dict(G.nodes(data='weight', default=1))
{0: 1, 1: 2, 2: 3}
```

nodes_with_selfloops()

number_of_edges (*u=None, v=None*)

Return the number of edges between two nodes.

u, v [nodes, optional (Gefault=all edges)] If *u* and *v* are specified, return the number of edges between *u* and *v*. Otherwise return the total number of all edges.

nedges [int] The number of edges in the graph. If nodes *u* and *v* are specified return the number of edges between those nodes. If the graph is directed, this only returns the number of edges from *u* to *v*.

size

For undirected multigraphs, this method counts the total number of edges in the graph:

```
>>> G = nx.MultiGraph()
>>> G.add_edges_from([(0, 1), (0, 1), (1, 2)])
[0, 1, 0]
>>> G.number_of_edges()
3
```

If you specify two nodes, this counts the total number of edges joining the two nodes:

```
>>> G.number_of_edges(0, 1)
2
```

For directed multigraphs, this method can count the total number of directed edges from *u* to *v*:

```
>>> G = nx.MultiDiGraph()
>>> G.add_edges_from([(0, 1), (0, 1), (1, 0)])
[0, 1, 0]
```

```
>>> G.number_of_edges(0, 1)
2
>>> G.number_of_edges(1, 0)
1
```

number_of_nodes (*doublecheck=False*)

number_of_selfloops ()

order ()

Return the number of nodes in the graph.

nnodes [int] The number of nodes in the graph.

number_of_nodes, __len__ which are identical

out_degree

Return an iterator for (node, out-degree) or out-degree for single node.

out_degree(self, nbunch=None, weight=None)

The node out-degree is the number of edges pointing out of the node. This function returns the out-degree for a single node or an iterator for a bunch of nodes or if nothing is passed as argument.

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights.

If a single node is requested deg : int

Degree of the node

OR if multiple nodes are requested nd_iter : iterator

The iterator returns two-tuples of (node, out-degree).

degree, in_degree

```
>>> G = nx.MultiDiGraph()
>>> nx.add_path(G, [0, 1, 2, 3])
>>> G.out_degree(0) # node 0 with degree 1
1
>>> list(G.out_degree([0, 1, 2]))
[(0, 1), (1, 1), (2, 1)]
```

out_edges

An OutMultiEdgeView of the Graph as G.edges or G.edges().

edges(self, nbunch=None, data=False, keys=False, default=None)

The OutMultiEdgeView provides set-like operations on the edge-tuples as well as edge attribute lookup. When called, it also provides an EdgeDataView object which allows control of access to edge attributes (but does not provide set-like operations). Hence, *G.edges[u, v]['color']* provides the value of the color attribute for edge (u, v) while *for (u, v, c) in G.edges(data='color', default='red')*: iterates through all the edges yielding the color attribute with default 'red' if no color attribute exists.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [single node, container, or all nodes (default= all nodes)] The view will only report edges incident to these nodes.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple (u, v, ddict[data]). If True, return edge attribute dict in 3-tuple (u, v, ddict). If False, return 2-tuple (u, v).

keys [bool, optional (default=False)] If True, return edge keys with each edge.

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

edges [EdgeView] A view of edge attributes, usually it iterates over (u, v) (u, v, k) or (u, v, k, d) tuples of edges, but can also be used for attribute lookup as `edges[u, v, k]['foo']`.

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiDiGraph()
>>> nx.add_path(G, [0, 1, 2])
>>> key = G.add_edge(2, 3, weight=5)
>>> [e for e in G.edges()]
[(0, 1), (1, 2), (2, 3)]
>>> list(G.edges(data=True)) # default data is {} (empty dict)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> list(G.edges(keys=True)) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> list(G.edges(data=True, keys=True))
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight', default=1, keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> list(G.edges([0, 2]))
[(0, 1), (2, 3)]
>>> list(G.edges(0))
[(0, 1)]
```

`in_edges`, `out_edges`

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

pos (*nodes=None*)

Parameters *nodes* – a single node, an iterator of all nodes if None

Returns the position of node(s)

pred

Graph adjacency object holding the predecessors of each node.

This object is a read-only dict-like structure with node keys and neighbor-dict values. The neighbor-dict is keyed by neighbor to the edgekey-dict. So `G.adj[3][2][0]['color'] = 'blue'` sets the color of the edge (3, 2, 0) to “blue”.

Iterating over `G.adj` behaves like a dict. Useful idioms include `for nbr, datadict in G.adj[n].items():`.

predecessors (*n*)

Return an iterator over predecessor nodes of *n*.

remove_edge (*u, v=None, key=None, clean=False*)

Parameters

- **u** – Node or Edge (Nodes tuple)
- **v** – Node if u is a single Node
- **clean** – bool removes disconnected nodes. must be False for certain nx algos to work

Result return attributes of removed edge

remove edge from graph. NetworkX graphs do not remove unused nodes

remove_edges_from (*ebunch*)

Remove all edges specified in ebunch.

ebunch: list or container of edge tuples Each edge given in the list or container will be removed from the graph. The edges can be:

- 2-tuples (u, v) All edges between u and v are removed.
- 3-tuples (u, v, key) The edge identified by key is removed.
- 4-tuples (u, v, key, data) where data is ignored.

remove_edge : remove a single edge

Will fail silently if an edge in ebunch is not in the graph.

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> ebunch=[(1, 2), (2, 3)]
>>> G.remove_edges_from(ebunch)
```

Removing multiple copies of edges

```
>>> G = nx.MultiGraph()
>>> keys = G.add_edges_from([(1, 2), (1, 2), (1, 2)])
>>> G.remove_edges_from([(1, 2), (1, 2)])
>>> list(G.edges())
[(1, 2)]
>>> G.remove_edges_from([(1, 2), (1, 2)]) # silently ignore extra copy
>>> list(G.edges) # now empty graph
[]
```

remove_node (*n*)

Parameters **n** – node tuple

remove node from graph and rtree

remove_nodes_from (*nodes*)

Remove multiple nodes.

nodes [iterable container] A container of nodes (list, dict, set, etc.). If a node in the container is not in the graph it is silently ignored.

remove_node

```
>>> G = nx.path_graph(3) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> e = list(G.nodes)
>>> e
[0, 1, 2]
>>> G.remove_nodes_from(e)
>>> list(G.nodes)
[]
```

render (*fmt='svg', **kwargs*)

render graph to bitmap stream :param *fmt*: string defining the format. 'svg' by default for INotepads
:return: matplotlib figure as a byte stream in specified format

reverse (*copy=True*)

Return the reverse of the graph.

The reverse is a graph with the same nodes and edges but with the directions of the edges reversed.

copy [bool optional (default=True)] If True, return a new DiGraph holding the reversed edges. If False, the reverse graph is created using a view of the original graph.

save (*filename, **kwargs*)

save graph in various formats

selfloop_edges (*data=False, keys=False, default=None*)

shortest_path (*source=None, target=None*)

size (*weight=None*)

Return the number of edges or total of all edge weights.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1.

size [numeric] The number of edges or (if weight keyword is provided) the total weight sum.

If weight is None, returns an int. Otherwise a float (or more general numeric if the weights are more general).

number_of_edges

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.size()
3
```

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a', 'b', weight=2)
>>> G.add_edge('b', 'c', weight=4)
>>> G.size()
2
>>> G.size(weight='weight')
6.0
```

stats ()

Returns dict of graph data to use in `__repr__` or usable otherwise

subgraph (*nodes*)

Return a SubGraph view of the subgraph induced on nodes in *nodes*.

The induced subgraph of the graph contains the nodes in *nodes* and the edges between those nodes.

nodes [list, iterable] A container of nodes which will be iterated through once.

G [SubGraph View] A subgraph view of the graph. The graph structure cannot be changed but node/edge attributes can and are shared with the original graph.

The graph, edge and node attributes are shared with the original graph. Changes to the graph structure is ruled out by the view, but changes to attributes are reflected in the original graph.

To create a subgraph with its own copy of the edge/node attributes use: `G.subgraph(nodes).copy()`

For an inplace reduction of a graph to a subgraph you can remove nodes: `G.remove_nodes_from([n for n in G if n not in set(nodes)])`

```
>>> G = nx.path_graph(4) # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> H = G.subgraph([0, 1, 2])
>>> list(H.edges)
[(0, 1), (1, 2)]
```

succ

Graph adjacency object holding the successors of each node.

This object is a read-only dict-like structure with node keys and neighbor-dict values. The neighbor-dict is keyed by neighbor to the edgekey-dict. So `G.adj[3][2][0]['color'] = 'blue'` sets the color of the edge (3, 2, 0) to “blue”.

Iterating over `G.adj` behaves like a dict. Useful idioms include *for nbr, datadict in G.adj[n].items()*:

The neighbor information is also provided by subscripting the graph. So *for nbr, foovalue in G[node].data('foo', default=1)*: works.

For directed graphs, `G.succ` is identical to `G.adj`.

successors (*n*)

Return an iterator over successor nodes of *n*.

`neighbors()` and `successors()` are the same.

svg (***kwargs*)

to_directed (*as_view=False*)

Return a directed representation of the graph.

G [`MultiDiGraph`] A directed graph with the same name, same nodes, and with each edge (u, v, data) replaced by two directed edges (u, v, data) and (v, u, data).

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `D=DiGraph(G)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <https://docs.python.org/2/library/copy.html>.

Warning: If you have subclassed `MultiGraph` to use dict-like objects in the data structure, those changes do not transfer to the `MultiDiGraph` created by this method.

```
>>> G = nx.Graph() # or MultiGraph, etc
>>> G.add_edge(0, 1)
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1), (1, 0)]
```

If already directed, return a (deep) copy

```
>>> G = nx.DiGraph() # or MultiDiGraph, etc
>>> G.add_edge(0, 1)
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1)]
```

to_undirected (*reciprocal=False, as_view=False*)

Return an undirected representation of the digraph.

reciprocal [bool (optional)] If True only keep edges that appear in both directions in the original digraph.

as_view [bool (optional, default=False)] If True return an undirected view of the original directed graph.

G [MultiGraph] An undirected graph with the same name and nodes and with edge (u, v, data) if either (u, v, data) or (v, u, data) is in the digraph. If both edges exist in digraph and their edge data is different, only one edge is created with an arbitrary choice of which edge data to use. You must check and correct for this manually if desired.

MultiGraph, copy, add_edge, add_edges_from

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `D=MultiGraph(G)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <https://docs.python.org/2/library/copy.html>.

Warning: If you have subclassed `MultiDiGraph` to use dict-like objects in the data structure, those changes do not transfer to the `MultiGraph` created by this method.

```
>>> G = nx.path_graph(2)      # or MultiGraph, etc
>>> H = G.to_directed()
>>> list(H.edges)
[(0, 1), (1, 0)]
>>> G2 = H.to_undirected()
>>> list(G2.edges)
[(0, 1)]
```

tol

`Goulib.graph.figure(g, box=None, **kwargs)`

Parameters

- **g** – `_Geo` derived Graph
- **box** – optional `interval.Box` if g has no box

Returns matplotlib axis suitable for drawing graph g

`Goulib.graph.draw_networkx(g, pos=None, **kwargs)`

improves `nx.draw_networkx` :param g: NetworkX Graph :param pos: can be either :

- optional dictionary of (x,y) node positions
- function of the form `lambda node:(x,y)` that maps node positions.
- None. in this case, nodes are directly used as positions if graph is a `GeoGraph`, otherwise `nx.draw_shell` is used

Parameters `**kwargs` – passed to `nx.draw` method as described in http://networkx.lanl.gov/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html with one tweak:

- if `edge_color` is a function of the form `lambda data:color` string, it is mapped over all edges

`Goulib.graph.to_drawing(g, d=None, edges=[])`

draws Graph to a *Drawing* :param g: Graph :param d: existing *Drawing* to draw onto, or None to create a new *Drawing* :param edges: iterable of edges (with data) that will be added, in the same order. By default all edges are drawn :return: *Drawing*

Graph edges with an ‘entity’ property

```
Goulib.graph.write_dxf(g, filename)
    writes networkx.Graph in .dxf format
```

```
Goulib.graph.write_dot(g, filename)
```

```
Goulib.graph.to_json(g, **kwargs)
```

Returns string JSON representation of a graph

```
Goulib.graph.write_json(g, filename, **kwargs)
    write a JSON file, suitable for D*.js representation
```

```
Goulib.graph.read_json(filename, directed=False, multigraph=True, attrs=None)
```

```
Goulib.graph.delauney_triangulation(nodes, qhull_options='', incremental=False, **kwargs)
    https://en.wikipedia.org/wiki/Delaunay\_triangulation :param nodes: _Geo graph or list of (x,y) or (x,y,z) node
    positions :param qhull_options: string passed to scipy.spatial.Delaunay(), which passes it to Qhull (
    http://www.qhull.org/) *‘Qt’ ensures all points are connected *‘Qz’ required when nodes lie on a sphere *‘QJ’
    solves some singularity situations
```

Parameters **kwargs** – passed to the `GeoGraph` constructor

Returns `GeoGraph` with delauney triangulation between nodes

```
Goulib.graph.euclidean_minimum_spanning_tree(nodes, **kwargs)
```

Parameters **nodes** – list of (x,y) nodes positions

Returns `GeoGraph` with minimum spanning tree between nodes

see https://en.wikipedia.org/wiki/Euclidean_minimum_spanning_tree

```
Goulib.graph.points_on_sphere(N)
```

2.10 Goulib.image module

```
class Goulib.image.Mode(name, nchannels, type, min, max)
```

Bases: `object`

```
__init__(name, nchannels, type, min, max)
```

```
__repr__()
```

```
__class__
    alias of type
```

```
__delattr__
    Implement delattr(self, name).
```

```
__dir__() → list
    default dir() implementation
```

```
__eq__
    Return self==value.
```

```
__format__()
    default object formatter
```

```
__ge__
    Return self>=value.
```

```
__getattr__  
    Return getattr(self, name).  
  
__gt__  
    Return self>value.  
  
__hash__  
    Return hash(self).  
  
__le__  
    Return self<=value.  
  
__lt__  
    Return self<value.  
  
__ne__  
    Return self!=value.  
  
__new__()  
    Create and return a new object. See help(type) for accurate signature.  
  
__reduce__()  
    helper for pickle  
  
__reduce_ex__()  
    helper for pickle  
  
__setattr__  
    Implement setattr(self, name, value).  
  
__sizeof__() → int  
    size of object in memory, in bytes  
  
__str__  
    Return str(self).
```

`Goulib.image.nchannels(arr)`

`Goulib.image.guessmode(arr)`

`Goulib.image.adapt_rgb(func)`

Decorator that adapts to RGB(A) images to a gray-scale filter. :param apply_to_rgb: function

Function that returns a filtered image from an image-filter and RGB image. This will only be called if the image is RGB-like.

class `Goulib.image.Image(data=None, mode=None, **kwargs)`

Bases: `Goulib.plot.Plot`

Parameters `data` – can be either:

- `PIL.Image` : makes a copy
- string : path of image to load
- memoryview (extracted from a db blob)
- None : creates an empty image with kwargs parameters:

**** size** : (y,x) pixel size tuple **** mode** : 'F' (gray) by default **** color**: to fill None=black by default **** colormap**: Palette or matplotlib colormap

`__init__(data=None, mode=None, **kwargs)`

Parameters `data` – can be either:

- *PIL.Image* : makes a copy
- string : path of image to load
- memoryview (extracted from a db blob)
- None : creates an empty image with kwargs parameters:

**** size** : (y,x) pixel size tuple **** mode** : 'F' (gray) by default **** color**: to fill None=black by default **** colormap**: Palette or matplotlib colormap

shape

size

nchannels

npixels

__nonzero__ ()

__lt__ (other)
is smaller

load (path)

save (path, autoconvert=True, format_str=None, **kwargs)

saves an image :param path: string with path/filename.ext :param autoconvert: bool, if True converts color planes formats to RGB :param format_str: str of file format. set to 'PNG' by skimage.io.imsave :param kwargs: optional params passed to skimage.io.imsave: :return: self for chaining

render (fmt='PNG', **kwargs)

static open (path)

PIL(low) compatibility

static new (mode, size, color='black')

PIL(low) compatibility

pil

convert to PIL(low) Image :see: <http://effbot.org/imagingbook/concepts.htm>

getdata (dtype=<class 'numpy.uint8'>, copy=True)

split (mode=None)

getpixel (yx)

putpixel (yx, value)

getpalette (maxcolors=256)

setpalette (p)

getcolors (maxcolors=256)

Returns an unsorted list of (count, color) tuples,

where count is the number of times the corresponding color occurs in the image. If the maxcolors value is exceeded, the method stops counting and returns None. The default maxcolors value is 256. To make sure you get all colors in an image, you can pass in size[0]*size[1] (but make sure you have lots of memory before you do that on huge images).

replace (pairs)

replace a color by another currently works only for indexed color images :param pairs: iterable of (from,to) ints

optimize (*maxcolors=256*)

remove unused colors from the palette

crop (*lurb*)

Parameters *lurb* – 4-tuple with left,up,right,bottom int coordinates

Returns Image

__getitem__ (*slice*)

resize (*size, filter=None, **kwargs*)

Resize image

Returns a resized copy of image.

Parameters

- **size** – int tuple (width, height) requested size in pixels
- **filter** –
 - NEAREST (use nearest neighbour),
 - BILINEAR (linear interpolation in a 2x2 environment),
 - BICUBIC (cubic spline interpolation in a 4x4 environment)
 - ANTIALIAS (a high-quality downsampling filter)
- **kwargs** – extra parameters passed to `skimage.transform.resize`

rotate (*angle, **kwargs*)

Rotate image

Returns a rotated copy of image.

Parameters

- **angle** – float rotation angle in degrees in counter-clockwork direction
- **kwargs** – extra parameters passed to `skimage.transform.rotate`

flip (*flipx=True, flipy=False*)

Flip image

Returns a flipped copy of image.

Parameters

- **flipx** – bool flip X direction
- **flipy** – bool flip Y direction
- **kwargs** – extra parameters passed to `skimage.transform.rotate`

paste (*image, box, mask=None*)

Pastes another image into this image.

Parameters

- **image** – image to paste, or color given as a single numerical value for single-band images, and a tuple for multi-band images.
- **box** – 2-tuple giving the upper left corner or 4-tuple defining the left, upper, right, and lower pixel coordinate, or None (same as (0, 0)). If a 4-tuple is given, the size of the pasted image must match the size of the region.

:param mask: optional image to update only the regions indicated by the mask. You can use either “1”, “L” or “RGBA” images (in the latter case, the alpha band is used as mask). Where the mask is 255, the given image is copied as is. Where the mask is 0, the current value is preserved. Intermediate values can be used for transparency effects. Note that if you paste an “RGBA” image, the alpha band is ignored. You can work around this by using the same image as both source image and mask.

threshold (*level=None*)

quantize (*colors=256, method=None, kmeans=0, palette=None*)

(PIL.Image compatible) Convert the image to ‘P’ mode with the specified number of colors. :param colors: The desired number of colors, <= 256 :param method: 0 = median cut

1 = maximum coverage 2 = fast octree 3 = libimagequant

Parameters

- **kmeans** – Integer
- **palette** – Quantize to the `PIL.ImagingPalette` palette.

Returns A new image

convert (*mode, **kwargs*)

convert image mode :param mode: string destination mode :param kwargs: optional params passed to converter(s). can contain: * **palette** : to force using a palette instead of the image’s one for indexed images :return: image in desired mode

__repr__ ()

average_hash (*hash_size=8*)

Average Hash

See <http://www.hackerfactor.com/blog/index.php/?archives/432-Looks-Like-It.html>

Parameters **hash_size** – int sqrt of the hash size. 8 (64 bits) is perfect for usual photos

Returns int of hash_size**2 bits

perceptual_hash (*hash_size=8, highfreq_factor=4*)

Perceptual Hash

See <http://www.hackerfactor.com/blog/index.php/?archives/432-Looks-Like-It.html>

Parameters **hash_size** – int sqrt of the hash size. 8 (64 bits) is perfect for usual photos

Returns int of hash_size**2 bits

dist (*other, method=<function Image.perceptual_hash>, hash_size=8, symmetries=False*)

distance between images

Parameters **hash_size** – int sqrt of the hash size. 8 (64 bits) is perfect for usual photos

Returns float =0 if images are equal or very similar (same average_hash) =1 if images are completely decorrelated (half of the hash bits are the same by luck) =2 if images are inverted

__hash__ ()

__abs__ ()

Returns float Frobenius norm of image

invert ()

__neg__ ()

`__inv__()`

grayscale (*mode=None*)

convert (color) to grayscale :param mode: string target mode (should be in 'FUIL') or automatic if none

colorize (*color0, color1=None*)

colorize a grayscale image

Parameters **color0, color1** – 2 colors. - If only one is specified, image is colorized from white (for 0) to the specified color (for 1) - if 2 colors are specified, image is colorized from color0 (for 0) to color1 (for 1)

Returns RGB(A) color

dither (*method=None, n=2*)

normalize (*newmax=None, newmin=None*)

filter (*f*)

correlation (*other*)

Compute the correlation between two, single-channel, grayscale input images. The second image must be smaller than the first. :param other: the Image we're looking for

scale (*s*)

resize image by factor s

Parameters **s** – (sx, sy) tuple of float scaling factor, or scalar s=sx=sy

Returns Image scaled

shift (*dx, dy, **kwargs*)

expand (*size, ox=None, oy=None*)

Returns image in larger canvas size, pasted at ox,oy

compose (*other, a=0.5, b=0.5, mode=None*)

compose new image from a*self + b*other

add (*other, pos=(0, 0), alpha=1, mode=None*)

simply adds other image at px,py (subbixel) coordinates

`__add__` (*other*)

`__radd__` (*other*)

only to allow sum(images) easily

sub (*other, pos=(0, 0), alpha=1, mode=None*)

`__sub__` (*other*)

deltaE (*other*)

`__mul__` (*other*)

`__div__` (*f*)

`__truediv__` (*f*)

`__class__`

alias of `type`

`__delattr__`

Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__le__
Return self<=value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

html (**kwargs)

plot (**kwargs)
renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

svg (**kwargs)

Goulib.image.**alpha_composite** (*front, back*)
Alpha composite two RGBA images.

Source: <http://stackoverflow.com/a/9166671/284318>

Keyword Arguments: front – PIL RGBA Image object back – PIL RGBA Image object

The algorithm comes from http://en.wikipedia.org/wiki/Alpha_compositing

Goulib.image.**alpha_composite_with_color** (*image, color=(255, 255, 255)*)

Alpha composite an RGBA image with a single color image of the specified color and the same size as the original image.

Keyword Arguments: image – PIL RGBA Image object color – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.pure_pil_alpha_to_color_v1 (image, color=(255, 255, 255))`

Alpha composite an RGBA Image with a specified color.

NOTE: This version is much slower than the `alpha_composite_with_color` solution. Use it only if numpy is not available.

Source: <http://stackoverflow.com/a/9168169/284318>

Keyword Arguments: `image` – PIL RGBA Image object `color` – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.pure_pil_alpha_to_color_v2 (image, color=(255, 255, 255))`

Alpha composite an RGBA Image with a specified color.

Simpler, faster version than the solutions above.

Source: <http://stackoverflow.com/a/9459208/284318>

Keyword Arguments: `image` – PIL RGBA Image object `color` – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.disk (radius, antialias=1)`

`Goulib.image.fspecial (name, **kwargs)`

mimics the Matlab image toolbox `fspecial` function <http://www.mathworks.com/help/images/ref/fspecial.html?refresh=true>

`Goulib.image.normalize (a, newmax=255, newmin=0)`

`Goulib.image.read_pdf (filename, **kwargs)`

reads a bitmap graphics on a .pdf file only the first page is parsed

`Goulib.image.fig2img (fig)`

Convert a Matplotlib figure to a PIL Image in RGBA format and return it

Parameters `fig` – matplotlib figure

Returns PIL image

`Goulib.image.quantize (image, N=2, L=None)`

Quantize a gray image. :param image: ndarray input image. :param N: int number of quantization levels. :param L: float max value.

`Goulib.image.randomize (image, N=2, L=None)`

class `Goulib.image.Ditherer (name, method)`

Bases: `object`

`__init__ (name, method)`

`__call__ (image, N=2)`

`__repr__ ()`

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__dir__ ()` → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__ ()`

default object formatter

```

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

class Goulib.image.**ErrorDiffusion** (*name, positions, weights, wsum=None*)

Bases: [Goulib.image.Ditherer](#)

```

__init__ (name, positions, weights, wsum=None)

__call__ (image, N=2)

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

```

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self > value`.

__hash__
Return `hash(self)`.

__le__
Return `self <= value`.

__lt__
Return `self < value`.

__ne__
Return `self != value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

class `Goulib.image.FloydSteinberg`
Bases: `Goulib.image.ErrorDiffusion`

__init__ ()

__call__ (*image*, *N*=2)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self == value`.

__format__ ()
default object formatter

__ge__
Return `self >= value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

Goulib.image.**dither** (*image*, *method*=3, *N*=2)

Quantize a gray image, using dithering. :param image: ndarray input image. :param method: key in dithering dict :param N: int number of quantization levels. References ——— <http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT>

Goulib.image.**gray2bool** (*image*, *method*=3, *N*=2)

Quantize a gray image, using dithering. :param image: ndarray input image. :param method: key in dithering dict :param N: int number of quantization levels. References ——— <http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT>

Goulib.image.**rgb2cmyk** (*rgb*)

Goulib.image.**cmyk2rgb** (*cmyk*)

Goulib.image.**gray2rgb** (*im*, *color0*=(0, 0, 0), *color1*=(1, 1, 1))

Goulib.image.**bool2rgb** (*im*, *color0*=(0, 0, 0), *color1*=(1, 1, 1))

Goulib.image.**bool2gray** (*im*)

Goulib.image.**rgb2rgba** (*array*)

Goulib.image.**palette** (*im*, *ncolors*, *tol*=0.01)

extract the color palette of image array (in its own colorspace. use Lab for best results) :param im: nparray (x,y,n) containing image :param ncolors: int number of colors :param tol: tolerance for precision/speed compromise. 1/100 means about 100 points per color are taken for kmeans segmentation :return: array of ncolors most used in image (center of kmeans centroids)

`Goulib.image.lab2ind(im, colors=256)`

convert a Lab image to indexed colors :param a: ndarray (x,y,n) containing image :param colors: int number of colors or predefined Palette :ref: http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html

`Goulib.image.ind2any(im, palette, dest)`

`Goulib.image.ind2rgb(im, palette)`

`Goulib.image.convert(a, source, target, **kwargs)`

convert an image between modes, eventually using intermediary steps :param a: ndarray (x,y,n) containing image :param source: string : key of source image mode in modes :param target: string : key of target image mode in modes

2.11 Goulib.interval module

operations on [a..b[intervals

`Goulib.interval.in_interval(interval, x, closed=True)`

Returns bool True if x is in interval [a,b] or [b,a] (tuple)

`Goulib.interval.intersect(t1, t2)`

Returns bool True if intervals [t1[[t2[intersect

`Goulib.interval.intersection(t1, t2)`

Returns tuple intersection between 2 intervals (tuples),

or None if intervals don't intersect

`Goulib.interval.intersectlen(t1, t2, none=0)`

Parameters

- **t1** – interval 1 (tuple)
- **t2** – interval 2 (tuple)
- **none** – value to return when t1 does not intersect t2

Returns len of intersection between 2 intervals (tuples),

or none if intervals don't intersect

class `Goulib.interval.Interval(start, end)`

Bases: list

Represents an interval. Defined as half-open interval [start,end), which includes the start position but not the end. Start and end do not have to be numeric types. They might especially be time, date or timedate as used in datetime2

inspired from <http://code.activestate.com/recipes/576816-interval/> alternatives could be <https://pypi.python.org/pypi/interval/>

(outdated, no more doc) or <https://pypi.python.org/pypi/pyinterval/>

Construct, start must be <= end.

`__init__(start, end)`

Construct, start must be <= end.

start
The interval's start

end
The interval's end

__str__()

__repr__()

__hash__()

__lt__(other)

__eq__(other)

size

center

separation(other)
Returns distance between self and other, negative if overlap

overlap(other, allow_contiguous=False)
Returns True iff self intersects other.

intersection(other)
Returns Intersection with other, or None if no intersection.

__iadd__(other)
expands self to contain other.

hull(other)
Returns new Interval containing both self and other.

__add__(other)

__contains__(x)
Returns True if x in self.

subset(other)
Returns True iff self is subset of other.

proper_subset(other)
Returns True iff self is proper subset of other.

empty()
Returns True iff self is empty.

__nonzero__()

singleton()
Returns True iff self.end - self.start == 1.

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

`__delitem__`
Delete self[key].

`__dir__` () → list
default dir() implementation

`__format__` ()
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__getitem__` ()
x.__getitem__(y) <==> x[y]

`__gt__`
Return self>value.

`__imul__`
Implement self*=value.

`__iter__`
Implement iter(self).

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__mul__`
Return self*value.n

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__reversed__` ()
L.__reversed__() – return a reverse iterator over the list

`__rmul__`
Return self*value.

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

`__sizeof__` ()
L.__sizeof__() – size of L in memory, in bytes

`append` (object) → None – append object to end

clear () → None – remove all items from L

copy () → list – a shallow copy of L

count (value) → integer – return number of occurrences of value

extend (iterable) → None – extend list by appending elements from the iterable

index (value[, start[, stop]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

pop ([index]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (value) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

sort (key=None, reverse=False) → None – stable sort **IN PLACE**

class Goulib.interval.**Intervals** (iterable=None, key=<function identity>, load=1000)

Bases: sortedcontainers.sortedlist.SortedListWithKey

a list of intervals kept in ascending order

SortedListWithKey provides most of the same methods as list but keeps the items in sorted order.

An optional *iterable* provides an initial series of items to populate the SortedListWithKey.

An optional *key* argument defines a callable that, like the *key* argument to Python's *sorted* function, extracts a comparison key from each element. The default is the identity function.

An optional *load* specifies the load-factor of the list. The default load factor of '1000' works well for lists from tens to tens of millions of elements. Good practice is to use a value that is the cube root of the list size. With billions of elements, the best load factor depends on your usage. It's best to leave the load factor at the default until you start benchmarking.

update (iterable)
Update the list by adding all elements from *iterable*.

add (item)

insert (item)

__iadd__ (item)

__add__ (item)

__call__ (x)
returns intervals containing x

__str__ ()
string representation : like a list of Intervals

__abstractmethods__ = frozenset()

__class__
alias of ABCMeta

__contains__ (val)
Return True if and only if *val* is an element in the list.

`__copy__()`
Return a shallow copy of the sorted list.

`__delattr__`
Implement `delattr(self, name)`.

`__delitem__` (*idx*)
Remove the element at *idx*. Supports slicing.

`__dir__()` → list
default `dir()` implementation

`__eq__` (*that*)
Return *True* if and only if Sequence is equal to *that*.

`__format__()`
default object formatter

`__ge__` (*that*)
Return *True* if and only if Sequence is greater than or equal to *that*.

`__getattr__`
Return `getattr(self, name)`.

`__getitem__` (*idx*)
Return the element at *idx*. Supports slicing.

`__gt__` (*that*)
Return *True* if and only if Sequence is greater than *that*.

`__hash__` = None

`__imul__` (*that*)
Increase the length of the list by appending *that* shallow copies of each item.

`__init__` (*iterable=None*, *key=<function identity>*, *load=1000*)
SortedListWithKey provides most of the same methods as list but keeps the items in sorted order.

An optional *iterable* provides an initial series of items to populate the SortedListWithKey.

An optional *key* argument defines a callable that, like the *key* argument to Python's *sorted* function, extracts a comparison key from each element. The default is the identity function.

An optional *load* specifies the load-factor of the list. The default load factor of '1000' works well for lists from tens to tens of millions of elements. Good practice is to use a value that is the cube root of the list size. With billions of elements, the best load factor depends on your usage. It's best to leave the load factor at the default until you start benchmarking.

`__iter__` ()
Return an iterator over the Sequence.

Iterating the Sequence while adding or deleting values may raise a *RuntimeError* or fail to iterate over all entries.

`__le__` (*that*)
Return *True* if and only if Sequence is less than or equal to *that*.

`__len__` ()
Return the number of elements in the list.

`__lt__` (*that*)
Return *True* if and only if Sequence is less than *that*.

__mul__ (*that*)
Return a new sorted list containing *that* shallow copies of each item in SortedListWithKey.

__ne__ (*that*)
Return *True* if and only if Sequence is not equal to *that*.

__new__ (*iterable=None*, *key=<function identity>*, *load=1000*)

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()
Return string representation of sequence.

__reversed__ ()
Return an iterator to traverse the Sequence in reverse.

Iterating the Sequence while adding or deleting values may raise a *RuntimeError* or fail to iterate over all entries.

__setattr__
Implement setattr(self, name, value).

__setitem__ (*index*, *value*)
Replace the item at position *index* with *value*.

Supports slice notation. Raises a *ValueError* if the sort order would be violated. When used with a slice and iterable, the *ValueError* is raised before the list is mutated if the sort order would be violated by the operation.

__sizeof__ () → int
size of object in memory, in bytes

__slots__ = ()

append (*val*)
Append the element *val* to the list. Raises a *ValueError* if the *val* would violate the sort order.

bisect (*val*)
Same as *bisect_left*, but if *val* is already present, the insertion point will be after (to the right of) any existing entries.

bisect_key (*key*)
Same as *bisect_key_left*, but if *key* is already present, the insertion point will be after (to the right of) any existing entries.

bisect_key_left (*key*)
Similar to the *bisect* module in the standard library, this returns an appropriate index to insert a value with a given *key*. If values with *key* are already present, the insertion point will be before (to the left of) any existing entries.

bisect_key_right (*key*)
Same as *bisect_key_left*, but if *key* is already present, the insertion point will be after (to the right of) any existing entries.

bisect_left (*val*)
Similar to the *bisect* module in the standard library, this returns an appropriate index to insert *val*. If *val* is already present, the insertion point will be before (to the left of) any existing entries.

bisect_right (*val*)

Same as *bisect_left*, but if *val* is already present, the insertion point will be after (to the right of) any existing entries.

clear ()

Remove all the elements from the list.

copy ()

Return a shallow copy of the sorted list.

count (*val*)

Return the number of occurrences of *val* in the list.

discard (*val*)

Remove the first occurrence of *val*.

If *val* is not a member, does nothing.

extend (*values*)

Extend the list by appending all elements from the *values*. Raises a *ValueError* if the sort order would be violated.

index (*val*, *start*=None, *stop*=None)

Return the smallest *k* such that *L*[*k*] == *val* and *i* <= *k* < *j*'. Raises *ValueError* if *val* is not present. *stop* defaults to the end of the list. *start* defaults to the beginning. Negative indices are supported, as for slice indices.

irange (*minimum*=None, *maximum*=None, *inclusive*=(True, True), *reverse*=False)

Create an iterator of values between *minimum* and *maximum*.

inclusive is a pair of booleans that indicates whether the minimum and maximum ought to be included in the range, respectively. The default is (True, True) such that the range is inclusive of both minimum and maximum.

Both *minimum* and *maximum* default to *None* which is automatically inclusive of the start and end of the list, respectively.

When *reverse* is *True* the values are yielded from the iterator in reverse order; *reverse* defaults to *False*.

irange_key (*min_key*=None, *max_key*=None, *inclusive*=(True, True), *reverse*=False)

Create an iterator of values between *min_key* and *max_key*.

inclusive is a pair of booleans that indicates whether the *min_key* and *max_key* ought to be included in the range, respectively. The default is (True, True) such that the range is inclusive of both *min_key* and *max_key*.

Both *min_key* and *max_key* default to *None* which is automatically inclusive of the start and end of the list, respectively.

When *reverse* is *True* the values are yielded from the iterator in reverse order; *reverse* defaults to *False*.

islice (*start*=None, *stop*=None, *reverse*=False)

Returns an iterator that slices *self* from *start* to *stop* index, inclusive and exclusive respectively.

When *reverse* is *True*, values are yielded from the iterator in reverse order.

Both *start* and *stop* default to *None* which is automatically inclusive of the beginning and end.

pop (*idx*=-1)

Remove and return item at *idx* (default last). Raises *IndexError* if list is empty or index is out of range. Negative indices are supported, as for slice indices.

remove (*val*)

Remove first occurrence of *val*.

Raises ValueError if *val* is not present.

reverse ()
S.reverse() – reverse *IN PLACE*

class Goulib.interval.Box (*args)
Bases: list

a N dimensional rectangular box defined by a list of N Intervals

__init__ (*args)

corner (n)
return n-th corner of box 0-th corner is “start” made of all minimal values of intervals -1.th corner is “end”, made of all maximal values of intervals

start

end

size

center

__call__ ()
Returns tuple of all intervals as tuples

__iadd__ (other)
enlarge box if required to contain specified point :param other: *Box* or (list of) N-tuple point(s)

__add__ (other)
enlarge box if required to contain specified point :param other: *Box* or (list of) N-tuple point(s) :return: new Box containing both

__contains__ (other)
Returns True if x in self.

__nonzero__ ()

empty ()
Returns True iff Box is empty.

__class__
alias of *type*

__delattr__
Implement delattr(self, name).

__delitem__
Delete self[key].

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

`__getattr__`
Return `getattr(self, name)`.

`__getitem__` (`y`)
`x.__getitem__(y) <==> x[y]`

`__gt__`
Return `self > value`.

`__hash__` = `None`

`__imul__`
Implement `self *= value`.

`__iter__`
Implement `iter(self)`.

`__le__`
Return `self <= value`.

`__len__`
Return `len(self)`.

`__lt__`
Return `self < value`.

`__mul__`
Return `self * value.n`

`__ne__`
Return `self != value`.

`__new__` (`y`)
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` (`y`)
helper for pickle

`__reduce_ex__` (`y`)
helper for pickle

`__repr__`
Return `repr(self)`.

`__reversed__` (`y`)
`L.__reversed__()` – return a reverse iterator over the list

`__rmul__`
Return `self * value`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__setitem__`
Set `self[key]` to value.

`__sizeof__` (`y`)
`L.__sizeof__()` – size of L in memory, in bytes

`__str__`
Return `str(self)`.

`append` (`object`) → `None` – append object to end

`clear` (`y`) → `None` – remove all items from L

copy () → list – a shallow copy of L

count (value) → integer – return number of occurrences of value

extend (iterable) → None – extend list by appending elements from the iterable

index (value[, start[, stop]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

max

min

pop ([index]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (value) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

sort (key=None, reverse=False) → None – stable sort **IN PLACE**

2.12 Goulib.itertools2 module

additions to `itertools` standard library

`Goulib.itertools2.take` (n, iterable)

Result first n items from iterable

`Goulib.itertools2.index` (n, iterable)

Result nth item

`Goulib.itertools2.first` (iterable)

Result first element in the iterable

`Goulib.itertools2.last` (iterable)

Result last element in the iterable

`Goulib.itertools2.takeevery` (n, iterable, start=0)

Take an element from iterator every n elements

`Goulib.itertools2.every` (n, iterable, start=0)

Take an element from iterator every n elements

`Goulib.itertools2.drop` (n, iterable)

Drop n elements from iterable and return the rest

`Goulib.itertools2.ilen` (it)

Result int length exhausting an iterator

`Goulib.itertools2.irange` (start_or_end, optional_end=None)

Result iterable that counts from start to end (both included).

`Goulib.itertools2.isiterable` (obj)

Result bool True if obj is iterable (but not a string)

`Goulib.itertools2.iscallable(f)`

`Goulib.itertools2.enumerates(iterable)`

generalizes enumerate to dicts :result: key,value pair for whatever iterable type

`Goulib.itertools2.arange(start, stop=None, step=1)`

range for floats or other types (*numpy.arange* without numpy)

Parameters

- **start** – optional number. Start of interval. The interval includes this value. The default start value is 0.
- **stop** – number. End of interval. The interval does not include this value, except in some cases where step is not an integer and floating point round-off affects the length of out.
- **step** – optional number. Spacing between values. For any output out, this is the distance between two adjacent values, out[i+1] - out[i]. The default step size is 1.

Result iterator

`Goulib.itertools2.linspace(start, end, n=100)`

iterator over n values linearly interpolated between (and including) start and end *numpy.linspace* without numpy

Parameters

- **start** – number, or iterable vector
- **end** – number, or iterable vector
- **n** – int number of interpolated values

Result iterator

`Goulib.itertools2.flatten(l, donotrecursein=(<class 'str'>,))`

iterator to flatten (depth-first) structure

Parameters

- **l** – iterable structure
- **donotrecursein** – iterable types in which algo doesn't recurse string type by default

`Goulib.itertools2.itemgetter(iterable, i)`

`Goulib.itertools2.compact(iterable, f=<class 'bool'>)`

Returns iterator skipping None values from iterable

`Goulib.itertools2.compress(iterable)`

generates (item,count) pairs by counting the number of consecutive items in iterable)

`Goulib.itertools2.recurrence(f, x)`

`Goulib.itertools2.swap(iterable)`

`Goulib.itertools2.tee(iterable, n=2, copy=None)`

tee or copy depending on type and goal

Parameters

- **iterable** – any iterable
- **n** – int number of tees/copies to return
- **copy** – optional copy function, for exemple copy.copy or copy.deepcopy

Result tee of iterable if it's an iterator or generator, or (deep)copies for other types
 this function is useful to avoid side effects at a lower memory cost depending on the case

Goulib.itertools2.**groups** (*iterable*, *n*, *step=None*)

Make groups of 'n' elements from the iterable advancing 'step' elements on each iteration

Goulib.itertools2.**pairwise** (*iterable*, *op=None*, *loop=False*)

iterates through consecutive pairs

Parameters

- **iterable** – input iterable s1,s2,s3, sn
- **op** – optional operator to apply to each pair
- **loop** – boolean True if last pair should be (sn,s1) to close the loop

Result pairs iterator (s1,s2), (s2,s3) ... (si,si+1), ... (sn-1,sn) + optional pair to close the loop

Goulib.itertools2.**shape** (*iterable*)

shape of a mutidimensional array, without numpy

Parameters **iterable** – iterable of iterable ... of iterable or numpy arrays...

Result list of n ints corresponding to iterable's len of each dimension

Warning if iterable is not a (hyper) rect matrix, shape is evaluated from

the [0,0,...0] element ... :see: <http://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.ndarray.shape.html>

Goulib.itertools2.**ndim** (*iterable*)

number of dimensions of a mutidimensional array, without numpy

Parameters **iterable** – iterable of iterable ... of iterable or numpy arrays...

Result int number of dimensions

Goulib.itertools2.**reshape** (*data*, *dims*)

Result data as a n-dim matrix

Goulib.itertools2.**compose** (*f*, *g*)

Compose two functions -> compose(f, g)(x) -> f(g(x))

Goulib.itertools2.**iterate** (*func*, *arg*)

After Haskell's iterate: apply function repeatedly.

Goulib.itertools2.**accumulate** (*iterable*, *func=<built-in function add>*, *skip_first=False*)

Return running totals. extends *python.accumulate*

accumulate([1,2,3,4,5]) -> 1 3 6 10 15 # accumulate([1,2,3,4,5], operator.mul) -> 1 2 6 24 120

Goulib.itertools2.**tails** (*seq*)

Get tails of a sequence

tails([1,2,3]) -> [1,2,3], [2,3], [3], [].

Goulib.itertools2.**ireduce** (*func*, *iterable*, *init=None*)

Like *python.reduce* but using iterators (a.k.a scanl)

Goulib.itertools2.**unique** (*iterable*, *key=None*, *buffer=None*)

generate unique elements, preserving order. :param iterable: iterable :param key: optional function defining which elements are considered equal :param buffer: optional integer defining how many of the last unique elements to keep in memory

`# unique('AAAABBBCCDAABBB') -> A B C D # unique('ABBCcAD', str.lower) -> A B C D`

`Goulib.itertools2.count_unique(iterable, key=None)`
Count unique elements

`# count_unique('AAAABBBCCDAABBB') -> 4 # count_unique('ABBCcAD', str.lower) -> 4`

`Goulib.itertools2.identity(x)`
Do nothing and return the variable untouched

`Goulib.itertools2.occurrences(it, exchange=False)`
Return dictionary with occurrences from iterable

`Goulib.itertools2.cartesian_product(*iterables, **kwargs)`
<http://stackoverflow.com/questions/12093364/cartesian-product-of-large-iterators-itertools>

`Goulib.itertools2.combinations_with_replacement(iterable, r)`
`combinations_with_replacement('ABC', 2) -> AA AB AC BB BC CC` same as `combinations_with_replacement` except it doesn't generate duplicates

`Goulib.itertools2.any(seq, pred=<class 'bool'>)`
Result bool True if `pred(x)` is True for at least one element in the iterable

`Goulib.itertools2.all(seq, pred=<class 'bool'>)`
Result bool True if `pred(x)` is True for all elements in the iterable

`Goulib.itertools2.no(seq, pred=<class 'bool'>)`
Result bool True if `pred(x)` is False for every element in the iterable

`Goulib.itertools2.takenth(n, iterable, default=None)`
Result nth item of iterable

`Goulib.itertools2.nth(n, iterable, default=None)`
Result nth item of iterable

`Goulib.itertools2.icross(*sequences)`
Cartesian product of sequences (recursive version)

`Goulib.itertools2.quantify(iterable, pred=<class 'bool'>)`
Result int count how many times the predicate is true

`Goulib.itertools2.interleave(l1, l2)`
Parameters

- **l1** – iterable
- **l2** – iterable of same length, or 1 less than l1

Result iterable interleaving elements from l1 and l2, starting by l1[0]

`Goulib.itertools2.shuffle(ary)`
Param array to shuffle by Fisher-Yates algorithm
Result shuffled array (IN PLACE!)

See <http://www.drgoulu.com/2013/01/19/comment-bien-brasser-les-cartes/>

`Goulib.itertools2.rand_seq(size)`
Result range(size) shuffled

Goulib.itertools2.**all_pairs** (*size*)

generates all i,j pairs for i,j from 0-size

Goulib.itertools2.**index_min** (*values, key=<function identity>*)

Result min_index, min_value

Goulib.itertools2.**index_max** (*values, key=<function identity>*)

Result max_index, max_value

Goulib.itertools2.**best** (*iterable, key=None, n=1, reverse=False*)

generate items corresponding to the n best values of key sort order

Goulib.itertools2.**sort_indexes** (*iterable, key=<function identity>, reverse=False*)

Returns iterator over indexes of iterable that correspond to the sorted iterable

Goulib.itertools2.**filter2** (*iterable, condition*)

like *python.filter* but returns 2 lists : - list of elements in iterable that satisfy condition - list of those that don't

Goulib.itertools2.**ifind** (*iterable, f, reverse=False*)

iterates through items in iterable where f(item) == True.

Goulib.itertools2.**iremove** (*iterable, f*)

removes items from an iterable based on condition :param iterable: iterable . will be modified in place :param f: function of the form lambda line:bool returning True if item should be removed :yield: removed items backwards

Goulib.itertools2.**removef** (*iterable, f*)

removes items from an iterable based on condition :param iterable: iterable . will be modified in place :param f: function of the form lambda line:bool returning True if item should be removed :result: list of removed items.

Goulib.itertools2.**find** (*iterable, f*)

Return first item in iterable where f(item) == True.

Goulib.itertools2.**isplit** (*iterable, sep, include_sep=False*)

split iterable by separators or condition :param sep: value or function(item) returning True for items that separate :param include_sep: bool. If True the separators items are included in output, at beginning of each sub-iterator :result: iterates through slices before, between, and after separators

Goulib.itertools2.**split** (*iterable, sep, include_sep=False*)

like <https://docs.python.org/2/library/stdtypes.html#str.split>, but for iterable :param sep: value or function(item) returning True for items that separate :param include_sep: bool. If True the separators items are included in output, at beginning of each sub-iterator :result: list of iterable slices before, between, and after separators

Goulib.itertools2.**dictsplit** (*dic, keys*)

extract keys from dic :param dic: dict source :param keys: iterable of dict keys :result: dict,dict : the first contains entries present in source, the second the remaining entries

Goulib.itertools2.**next_permutation** (*seq, pred=<function <lambda>>*)

Like C++ std::next_permutation() but implemented as generator. see <http://blog.bjrn.se/2008/04/lexicographic-permutations-using.html> :param seq: iterable :param pred: a function (a,b) that returns a negative number if a<b, like cmp(a,b) in Python 2.7

class Goulib.itertools2.**iter2** (*iterable*)

Bases: *object*

Takes in an object that is iterable. <http://code.activestate.com/recipes/578092-flattening-an-arbitrarily-deep-list-or-any-iterato/> Allows for the following method calls (that should be built into iterators anyway...) calls: - append - appends another iterable onto the iterator. - insert - only accepts inserting at the 0 place, inserts an iterable before other iterables. - adding. an iter2 object can be added to another object that is iterable. i.e. iter2 + iter (not iter + iter2). It's best to make all objects iter2 objects to avoid syntax errors. :D

`__init__` (*iterable*)

`append` (*iterable*)

`insert` (*place, iterable*)

`__add__` (*iterable*)

`__next__` ()

`next` ()

`__iter__` ()

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

Goulib.itertools2.**subdict** (d, keys)

extract “sub-dictionary” :param d: dict :param keys: container of keys
to extract: :result: dict: :see: <http://stackoverflow.com/questions/5352546/best-way-to-extract-subset-of-key-value-pairs-from-python-dictionary-object/5352649#5352649>

exception Goulib.itertools2.**SortingError**

Bases: Exception

__cause__
exception cause

__class__
alias of `type`

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

Goulib.itertools2.**ensure_sorted** (iterable, key=None)
makes sure iterable is sorted according to key

Yields items of iterable

Raise SortingError if not

Goulib.itertools2.**sorted_iterable** (iterable, key=None, buffer=100)
sorts an “almost sorted” (infinite) iterable

Parameters

- **iterable** – iterable
- **key** – function used as sort key
- **buffer** – int size of buffer. elements to swap should not be further than that

Goulib.itertools2.**diff** (iterable1, iterable2)
generate items in sorted iterable1 that are not in sorted iterable2

Goulib.itertools2.**intersect** (*its)
generates intersection of N iterables

Parameters **its** – any number of SORTED iterables

Yields elements that belong to all iterables

See <http://stackoverflow.com/questions/969709/joining-a-set-of-ordered-integer-yielding-python-iterators>

class Goulib.itertools2.**keep** (iterable)
Bases: collections.abc.Iterator

iterator that keeps the last value

__init__ (iterable)

__next__ ()

next ()

__abstractmethods__ = frozenset()

```

__class__
    alias of ABCMeta

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__iter__ ()

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__slots__ = ()

__str__
    Return str(self).

```

```

Goulib.itertools2.first_match (iter1, iter2, limit=None)
    ”:param limit: int max number of loops :return: integer i first index where iter1[i]==iter2[i]

```

`Goulib.itertools2.floyd(iterable, limit=1000000.0)`

Detect a cycle in iterable using Floyd “tortue hand hare” algorithm

See https://en.wikipedia.org/wiki/Cycle_detection

Parameters

- **iterable** – iterable
- **limit** – int limit to prevent infinite loop. no limit if None

Result (i,l) tuple of integers where i=index of cycle start, l=length if no cycle is found, return (None,None)

`Goulib.itertools2.detect_cycle(iterable, limit=1000000.0)`

2.13 Goulib.markup module

simple HTML/XML generation (forked from [markup](#))

`Goulib.markup.cgiprint(line='', unbuff=True, line_end='\r\n')`

Print to the stdout. :param line: string to print, followed by line_end :param unbuff: boolean, True to flush the buffer after every write. :param line_end: string to print after each line. By default this is

, which is the standard specified by the RFC for http headers.

`Goulib.markup.style_dict2str(style)`

`Goulib.markup.style_str2dict(style)`

`Goulib.markup.tag(tag, between, **kwargs)`
generate full tag.

class `Goulib.markup.element(tag, case='lower', parent=None)`
Bases: `object`

This class handles the addition of a new element.

`__init__` (tag, case='lower', parent=None)

`__call__` (*args, **kwargs)

render (t, single, args, **kwargs)

Append the actual tags to content.

close ()

Append a closing tag unless element has only opening tag.

open (*kwargs)

Append an opening tag.

`__class__`

alias of `type`

`__delattr__`

Implement delattr(self, name).

`__dir__` () → list

default dir() implementation

`__eq__`

Return self==value.

```

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

class Goulib.markup.**page** (*mode='strict_html', case='lower', onetags=None, twotags=None, separator='n', class_=None*)

Bases: `object`

This is our main class representing a document. Elements are added as attributes of an instance of this class.

Stuff that effects the whole document.

Parameters **mode** – string. can be either:

- 'strict_html' for HTML 4.01 (default)
- 'html' alias for 'strict_html'
- 'loose_html' to allow some deprecated elements
- 'xml' to allow arbitrary elements

Parameters **case** – string. can be either:

- ‘lower’ element names will be printed in lower case (default)
- ‘upper’ they will be printed in upper case
- ‘given’ element names will be printed as they are given

Parameters

- **onetags** – list or tuple of valid elements with opening tags only
- **twotags** – list or tuple of valid elements with both opening and closing tags

these two keyword arguments may be used to select the set of valid elements in ‘xml’ mode invalid elements will raise appropriate exceptions

Parameters

- **separator** – string to place between added elements, defaults to newline
- **class** – a class that will be added to every element if defined

__init__ (*mode*=‘strict_html’, *case*=‘lower’, *onetags*=None, *twotags*=None, *separator*=‘\n’,
class_=None)

Stuff that effects the whole document.

Parameters mode – string. can be either:

- ‘strict_html’ for HTML 4.01 (default)
- ‘html’ alias for ‘strict_html’
- ‘loose_html’ to allow some deprecated elements
- ‘xml’ to allow arbitrary elements

Parameters case – string. can be either:

- ‘lower’ element names will be printed in lower case (default)
- ‘upper’ they will be printed in upper case
- ‘given’ element names will be printed as they are given

Parameters

- **onetags** – list or tuple of valid elements with opening tags only
- **twotags** – list or tuple of valid elements with both opening and closing tags

these two keyword arguments may be used to select the set of valid elements in ‘xml’ mode invalid elements will raise appropriate exceptions

Parameters

- **separator** – string to place between added elements, defaults to newline
- **class** – a class that will be added to every element if defined

__getattr__ (*attr*)

__str__ ()

__call__ (*escape*=False)

Return the document as a string.

escape – False print normally

True replace < and > by < and > the default escape sequences in most browsers

add (*text*)

This is an alias to addcontent.

addfooter (*text*)

Add some text to the bottom of the document

addheader (*text*)

Add some text to the top of the document

addcontent (*text*)

Add some text to the main part of the document

init (*lang='en', css=None, metainfo=None, title=None, header=None, footer=None, charset=None, encoding=None, doctype=None, bodyattrs=None, script=None, base=None*)

This method is used for complete documents with appropriate doctype, encoding, title, etc information. For an /XML snippet omit this method.

lang – language, usually a two character string, will appear as <html lang='en'> in html mode (ignored in xml mode)

css – Cascading Style Sheet filename as a string or a list of strings for multiple css files (ignored in xml mode)

metainfo – a dictionary in the form { 'name': 'content' } to be inserted into meta element(s) as <meta name='name' content='content'> (ignored in xml mode)

base – set the <base href="..."> tag in <head>

bodyattrs – a dictionary in the form { 'key': 'value', ... } which will be added as attributes of the <body> element as <body key='value' ... > (ignored in xml mode)

script – dictionary containing src:type pairs, <script type='text/type' src=src></script> or a list of ['src1', 'src2', ...] in which case 'javascript' is assumed for all

title – the title of the document as a string to be inserted into a title element as <title>my title</title> (ignored in xml mode)

header – some text to be inserted right after the <body> element (ignored in xml mode)

footer – some text to be inserted right before the </body> element (ignored in xml mode)

charset – a string defining the character set, will be inserted into a <meta http-equiv='Content-Type' content='text/html; charset=myset'> element (ignored in xml mode)

encoding – a string defining the encoding, will be put into to first line of the document as <?xml version='1.0' encoding='myencoding' ?> in xml mode (ignored in html mode)

doctype – the document type string, defaults to <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> in html mode (ignored in xml mode)

css (*filelist*)

This convenience function is only useful for html. It adds css stylesheet(s) to the document via the <link> element.

metainfo (*mydict*)

This convenience function is only useful for html. It adds meta information via the <meta> element, the argument is a dictionary of the form { 'name': 'content' }.

scripts (*mydict*)

Only useful in html, mydict is dictionary of src:type pairs or a list of script sources ['src1', 'src2', ...] in which case 'javascript' is assumed for type. Will be rendered as <script type='text/type' src=src></script>

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return `repr(self)`.

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

`Goulib.markup.escape` (*text*, *newline=False*)
Escape special html characters.

`Goulib.markup.unescape` (*text*)
Inverse of `escape`.

class `Goulib.markup.dummy`
Bases: `object`

A dummy class for attaching attributes.

```

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__init__
    Initialize self. See help(type(self)) for accurate signature.

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

```

class Goulib.markup.russell
    Bases: object

    A dummy class that contains anything.

```

__contains__ (*item*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__init__
Initialize self. See `help(type(self))` for accurate signature.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return `repr(self)`.

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

exception `Goulib.markup.MarkupError`
Bases: `Exception`

All our exceptions subclass this.

```

__str__()
__cause__
    exception cause
__class__
    alias of type
__context__
    exception context
__delattr__
    Implement delattr(self, name).
__dir__() → list
    default dir() implementation
__eq__
    Return self==value.
__format__()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__
    Return hash(self).
__init__
    Initialize self. See help(type(self)) for accurate signature.
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__()
    Create and return a new object. See help(type) for accurate signature.
__reduce__()
__reduce_ex__()
    helper for pickle
__repr__
    Return repr(self).
__setattr__
    Implement setattr(self, name, value).
__setstate__()

```

__sizeof__ () → int
size of object in memory, in bytes

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.ClosingError (tag)
Bases: *Goulib.markup.MarkupError*

__init__ (tag)

__cause__
exception cause

__class__
alias of `type`

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

```

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
    size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
    Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.
exception Goulib.markup.OpeningError (tag)
    Bases: Goulib.markup.MarkupError

    __init__ (tag)

    __cause__
        exception cause

    __class__
        alias of type

    __context__
        exception context

    __delattr__
        Implement delattr(self, name).

    __dir__ () → list
        default dir() implementation

    __eq__
        Return self==value.

    __format__ ()
        default object formatter

    __ge__
        Return self>=value.

    __getattribute__
        Return getattr(self, name).

    __gt__
        Return self>value.

    __hash__
        Return hash(self).

    __le__
        Return self<=value.

```

__lt__
Return self<value.

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__reduce__()

__reduce_ex__()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__()

__sizeof__() → int
size of object in memory, in bytes

__str__()

__suppress_context__

__traceback__

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.**ArgumentError** (*tag*)
Bases: *Goulib.markup.MarkupError*

__init__ (*tag*)

__cause__
exception cause

__class__
alias of *type*

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__() → list
default dir() implementation

__eq__
Return self==value.

__format__()
default object formatter

__ge__
Return self>=value.

```

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
    size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
    Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

```

exception `Goulib.markup.InvalidElementError (tag, mode)`
 Bases: `Goulib.markup.MarkupError`

```

__cause__
    exception cause

__class__
    alias of type

__context__
    exception context

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

```

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__ (tag, mode)

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.**DeprecationError** (tag)
Bases: *Goulib.markup.MarkupError*

__cause__
exception cause

__class__
alias of `type`

__context__
exception context

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return `repr(self)`.

__setattr__
Implement `setattr(self, name, value)`.

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

__init__(tag)

exception Goulib.markup.ModeError(mode)

Bases: *Goulib.markup.MarkupError*

__cause__

exception cause

__class__

alias of *type*

__context__

exception context

__delattr__

Implement delattr(self, name).

__dir__() → list

default dir() implementation

__eq__

Return self==value.

__format__()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__

Return hash(self).

__le__

Return self<=value.

__lt__

Return self<value.

__ne__

Return self!=value.

__new__()

Create and return a new object. See help(type) for accurate signature.

__reduce__()

__reduce_ex__()

helper for pickle

__repr__

Return repr(self).

__setattr__

Implement setattr(self, name, value).

__setstate__()

```

__sizeof__() → int
    size of object in memory, in bytes

__str__()

__suppress_context__

__traceback__

args

with_traceback()
    Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

__init__(mode)

exception Goulib.markup.CustomizationError
    Bases: Goulib.markup.MarkupError

    __cause__
        exception cause

    __class__
        alias of type

    __context__
        exception context

    __delattr__
        Implement delattr(self, name).

    __dir__() → list
        default dir() implementation

    __eq__
        Return self==value.

    __format__()
        default object formatter

    __ge__
        Return self>=value.

    __getattr__
        Return getattr(self, name).

    __gt__
        Return self>value.

    __hash__
        Return hash(self).

    __le__
        Return self<=value.

    __lt__
        Return self<value.

    __ne__
        Return self!=value.

    __new__()
        Create and return a new object. See help(type) for accurate signature.

    __reduce__()

```

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__setstate__()`

`__sizeof__()` → int
size of object in memory, in bytes

`__str__()`

`__suppress_context__`

`__traceback__`

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`__init__()`

2.14 Goulib.math2 module

more math than `math` standard library, without numpy

`Goulib.math2.is_number(x)`

Returns True if x is a number of any type

`Goulib.math2.sign(number)`

Returns 1 if number is positive, -1 if negative, 0 if ==0

`Goulib.math2.cmp(x, y)`

Compare the two objects x and y and return an integer according to the outcome. The return value is negative if $x < y$, zero if $x == y$ and strictly positive if $x > y$.

`Goulib.math2.gcd(*args)`

greatest common divisor of an arbitrary number of args

`Goulib.math2.lcm(*args)`

least common multiple of any number of integers

`Goulib.math2.xgcd(a, b)`

Extended GCD

Returns (gcd, x, y) where gcd is the greatest common divisor of a and b

with the sign of b if b is nonzero, and with the sign of a if b is 0. The numbers x,y are such that $\text{gcd} = ax+by$.

`Goulib.math2.coprime(*args)`

Returns True if args are coprime to each other

`Goulib.math2.coprimes_gen(limit)`

generates coprime pairs using Farey sequence

`Goulib.math2.is_primitive_root(x, m, s={})`
 returns True if x is a primitive root of m

Parameters *s* – set of coprimes to m, if already known

`Goulib.math2.primitive_root_gen(m)`
 generate primitive roots modulo m

`Goulib.math2.primitive_roots(modulo)`

`Goulib.math2.quad(a, b, c, allow_complex=False)`
 solves quadratic equations $aX^2+bX+c=0$

Parameters

- *a, b, c* – floats
- *allow_complex* – function returns complex roots if True

Returns *x1, x2* real or complex solutions

`Goulib.math2.isclose(a, b, rel_tol=1e-09, abs_tol=0.0)`
 approximately equal. Use this instead of `a==b` in floating point ops

implements <https://www.python.org/dev/peps/pep-0485/> :param *a, b*: the two values to be tested to relative closeness :param *rel_tol*: relative tolerance

it is the amount of error allowed, relative to the larger absolute value of *a* or *b*. For example, to set a tolerance of 5%, pass `tol=0.05`. The default tolerance is `1e-9`, which assures that the two values are the same within about 9 decimal digits. *rel_tol* must be greater than 0.0

Parameters *abs_tol* – minimum absolute tolerance level – useful for comparisons near zero.

`Goulib.math2.is_integer(x, epsilon=1e-06)`

Returns True if float *x* is almost an integer

`Goulib.math2 rint(v)`

Returns int value nearest to float *v*

`Goulib.math2.int_or_float(x, epsilon=1e-06)`

Parameters *x* – int or float

Returns int if *x* is (almost) an integer, otherwise float

`Goulib.math2.format(x, decimals=3)`
 formats a float with given number of decimals, but not an int

Returns string repr of *x* with decimals if not int

`Goulib.math2.ceildiv(a, b)`

`Goulib.math2.isqrt(n)`
 integer square root

Returns largest int *x* for which $x * x \leq n$

`Goulib.math2.sqrt(n)`
 square root :return: float, or int if *n* is a perfect square

`Goulib.math2.is_square(n)`

`Goulib.math2.multiply(x, y)`

Karatsuba fast multiplication algorithm

https://en.wikipedia.org/wiki/Karatsuba_algorithm

Copyright (c) 2014 Project Nayuki <http://www.nayuki.io/page/karatsuba-multiplication>

`Goulib.math2.accsum(it)`

Yield accumulated sums of iterable: `accsum(count(1)) -> 1,3,6,10,...`

`Goulib.math2.cumsum(it)`

Yield accumulated sums of iterable: `accsum(count(1)) -> 1,3,6,10,...`

`Goulib.math2.mul(nums, init=1)`

Returns Product of nums

`Goulib.math2.dot_vv(a, b, default=0)`

dot product for vectors

Parameters

- **a** – vector (iterable)
- **b** – vector (iterable)
- **default** – default value of the multiplication operator

`Goulib.math2.dot_mv(a, b, default=0)`

dot product for vectors

Parameters

- **a** – matrix (iterable or iterables)
- **b** – vector (iterable)
- **default** – default value of the multiplication operator

`Goulib.math2.dot_mm(a, b, default=0)`

dot product for matrices

Parameters

- **a** – matrix (iterable or iterables)
- **b** – matrix (iterable or iterables)
- **default** – default value of the multiplication operator

`Goulib.math2.dot(a, b, default=0)`

dot product

general but slow : use `dot_vv`, `dot_mv` or `dot_mm` if you know a and b's dimensions

`Goulib.math2.zeros(shape)`

See <https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html>

`Goulib.math2.diag(v)`

Create a two-dimensional array with the flattened input as a diagonal.

Parameters **v** – If v is a 2-D array, return a copy of its diagonal. If v is a 1-D array, return a 2-D array with v on the diagonal

See <https://docs.scipy.org/doc/numpy/reference/generated/numpy.diag.html#numpy.diag>

`Goulib.math2.identity(n)`

`Goulib.math2.eye (n)`

`Goulib.math2.transpose (m)`

Returns matrix m transposed

`Goulib.math2.maximum (m)`

Compare N arrays and returns a new array containing the element-wise maxima

Parameters *m* – list of arrays (matrix)

Returns list of maximal values found in each column of m

See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.maximum.html>

`Goulib.math2.minimum (m)`

Compare N arrays and returns a new array containing the element-wise minima

Parameters *m* – list of arrays (matrix)

Returns list of minimal values found in each column of m

See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.minimum.html>

`Goulib.math2.vecadd (a, b, fillvalue=0)`

addition of vectors of unequal lengths

`Goulib.math2.vecsub (a, b, fillvalue=0)`

subtraction of vectors of unequal lengths

`Goulib.math2.vecneg (a)`

unary negation

`Goulib.math2.vecmul (a, b)`

product of vectors of unequal lengths

`Goulib.math2.vecdiv (a, b)`

quotient of vectors of unequal lengths

`Goulib.math2.veccompare (a, b)`

compare values in 2 lists. returns triple number of pairs where [a<b, a==b, a>b]

`Goulib.math2.sat (x, low=0, high=None)`

saturates x between low and high

`Goulib.math2.norm_2 (v)`

Returns “normal” euclidian norm of vector v

`Goulib.math2.norm_1 (v)`

Returns “manhattan” norm of vector v

`Goulib.math2.norm_inf (v)`

Returns infinite norm of vector v

`Goulib.math2.norm (v, order=2)`

See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.norm.html>

`Goulib.math2.dist (a, b, norm=<function norm_2>)`

`Goulib.math2.vecunit (v, norm=<function norm_2>)`

Returns vector normalized

`Goulib.math2.hamming(s1, s2)`

Calculate the Hamming distance between two iterables

`Goulib.math2.sets_dist(a, b)`

See <http://stackoverflow.com/questions/11316539/calculating-the-distance-between-two-unordered-sets>

`Goulib.math2.sets_levenshtein(a, b)`

levenshtein distance on sets

See http://en.wikipedia.org/wiki/Levenshtein_distance

`Goulib.math2.levenshtein(seq1, seq2)`

levenshtein distance

Returns distance between 2 iterables

See http://en.wikipedia.org/wiki/Levenshtein_distance

`Goulib.math2.recurrence(coefficients, values, cst=0, max=None, mod=0)`

general generator for recurrences

Parameters

- **values** – list of initial values
- **coefficients** – list of factors defining the recurrence

`Goulib.math2.fibonacci_gen(max=None, mod=0)`

Generate fibonacci serie

`Goulib.math2.fibonacci(n, mod=0)`

fibonacci series n-th element

Parameters

- **n** – int can be extremely high, like $1e19$!
- **mod** – int optional modulo

`Goulib.math2.is_fibonacci(n)`

returns True if n is in Fibonacci series

`Goulib.math2.pisano_cycle(mod)`

`Goulib.math2.pisano_period(mod)`

`Goulib.math2.pascal_gen()`

Pascal's triangle read by rows: $C(n,k) = \text{binomial}(n,k) = n!/(k!*(n-k)!)$, $0 \leq k \leq n$.

<https://oeis.org/A007318>

`Goulib.math2.catalan(n)`

Catalan numbers: $C(n) = \text{binomial}(2n,n)/(n+1) = (2n)!/(n!(n+1)!)$.

`Goulib.math2.catalan_gen()`

Generate Catalan numbers: $C(n) = \text{binomial}(2n,n)/(n+1) = (2n)!/(n!(n+1)!)$. Also called Segner numbers.

`Goulib.math2.is_pythagorean_triple(a, b, c)`

`Goulib.math2.primitive_triples()`

generates primitive Pythagorean triplets $x < y < z$

sorted by hypotenuse z , then longest side y through Berggren's matrices and breadth first traversal of ternary tree :see: https://en.wikipedia.org/wiki/Tree_of_primitive_Pythagorean_triples

`Goulib.math2.triples()`

generates all Pythagorean triplets triplets $x < y < z$ sorted by hypotenuse z , then longest side y

`Goulib.math2.divisors(n)`

Returns all divisors of n : `divisors(12)` -> 1,2,3,6,12

including 1 and n , except for 1 which returns a single 1 to avoid messing with sum of divisors...

`Goulib.math2.proper_divisors(n)`

Returns all divisors of n except n itself.

`Goulib.math2.sieve(n, oneisprime=False)`

prime numbers from 2 to a prime $< n$ Very fast ($n < 10,000,000$) in 0.4 sec.

Example: `>>>prime_sieve(25)` [2, 3, 5, 7, 11, 13, 17, 19, 23]

Algorithm & Python source: Robert William Hanks <http://stackoverflow.com/questions/17773352/python-sieve-prime-numbers>

`Goulib.math2.primes(n)`

memoized list of n first primes

Warning do not call with large n , use `prime_gen` instead

`Goulib.math2.is_prime(n, oneisprime=False, precision_for_huge_n=16)`

primality test. Uses Miller-Rabin for large n

Parameters

- **n** – int number to test
- **oneisprime** – bool True if 1 should be considered prime (it was, a long time ago)
- **precision_for_huge_n** – int number of primes to use in Miller

Returns True if n is a prime number

`Goulib.math2.primes_gen(start=2, stop=None)`

generate prime numbers from 'start'

`Goulib.math2.random_prime(bits)`

returns a random number of the specified bit length

`Goulib.math2.euclid_gen()`

generates Euclid numbers: $1 + \text{product of the first } n \text{ primes}$

`Goulib.math2.prime_factors(num, start=2)`

generates all prime factors (ordered) of num

`Goulib.math2.prime_divisors(num, start=2)`

generates unique prime divisors (ordered) of num

`Goulib.math2.is_multiple(n, factors)`

return True if n has ONLY factors as prime factors

`Goulib.math2.factorize(n)`

find the prime factors of n along with their frequencies. Example:

```
>>> factor(786456)
[(2,3), (3,3), (11,1), (331,1)]
```

`Goulib.math2.factors(n)`

`Goulib.math2.number_of_divisors(n)`

`Goulib.math2.omega(n)`

Number of distinct primes dividing n

`Goulib.math2.bigomega(n)`

Number of prime divisors of n counted with multiplicity

`Goulib.math2.moebius(n)`

Möbius (or Moebius) function $\mu(n)$. $\mu(1) = 1$; $\mu(n) = (-1)^k$ if n is the product of k different primes; otherwise $\mu(n) = 0$.

`Goulib.math2.euler_phi(n)`

Euler totient function

See <http://stackoverflow.com/questions/1019040/how-many-numbers-below-n-are-coprimes-to-n>

`Goulib.math2.totient(n)`

Euler totient function

See <http://stackoverflow.com/questions/1019040/how-many-numbers-below-n-are-coprimes-to-n>

`Goulib.math2.prime_ktuple(constellation)`

generates tuples of primes with specified differences

Parameters `constellation` – iterable of int differences between primes to return

Note negative int means the difference must NOT be prime

See https://en.wikipedia.org/wiki/Prime_k-tuple

(0, 2) twin primes (0, 4) cousin primes (0, 6) sexy primes (0, 2, 6), (0, 4, 6) prime triplets (0, 6, 12, -18) sexy prime triplets (0, 2, 6, 8) prime quadruplets (0, 6, 12, 18) sexy prime quadruplets (0, 2, 6, 8, 12), (0, 4, 6, 10, 12) quintuplet primes (0, 4, 6, 10, 12, 16) sextuplet primes

`Goulib.math2.twin_primes()`

`Goulib.math2.cousin_primes()`

`Goulib.math2.sexy_primes()`

`Goulib.math2.sexy_prime_triplets()`

`Goulib.math2.sexy_prime_quadruplets()`

`Goulib.math2.lucas_lehmer(p)`

Lucas Lehmer primality test for Mersenne exponent p

Parameters `p` – int

Returns True if $2^p - 1$ is prime

`Goulib.math2.digits_gen(num, base=10)`

generates int digits of num in base BACKWARDS

`Goulib.math2.digits(num, base=10, rev=False)`

Returns list of digits of num expressed in base, optionally reversed

`Goulib.math2.digsum(num, f=None, base=10)`

sum of digits

Parameters

- `num` – number
- `f` – int power or function applied to each digit
- `base` – optional base

Returns sum of f(digits) of num

digsum(num) -> sum of digits digsum(num,base=2) -> number of 1 bits in binary representation of num digsum(num,2) -> sum of the squares of digits digsum(num,f=lambda x:x**x) -> sum of the digits elevated to their own power

Goulib.math2.**integer_exponent** (a, b=10)

Returns int highest power of b that divides a.

See <https://reference.wolfram.com/language/ref/IntegerExponent.html>

Goulib.math2.**trailing_zeros** (a, b=10)

Returns int highest power of b that divides a.

See <https://reference.wolfram.com/language/ref/IntegerExponent.html>

Goulib.math2.**power_tower** (v)

Returns v[0]**v[1]**v[2] ...

See <http://ajcr.net/Python-power-tower/>

Goulib.math2.**carries** (a, b, base=10, pos=0)

Returns int number of carries required to add a+b in base

Goulib.math2.**powertrain** (n)

Returns v[0]**v[1]*v[2]**v[3] ...**(v[-1] or 0)

Author # Chai Wah Wu, Jun 16 2017

See <http://oeis.org/A133500>

Goulib.math2.**str_base** (num, base=10, numerals='0123456789abcdefghijklmnopqrstuvwxyz')

Returns string representation of num in base

Parameters

- **num** – int number (decimal)
- **base** – int base, 10 by default
- **numerals** – string with all chars representing numbers in base base. chars after the base-th are ignored

Goulib.math2.**num_from_digits** (digits, base=10)

Parameters

- **digits** – string or list of digits representing a number in given base
- **base** – int base, 10 by default

Returns int number

Goulib.math2.**reverse** (i)

Goulib.math2.**is_palindromic** (num, base=10)

Check if 'num' in base 'base' is a palindrome, that's it, if it can be read equally from left to right and right to left.

Goulib.math2.**is_permutation** (num1, num2, base=10)

Check if 'num1' and 'num2' have the same digits in base

Goulib.math2.**is_pandigital** (num, base=10)

Returns True if num contains all digits in specified base

Goulib.math2.**bouncy** (*n*)

Goulib.math2.**repunit_gen** (*digit=1*)
generate repunits

Goulib.math2.**repunit** (*n, digit=1*)

Returns nth repunit

Goulib.math2.**rational_form** (*numerator, denominator*)
information about the decimal representation of a rational number.

Returns 5 integer : integer, decimal, shift, repeat, cycle

- shift is the len of decimal with leading zeroes if any
- cycle is the len of repeat with leading zeroes if any

Goulib.math2.**rational_str** (*n, d*)

Goulib.math2.**rational_cycle** (*num, den*)
periodic part of the decimal expansion of num/den. Any initial 0's are placed at end of cycle.

See <https://oeis.org/A036275>

Goulib.math2.**tetrahedral** (*n*)

Returns int n-th tetrahedral number

See https://en.wikipedia.org/wiki/Tetrahedral_number

Goulib.math2.**sum_of_squares** (*n*)

Returns $1^2 + 2^2 + 3^2 + \dots + n^2$

See https://en.wikipedia.org/wiki/Square_pyramidal_number

Goulib.math2.**pyramidal** (*n*)

Returns $1^2 + 2^2 + 3^2 + \dots + n^2$

See https://en.wikipedia.org/wiki/Square_pyramidal_number

Goulib.math2.**sum_of_cubes** (*n*)

Returns $1^3 + 2^3 + 3^3 + \dots + n^3$

See https://en.wikipedia.org/wiki/Squared_triangular_number

Goulib.math2.**bernouilli_gen** (*init=1*)
generator of Bernouilli numbers

Parameters *init* – int -1 or +1.

- -1 for “first Bernoulli numbers” with $B_1 = -1/2$
- +1 for “second Bernoulli numbers” with $B_1 = +1/2$

https://en.wikipedia.org/wiki/Bernoulli_number
[_Optimised_task_algorithm](#)

https://rosettacode.org/wiki/Bernoulli_numbers#Python:

Goulib.math2.**bernouilli** (*n, init=1*)

Goulib.math2.**faulhaber** (*n, p*)
sum of the p-th powers of the first n positive integers

Returns $1^p + 2^p + 3^p + \dots + n^p$

See https://en.wikipedia.org/wiki/Faulhaber%27s_formula

`Goulib.math2.is_happy(n)`

`Goulib.math2.lychrel_seq(n)`

`Goulib.math2.lychrel_count(n, limit=96)`

number of lychrel iterations before n becomes palindromic

Parameters

- **n** – int number to test
- **limit** – int max number of loops. default 96 corresponds to the known most retarded non lychrel number

Warning there are palindrom lychrel numbers such as 4994

`Goulib.math2.is_lychrel(n, limit=96)`

Warning there are palindrom lychrel numbers such as 4994

`Goulib.math2.polygonal(s, n)`

`Goulib.math2.triangle(n)`

Returns nth triangle number, defined as the sum of [1,n] values.

See http://en.wikipedia.org/wiki/Triangular_number

`Goulib.math2.triangular(n)`

Returns nth triangle number, defined as the sum of [1,n] values.

See http://en.wikipedia.org/wiki/Triangular_number

`Goulib.math2.is_triangle(x)`

Returns True if x is a triangle number

`Goulib.math2.is_triangular(x)`

Returns True if x is a triangle number

`Goulib.math2.square(n)`

`Goulib.math2.pentagonal(n)`

Returns nth pentagonal number

See https://en.wikipedia.org/wiki/Pentagonal_number

`Goulib.math2.is_pentagonal(n)`

Returns True if x is a pentagonal number

`Goulib.math2.hexagonal(n)`

Returns nth hexagonal number

See https://en.wikipedia.org/wiki/Hexagonal_number

`Goulib.math2.is_hexagonal(n)`

`Goulib.math2.heptagonal(n)`

`Goulib.math2.is_heptagonal(n)`

`Goulib.math2.octagonal(n)`

`Goulib.math2.is_octagonal(n)`

`Goulib.math2.partition(n)`

The partition function $p(n)$

gives the number of partitions of a nonnegative integer n into positive integers. (There is one partition of zero into positive integers, i.e. the empty partition, since the empty sum is defined as 0.)

See http://oeis.org/wiki/Partition_function

`Goulib.math2.get_cardinal_name(num)`

Get cardinal name for number (0 to 1 million)

`Goulib.math2.abundance(n)`

`Goulib.math2.is_perfect(n)`

Returns -1 if n is deficient, 0 if perfect, 1 if abundant

See https://en.wikipedia.org/wiki/Perfect_number,

https://en.wikipedia.org/wiki/Abundant_number, https://en.wikipedia.org/wiki/Deficient_number

`Goulib.math2.number_of_digits(num, base=10)`

Return number of digits of num (expressed in base 'base')

`Goulib.math2.chakravala(n)`

solves $x^2 - n*y^2 = 1$ for x, y integers

https://en.wikipedia.org/wiki/Pell%27s_equation https://en.wikipedia.org/wiki/Chakravala_method

`Goulib.math2.factorial_gen()`

Generator of factorial

`Goulib.math2.binomial(n, k)`

<https://en.wikipedia.org/wiki/binomial>

`Goulib.math2.ncombinations(n, k)`

<https://en.wikipedia.org/wiki/binomial>

`Goulib.math2.binomial_exponent(n, k, p)`

Returns int largest power of p that divides $\text{binomial}(n, k)$

`Goulib.math2.log_factorial(n)`

Returns float approximation of $\ln(n!)$ by Ramanujan formula

`Goulib.math2.log_binomial(n, k)`

Returns float approximation of $\ln(\text{binomial}(n, k))$

`Goulib.math2.ilog(a, b, upper_bound=False)`

discrete logarithm x such that $b^x = a$

Parameters

- **a, b** – integer
- **upper_bound** – bool. if True, returns smallest x such that $b^x \geq a$

Returns x integer such that $b^x = a$, or `upper_bound`, or `None`

https://en.wikipedia.org/wiki/Discrete_logarithm

`Goulib.math2.angle(u, v, unit=True)`

Parameters

- **u, v** – iterable vectors
- **unit** – bool True if vectors are unit vectors. False increases computations

Returns float angle n radians between u and v unit vectors i

`Goulib.math2.sin_over_x(x)`
numerically safe $\sin(x)/x$

`Goulib.math2.slerp(u, v, t)`
spherical linear interpolation

Parameters

- **u, v** – 3D unit vectors
- **t** – float in $[0,1]$ interval

Returns vector interpolated between u and v

`Goulib.math2.proportional(nseats, votes)`
assign n seats proportionally to votes using the https://en.wikipedia.org/wiki/Hagenbach-Bischoff_quota method

Parameters

- **nseats** – int number of seats to assign
- **votes** – iterable of int or float weighting each party

Result list of ints seats allocated to each party

`Goulib.math2.triangular_repartition(x, n)`
divide 1 into n fractions such that:

- their sum is 1
- they follow a triangular linear repartition (sorry, no better name for now) where $x/1$ is the maximum

`Goulib.math2.rectangular_repartition(x, n, h)`
divide 1 into n fractions such that:

- their sum is 1
- they follow a repartition along a pulse of height $h < 1$

`Goulib.math2.de_bruijn(k, n)`
De Bruijn sequence for alphabet k and subsequences of length n .
https://en.wikipedia.org/wiki/De_Bruijn_sequence

`Goulib.math2.mod_pow(a, b, m)`

Returns $(a^b) \bmod m$

`Goulib.math2.mod_inv(a, b)`

`Goulib.math2.mod_div(a, b, m)`

Returns x such that $(b \cdot x) \bmod m = a \bmod m$

`Goulib.math2.mod_fact(n, m)`

Returns $n! \bmod m$

`Goulib.math2.chinese_remainder(m, a)`
http://en.wikipedia.org/wiki/Chinese_remainder_theorem

Parameters

- **m** – list of int moduli
- **a** – list of int remainders

Returns smallest int x such that $x \bmod n_i = a_i$

`Goulib.math2.mod_binomial` ($n, k, m, q=None$)
calculates $C(n,k) \bmod m$ for large n,k,m

Parameters

- **n** – int total number of elements
- **k** – int number of elements to pick
- **m** – int modulo (or iterable of (m,p) tuples used internally)
- **q** – optional int power of m for prime m, used internally

`Goulib.math2.baby_step_giant_step` (y, a, n)
solves Discrete Logarithm Problem (DLP) $y = a^{**}x \bmod n$

`Goulib.math2.mod_matmul` ($A, B, mod=0$)

`Goulib.math2.mod_matpow` ($M, power, mod=0$)

`Goulib.math2.matrix_power` ($M, power, mod=0$)

`Goulib.math2.pi_digits_gen` ()
generates pi digits as a sequence of INTEGERS ! using Jeremy Gibbons spigot generator
:see :<http://www.cs.ox.ac.uk/people/jeremy.gibbons/publications/spigot.pdf>

2.15 Goulib.motion module

motion simulation (kinematics)

class `Goulib.motion.PVA` (*funcs*)

Bases: `Goulib.plot.Plot`

represents a function of time returning position, velocity, and acceleration

`__init__` (*funcs*)

`__call__` ($t, t0=0$)

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () $\rightarrow$ list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

```

__getattribute__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

html ( **kwargs )

plot ( **kwargs )
    renders on IPython Notebook (alias to make usage more straightforward)

png ( **kwargs )

render (fmt='svg', **kwargs)

save (filename, **kwargs)

svg ( **kwargs )

class Goulib.motion.Segment (t0, t1, funcs)
    Bases: Goulib.motion.PVA
    a PVA defined between 2 times, null elsewhere

    __init__ (t0, t1, funcs)

    dt ()

    start ()

    startPos ()

    startSpeed ()

```

```
startAcc()  
startJerk()  
startTime()  
end()  
endPos()  
endSpeed()  
endAcc()  
endJerk()  
endTime()  
timeWhenPosBiggerThan (pos, resolution=0.01)  
    search the first time when the position is bigger than pos :params pos: the pos that must at least be reached  
    :params resolution: the time resolution in sec  
  
__call__ (t)  
  
__class__  
    alias of type  
  
__delattr__  
    Implement delattr(self, name).  
  
__dir__ () → list  
    default dir() implementation  
  
__eq__  
    Return self==value.  
  
__format__ ()  
    default object formatter  
  
__ge__  
    Return self>=value.  
  
__getattr__  
    Return getattr(self, name).  
  
__gt__  
    Return self>value.  
  
__hash__  
    Return hash(self).  
  
__le__  
    Return self<=value.  
  
__lt__  
    Return self<value.  
  
__ne__  
    Return self!=value.  
  
__new__ ()  
    Create and return a new object. See help(type) for accurate signature.  
  
__reduce__ ()  
    helper for pickle
```

```

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

html ( **kwargs )

plot ( **kwargs )
    renders on IPython Notebook (alias to make usage more straightforward)

png ( **kwargs )

render (fmt='svg', **kwargs)

save (filename, **kwargs)

svg ( **kwargs )

class Goulib.motion.Segments (segments=[], label='Segments')
    Bases: Goulib.motion.Segment

    can be initialized with a list of segment (that of course can also be a Segments) :param label: a label can be
    given

    __init__ (segments=[], label='Segments')
        can be initialized with a list of segment (that of course can also be a Segments) :param label: a label can
        be given

    __str__ ()

    html ()

    update ()
        yet only calculates t0 and t1

    insert (segment, autoJoin=True)
        insert a segment into Segments :param segment: the segment to add. must be in a range that is not already
        defined or it will rise a value error exception :param autoJoin: if True and the added segment has the same
        starting position as the last segment's end

        and both velocity are 0 then a segment of (pos,v=0,a=0) is automatically added. this help describ-
        ing movements only where there is curently a movement

    add (segments, autoJoin=True)
        add a segment or a list of segment to the segments

    start ()

    end ()

    __call__ (t)

    __class__
        alias of type

```

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`dt` ()

`endAcc` ()

`endJerk` ()

`endPos` ()

`endSpeed` ()

`endTime` ()

`plot` (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

```

png (**kwargs)
render (fmt='svg', **kwargs)
save (filename, **kwargs)
startAcc ()
startJerk ()
startPos ()
startSpeed ()
startTime ()
svg (**kwargs)
timeWhenPosBiggerThan (pos, resolution=0.01)
    search the first time when the position is bigger than pos :params pos: the pos that must at least be reached
    :params resolution: the time resolution in sec

```

class Goulib.motion.**SegmentPoly** (*t0, t1, p*)
 Bases: *Goulib.motion.Segment*

a segment defined by a polynomial position law

```

__init__ (t0, t1, p)
__call__ (t)
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__ () → list
    default dir() implementation
__eq__
    Return self==value.
__format__ ()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__
    Return hash(self).
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.

```

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

dt ()

end ()

endAcc ()

endJerk ()

endPos ()

endSpeed ()

endTime ()

html (**kwargs)

plot (**kwargs)
renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

render (fmt='svg', **kwargs)

save (filename, **kwargs)

start ()

startAcc ()

startJerk ()

startPos ()

startSpeed ()

startTime ()

svg (**kwargs)

timeWhenPosBiggerThan (pos, resolution=0.01)
search the first time when the position is bigger than pos :params pos: the pos that must at least be reached
:params resolution: the time resolution in sec

```
class Goulib.motion.Actuator (stateMachine, vmax, acc, name='', pos=<Quantity(0, 'meter')>,
                             distPerTurn=<Quantity(1, 'millimeter')>, angle=<Quantity(0, 'degree')>, mass=<Quantity(1, 'kilogram')>, friction=<Quantity(0, 'newton')>)
```

Bases: `object`

simulate an actuator. each movements are recorded in a Segments object the goal of this class is to simplify the writing in most common cases

Params stateMachine a stateMachine. the only requirement for the simulation is to have a .time as V(time,'s') and a .displayMove boolean

Params acc the default acceleration of the actuator

Params vmax the default vmax

Params name name of the actuator

Params pos the initial position

Params distPerTurn the distance of the actuator per motor (or reductor) turn

Params angle if 0, the mass is moving horizontally, if 90° vertically: CAREFULL in that case the bigger the position, the higher

Params mass the mass to move (for both inertia and lifting force)

Params friction the friction force TODO: currently no difference between u0 and uynamique

intertia of the pulley should be taken into consideration

WARNING: the maxForce simulation is minimalist: as we know nothing about the reversibility of the gears and where is the friction, friction is always added even if it might be compensated by the mass in the case we go down

```
__init__ (stateMachine, vmax, acc, name='', pos=<Quantity(0, 'meter')>, distPerTurn=<Quantity(1, 'millimeter')>, angle=<Quantity(0, 'degree')>, mass=<Quantity(1, 'kilogram')>, friction=<Quantity(0, 'newton')>)
```

Params stateMachine a stateMachine. the only requirement for the simulation is to have a .time as V(time,'s') and a .displayMove boolean

Params acc the default acceleration of the actuator

Params vmax the default vmax

Params name name of the actuator

Params pos the initial position

Params distPerTurn the distance of the actuator per motor (or reductor) turn

Params angle if 0, the mass is moving horizontally, if 90° vertically: CAREFULL in that case the bigger the position, the higher

Params mass the mass to move (for both inertia and lifting force)

Params friction the friction force TODO: currently no difference between u0 and uynamique

intertia of the pulley should be taken into consideration

WARNING: the maxForce simulation is minimalist: as we know nothing about the reversibility of the gears and where is the friction, friction is always added even if it might be compensated by the mass in the case we go down

move (*newpos*, *relative=False*, *time=None*, *wait=True*, *vmax=None*, *acc=None*)

moves the actuator to *newpos* :params *newpos*: the new absolute position :params *time*: the starting time of the move. by default (None) the state machine time will be used but

one can force the starting point in the past typically to do parallel moves of different actuators

Params *vmax* by default the values given at initialisation, but a value for this move can be given

Params *acc* by default the values given at initialisation, but a value for this move can be given

endTime ()

P (*t*)

maxAbsAcc ()

maxAbsSpeed ()

maxForce ()

maxTork ()

maxRpm ()

displayLast ()

display (*fromTime=None*, *toTime=None*)

varNames ()

returns a list of internal variables. intended to be used in the Table.__init__

varRowUnits ()

varDict ()

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

```

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

class `Goulib.motion.TimeDiagram` (*actuators*, *stateMachines*=[], *fromTime*=None, *toTime*=None)

Bases: `Goulib.plot.Plot`

Params *stateMachines* [(stateMachine,pos,posShift),...]

__init__ (*actuators*, *stateMachines*=[], *fromTime*=None, *toTime*=None)

Params *stateMachines* [(stateMachine,pos,posShift),...]

__repr__ ()

saveAsCsv (*filename*)

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

html (***kwargs*)

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

Goulib.motion.**ramp** (*dp, v0, v1, a*)

Parameters

- **dp** – float delta position or None if unknown
- **v0** – float initial velocity or None if unknown
- **v1** – float final velocity or None if unknown
- **a** – float acceleration

Returns float shortest time to accelerate between constraints

Goulib.motion.**trapeze** (*dp, vmax, a, v0=0, v2=0*)

Parameters

- **dp** – float delta position
- **vmax** – float maximal velocity
- **a** – float acceleration
- **v0** – float initial velocity, 0 by default

- **v2** – float final velocity, 0 by default

Returns tuple of 6 values:

- time at end of acceleration
- position at end of acceleration
- velocity at end of acceleration
- time at begin of deceleration
- position at begin of deceleration
- total time

`Goulib.motion.Segment2ndDegree (t0, t1, start, end=None)`
calculates a constant acceleration Segment between start and end

Parameters

- **t0, t1** – float start,end time. one of both may be None for undefined
- **start** – (position, velocity, acceleration) float tuple. some values may be None for undefined
- **end** – (position, velocity, acceleration) float tuple. some values may be None for undefined

Returns *SegmentPoly*

the function can cope with almost any combination of defined/undefined parameters, among others (see tests):

- `Segment2ndDegree(t0,t1,(p0,v0),p1)` # time interval and start + end positions + initial speed
- `Segment2ndDegree(t0,t1,(p0,v0,a))` # time interval and start with acceleration
- `Segment2ndDegree(t0,t1,None,(p1,v1,a))` # time interval and end pva
- `Segment2ndDegree(t0,None,(p0,v0),(p1,v1))` # start + end positions + velocities
- `Segment2ndDegree(t0,None,(p0,v0,a),(None,v1))` # start pva + end velocity
- `Segment2ndDegree(None,t1,p0,(p1,v1,a))` # end pva + start position

the function also accepts some combinations of overconstraining parameters:

- `Segment2ndDegree(t0,t1,(p0,v0,a),p1)` # time interval, start pva, end position => adjust t1
- `Segment2ndDegree(t0,t1,(p0,v0,a),(None,v1))` # time interval, start pva, v1=max vel => adjust t1

Raises **ValueError** – when not enough parameters are specified to define the Segment univoquely

`Goulib.motion.Segment4thDegree (t0, t1, start, end)`
smooth trajectory from an initial position and initial speed (p0,v0) to a final position and speed (p1,v1) * if t1<=t0, t1 is calculated

`Goulib.motion.SegmentsTrapezoidalSpeed (t0, p0, p3, a, T=0, vmax=inf, v0=0, v3=0)`

Parameters

- **t0** – float start time
- **p0** – float start position
- **p3** – float end position
- **a** – float specified acceleration. if =0, use specified time

- **T** – float specified time. if =0 (default), use specified acceleration
- **vmax** – float max speed. default is infinity (i.e. triangular speed)
- **v0** – initial speed
- **v3** – final speed if $T < 0$ then $v3 = v0 \cdot v1 + \dots +$
// + v3

```
v0 +  
    |||  
t0 t1 t2 t3
```

2.16 Goulib.optim module

various optimization algorithms : knapsack, traveling salesman, simulated annealing, differential evolution

class Goulib.optim.**ObjectiveFunction** (*objective_function*)

Bases: `object`

class to wrap an objective function and keep track of the best solution evaluated

__init__ (*objective_function*)

__call__ (*solution*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

```

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

`Goulib.optim.nelder_mead(f, x_start, step=0.1, no_improve_thr=1e-05, no_improv_break=10, max_iter=0, alpha=1.0, gamma=2.0, rho=-0.5, sigma=0.5)`

Pure Python implementation of the Nelder-Mead algorithm. also called “downhill simplex method” taken from <https://github.com/fchollet/nelder-mead>

Reference: https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method :param f: function to optimize, must return a scalar score

and operate over a numpy array of the same dimensions as `x_start`

Parameters

- **x_start** – (numpy array) initial position
- **step** – (float) look-around radius in initial step
- **no_improv_break** (*no_improv_thr*) – (float,int): break after `no_improv_break` iterations with an improvement lower than `no_improv_thr`
- **max_iter** – (int): always break after this number of iterations. Set it to 0 to loop indefinitely.
- **gamma, rho, sigma** (*alpha*) – (floats): parameters of the algorithm (see Wikipedia page for reference)

class `Goulib.optim.BinDict(capacity, f=<function _Bin.<lambda>>)`

Bases: `Goulib.optim._Bin, dict`

a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin :param f: function `f(x)` returning the capacity used by item `x`. Must return the empty capacity when `f(0)` is called

```

__isub__ (key)

__delitem__ (key)

__iadd__ (key, item)

__setitem__ (key, item)

```

__class__
alias of `type`

__contains__ ()
True if D has a key k, else False.

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__getitem__ ()
`x.__getitem__(y) <==> x[y]`

__gt__
Return `self>value`.

__hash__ = None

__init__ (*capacity*, *f*=<function `_Bin.<lambda>>`>)
a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin :param f: function `f(x)` returning the capacity used by item x. Must return the empty capacity when `f(0)` is called

__iter__
Implement `iter(self)`.

__le__
Return `self<=value`.

__len__
Return `len(self)`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

```

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → size of D in memory, in bytes

__str__
    Return str(self).

clear () → None. Remove all items from D.

copy () → a shallow copy of D

fits (item)

    Returns bool True if item fits in bin without exceeding capacity

fromkeys ()
    Returns a new dict with keys from iterable and values equal to value.

get (k[, d]) → D[k] if k in D, else d. d defaults to None.

items () → a set-like object providing a view on D's items

keys () → a set-like object providing a view on D's keys

pop (k[, d]) → v, remove specified key and return the corresponding value.
    If key is not found, d is returned if given, otherwise KeyError is raised

popitem () → (k, v), remove and return some (key, value) pair as a
    2-tuple; but raise KeyError if D is empty.

setdefault (k[, d]) → D.get(k,d), also set D[k]=d if k not in D

size ()

update ([E], **F) → None. Update D from dict/iterable E and F.
    If E is present and has a .keys() method, then does: for k in E: D[k] = E[k]
    If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v
    In either case, this is followed by: for k in F: D[k] = F[k]

values () → an object providing a view on D's values

class Goulib.optim.BinList (capacity, f=<function _Bin.<lambda>>)
    Bases: Goulib.optim._Bin, list

    a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin
    :param f: function f(x) returning the capacity used by item x. Must return the empty capacity when f(0) is called

    __iadd__ (item)

    append (item)

    insert (i, item)

    extend (more)

    __isub__ (item)

    remove (item)

    pop (i=-1)

    __setitem__ (i, item)
        called when replacing a value in list

    __add__
        Return self+value.

```

__class__
alias of `type`

__contains__
Return key in self.

__delattr__
Implement `delattr(self, name)`.

__delitem__
Delete `self[key]`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__getitem__ ()
`x.__getitem__(y) <==> x[y]`

__gt__
Return `self>value`.

__hash__ = None

__imul__
Implement `self*=value`.

__init__ (*capacity*, *f*=<function `_Bin.<lambda>>`>)
a container with a limited capacity :param *capacity*: int, float, tuple of whatever defines the capacity of the Bin :param *f*: function `f(x)` returning the capacity used by item *x*. Must return the empty capacity when `f(0)` is called

__iter__
Implement `iter(self)`.

__le__
Return `self<=value`.

__len__
Return `len(self)`.

__lt__
Return `self<value`.

__mul__
Return `self*value.n`

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

```

__reduce__()
    helper for pickle

__reduce_ex__()
    helper for pickle

__repr__()

__reversed__()
    L.__reversed__() – return a reverse iterator over the list

__rmul__
    Return self*value.

__setattr__
    Implement setattr(self, name, value).

__sizeof__()
    L.__sizeof__() – size of L in memory, in bytes

__str__
    Return str(self).

clear() → None – remove all items from L

copy() → list – a shallow copy of L

count(value) → integer – return number of occurrences of value

fits(item)

    Returns bool True if item fits in bin without exceeding capacity

index(value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

reverse()
    L.reverse() – reverse IN PLACE

size()

sort(key=None, reverse=False) → None – stable sort *IN PLACE*

Goulib.optim.first_fit_decreasing(items, bins, maxbins=0)
    fit items in bins using the “first fit decreasing” method :param items: iterable of items :param bins: iterable of
    Bin s. Must have at least one Bin :return: list of items that didn’t fit. (bins are filled by side-effect)

Goulib.optim.hillclimb(init_function, move_operator, objective_function, max_evaluations)
    hillclimb until either max_evaluations is reached or we are at a local optima

Goulib.optim.hillclimb_and_restart(init_function, move_operator, objective_function,
    max_evaluations)
    repeatedly hillclimb until max_evaluations is reached

Goulib.optim.P(prev_score, next_score, temperature)

Goulib.optim.kirkpatrick_cooling(start_temp, alpha)

Goulib.optim.anneal(init_function, move_operator, objective_function, max_evaluations, start_temp,
    alpha)

Goulib.optim.reversed_sections(tour)
    generator to return all possible variations where the section between two cities are swapped

Goulib.optim.swapped_cities(tour)
    generator to create all possible variations where two cities have been swapped

```

`Goulib.optim.tour_length` (*points, dist, tour=None*)
generator of point-to-point distances along a tour

`Goulib.optim.tsp` (*points, dist, max_iterations=100, start_temp=None, alpha=None, close=True, rand=True*)

Traveling Salesman Problem @see http://en.wikipedia.org/wiki/Travelling_salesman_problem @param points : iterable containing all points @param dist : function returning the distance between 2 points : def dist(a,b): @param max_iterations :max number of optimization steps @param start_temp, alpha : params for the simulated annealing algorithm. if None, hill climbing is used @param close : computes closed TSP. if False, open TSP starting at points[0] @return iterations,score,best : number of iterations used, minimal length found, best path as list of indexes of points

class `Goulib.optim.DifferentialEvolution` (*evaluator, population_size=50, f=None, cr=0.9, eps=0.01, n_cross=1, max_iter=10000, monitor_cycle=200, out=None, show_progress=False, show_progress_nth_cycle=1, insert_solution_vector=None, dither_constant=0.4*)

Bases: `object`

This is a python implementation of differential evolution taken from http://cci.lbl.gov/cctbx_sources/scitbx/differential_evolution.py

It assumes an evaluator class is passed in that has the following functionality data members:

`n` :: The number of parameters domain :: a list [(low,high)]*n

with approximate upper and lower limits for each parameter

`x` :: a place holder for a final solution

also a function called 'target' is needed. This function should take a parameter vector as input and return a the function to be minimized.

The code below was implemented on the basis of the following sources of information: 1. <http://www.icsi.berkeley.edu/~storn/code.html> 2. http://www.daimi.au.dk/~krink/fec05/articles/JV_ComparativeStudy_CEC04.pdf 3. http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-099Fall2003/A40397B9-E8FB-4B45-A41B-D1F69218901F/0/ses2_storn_price.pdf

The developers of the differential evolution method have this advice: (taken from ref. 1)

If you are going to optimize your own objective function with DE, you may try the following classical settings for the input file first: Choose method e.g. DE/rand/1/bin, set the number of parents NP to 10 times the number of parameters, select weighting factor F=0.8, and crossover constant CR=0.9. It has been found recently that selecting F from the interval [0.5, 1.0] randomly for each generation or for each difference vector, a technique called dither, improves convergence behaviour significantly, especially for noisy objective functions. It has also been found that setting CR to a low value, e.g. CR=0.2 helps optimizing separable functions since it fosters the search along the coordinate axes. On the contrary this choice is not effective if parameter dependence is encountered, something which is frequently occurring in real-world optimization problems rather than artificial test functions. So for parameter dependence the choice of CR=0.9 is more appropriate. Another interesting empirical finding is that raising NP above, say, 40 does not substantially improve the convergence, independent of the number of parameters. It is worthwhile to experiment with these suggestions. Make sure that you initialize your parameter vectors by exploiting their full numerical range, i.e. if a parameter is allowed to exhibit values in the range [-100, 100] it's a good idea to pick the initial values from this range instead of unnecessarily restricting diversity.

Keep in mind that different problems often require different settings for NP, F and CR (have a look into the different papers to get a feeling for the settings). If you still get misconvergence you might want to try a different method. We mostly use DE/rand/1/... or DE/best/1/... . The crossover method is not so important although Ken Price claims that binomial is never worse than exponential. In case of misconvergence also check your choice of objective function. There might be a better one to describe your problem. Any knowledge that

you have about the problem should be worked into the objective function. A good objective function can make all the difference.

Note: NP is called population size in the routine below.) Note: [0.5,1.0] dither is the default behavior unless f is set to a value other than None.

```
__init__ (evaluator, population_size=50, f=None, cr=0.9, eps=0.01, n_cross=1, max_iter=10000,
          monitor_cycle=200, out=None, show_progress=False, show_progress_nth_cycle=1, insert_solution_vector=None, dither_constant=0.4)
```

```
optimize ()
```

```
make_random_population ()
```

```
score_population ()
```

```
__class__  
alias of type
```

```
__delattr__  
Implement delattr(self, name).
```

```
__dir__ () → list  
default dir() implementation
```

```
__eq__  
Return self==value.
```

```
__format__ ()  
default object formatter
```

```
__ge__  
Return self>=value.
```

```
__getattr__  
Return getattr(self, name).
```

```
__gt__  
Return self>value.
```

```
__hash__  
Return hash(self).
```

```
__le__  
Return self<=value.
```

```
__lt__  
Return self<value.
```

```
__ne__  
Return self!=value.
```

```
__new__ ()  
Create and return a new object. See help(type) for accurate signature.
```

```
__reduce__ ()  
helper for pickle
```

```
__reduce_ex__ ()  
helper for pickle
```

```
__repr__  
Return repr(self).
```

```
__setattr__  
    Implement setattr(self, name, value).  
  
__sizeof__ () → int  
    size of object in memory, in bytes  
  
__str__  
    Return str(self).  
  
evolve ()
```

2.17 Goulib.piecewise module

piecewise-defined functions

```
class Goulib.piecewise.Piecewise (init=[], default=0, start=-inf)  
    Bases: Goulib.expr.Expr  
  
    piecewise function defined by a sorted list of (startx, Expr)  
  
    __init__ (init=[], default=0, start=-inf)  
  
    __call__ (x)  
        returns value of Expr at point x  
  
    index (x, v=None)  
        finds an existing point or insert one and returns its index  
  
    __len__ ()  
  
    __iter__ ()  
        iterators through discontinuities. take the opportunity to delete redundant tuples  
  
    append (item)  
        appends a (x,y) item. In fact inserts it at correct position and returns the corresponding index  
  
    extend (iterable)  
        appends an iterable of (x,y) values  
  
    __getitem__ (i)  
  
    __str__ ()  
  
    iapply (f, right)  
        apply function to self  
  
    apply (f, right=None)  
        apply function to copy of self  
  
    applx (f)  
        apply a function to each x value  
  
    __lshift__ (dx)  
  
    __rshift__ (dx)  
  
    __add__ (right)  
  
    __and__ (right)  
  
    __class__  
        alias of type
```

```

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__div__ (right)

__eq__ (other)

__float__ ()

__format__ ()
    default object formatter

__ge__ (other)

__getattr__
    Return getattr(self, name).

__gt__ (other)

__hash__ = None

__invert__ ()

__le__ (other)

__lt__ (other)

__mul__ (right)

__ne__ (other)

__neg__ ()

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__or__ (right)

__pow__ (right)

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__ ()

__rmul__ (right)

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__sub__ (right)

__truediv__ (right)

__xor__ (right)

complexity ()
    measures the complexity of Expr :return: int, sum of the precedence of used ops

```

html (***kwargs*)

isNum

isconstant

Returns True if Expr evaluates to a constant number or bool

latex ()

Returns string LaTeX formula

plot (***kwargs*)

 renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

points (*min=0, xmax=None, eps=0*)

 @return x and y for a line plot

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

2.18 Goulib.plot module

plotable rich object display on IPython/Jupyter notebooks

class Goulib.plot.**Plot**

 Bases: `object`

 base class for plotable rich object display on IPython notebooks inspired from http://nbviewer.ipython.org/github/ipython/ipython/blob/3607712653c66d63e0d7f13f073bde8c0f209ba8/docs/examples/notebooks/display_protocol.ipynb

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

html (***kwargs*)

svg (***kwargs*)

png (***kwargs*)

plot (***kwargs*)

 renders on IPython Notebook (alias to make usage more straightforward)

__class__

 alias of `type`

__delattr__

 Implement `delattr(self, name)`.

__dir__ () → list

 default `dir()` implementation

__eq__

 Return `self==value`.

__format__ ()

 default object formatter

```

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__init__
    Initialize self. See help(type(self)) for accurate signature.

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

Goulib.plot.**render** (*plotables*, *fmt*='svg', ***kwargs*)
 renders several Plot objects

Goulib.plot.**png** (*plotables*, ***kwargs*)

Goulib.plot.**svg** (*plotables*, ***kwargs*)

Goulib.plot.**plot** (*plotables*, ***kwargs*)

Goulib.plot.**save** (*plotables*, *filename*, ***kwargs*)

2.19 Goulib.polynomial module

simple manipulation of polynomials (without SimPy) see <http://docs.sympy.org/dev/modules/polys/reference.html> if you need more ...

class Goulib.polynomial.**Polynomial** (*val*)

Bases: *Goulib.expr.Expr*

Param *val* can be:

- an iterable of the factors in ascending powers order : Polynomial([1,2,3]) holds $3*x^2+2*x+1$
- a string of the form “ $ax^n + b*x^m + \dots + c x + d$ ” where a,b,c,d, are floats and n,m ... are integers the ‘x’ variable name is fixed, and the spaces and ‘*’ chars are optional. terms can be in any order, and even “overlap” : Polynomial(‘3x+x^2-x’) holds x^2+2*x

__init__ (*val*)

Param *val* can be:

- an iterable of the factors in ascending powers order : Polynomial([1,2,3]) holds $3*x^2+2*x+1$
- a string of the form “ $ax^n + b*x^m + \dots + c x + d$ ” where a,b,c,d, are floats and n,m ... are integers the ‘x’ variable name is fixed, and the spaces and ‘*’ chars are optional. terms can be in any order, and even “overlap” : Polynomial(‘3x+x^2-x’) holds x^2+2*x

__lt__ (*other*)

__eq__ (*other*)

__add__ (*other*)

__radd__ (*other*)

__sub__ (*other*)

__rsub__ (*other*)

__mul__ (*other*)

__rmul__ (*other*)

__neg__ ()

__pow__ (*e*)

integral ()

derivative ()

__and__ (*right*)

__call__ (*x=None, **kwargs*)
evaluate the Expr at x OR compose self(x())

__class__
alias of *type*

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__div__ (*right*)

__float__ ()

__format__ ()
default object formatter

```

__ge__(other)
__getattr__
    Return getattr(self, name).
__gt__(other)
__hash__ = None
__invert__()
__le__(other)
__lshift__(dx)
__ne__(other)
__new__()
    Create and return a new object. See help(type) for accurate signature.
__or__(right)
__reduce__()
    helper for pickle
__reduce_ex__()
    helper for pickle
__repr__()
__rshift__(dx)
__setattr__
    Implement setattr(self, name, value).
__sizeof__() → int
    size of object in memory, in bytes
__str__()
__truediv__(right)
__xor__(right)
applx (f, var='x')
    function composition f o self = self(f(x))
apply (f, right=None)
    function composition self o f = f(self(x))
complexity ()
    measures the complexity of Expr :return: int, sum of the precedence of used ops
html (**kwargs)
isNum
isconstant
    Returns True if Expr evaluates to a constant number or bool
latex ()
    Returns string LaTeX formula
plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)

```

png (***kwargs*)

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

Goulib.polynomial.**plist** (*term*)

Force term to have the form of a polynomial list

Goulib.polynomial.**peval** (*plist, x, x2=None*)

Eval the plist at value x. If two values are given, the difference between the second and the first is returned. This latter feature is included for the purpose of evaluating definite integrals.

Goulib.polynomial.**integral** (*plist*)

Return a new plist corresponding to the integral of the input plist. This function uses zero as the constant term, which is okay when evaluating a definite integral, for example, but is otherwise ambiguous.

The math forces the coefficients to be turned into floats. Consider importing `__future__` division to simplify this.

Goulib.polynomial.**derivative** (*plist*)

Return a new plist corresponding to the derivative of the input plist.

Goulib.polynomial.**add** (*p1, p2*)

Return a new plist corresponding to the sum of the two input plists.

Goulib.polynomial.**sub** (*p1, p2*)

Goulib.polynomial.**mult_const** (*p, c*)

Return a new plist corresponding to the input plist multiplied by a const

Goulib.polynomial.**multiply** (*p1, p2*)

Return a new plist corresponding to the product of the two input plists

Goulib.polynomial.**mult_one** (*p, c, i*)

Return a new plist corresponding to the product of the input plist p with the single term $c \cdot x^i$

Goulib.polynomial.**power** (*p, e*)

Return a new plist corresponding to the e-th power of the input plist p

Goulib.polynomial.**parse_string** (*s*)

Do very, very primitive parsing of a string into a plist. 'x' is the only term considered for the polynomial, and this routine can only handle terms of the form: $7x^2 + 6x - 5$ and will choke on seemingly simple forms such as $x^2 \cdot 7 - 1$ or $x^{**2} - 1$

Goulib.polynomial.**tostring** (*p, **kwargs*)

Convert a plist into a string. This looks overly complex at first, but most of the complexity is caused by special cases.

2.20 Goulib.statemachine module

state machines with graph representation

class Goulib.statemachine.**StateDiagram** (*name=None, comment=None, filename=None, directory=None, format=None, engine=None, encoding=None, graph_attr=None, node_attr=None, edge_attr=None, body=None, strict=False*)

Bases: graphviz.dot.Digraph

helper to write State Diagrams graph in iPython notebook This library uses Graphviz that has to be installed separately (<http://www.graphviz.org/>)

state (*number, descr, actions, transitions*)

Parameters

- **number** – the state number (int)
- **descr** – a string describing the state
- **actions** – a string describing the various actions. use
 to make a new line
- **transitions** – a array of tuple (<new_state>,"condition")

Goulib.statemachine.noPrint (*args)

class Goulib.statemachine.Simulation

Bases: object

all simulation should derive from this class that has some helper

__init__ ()

setOutput (h1=<function noPrint>, h2=<function noPrint>, h3=<function noPrint>, h=<function noPrint>, hinfo=<function noPrint>, hsuccess=<function noPrint>, hwarning=<function noPrint>, herror=<function noPrint>, displayState=<function noPrint>, displayObj=<function noPrint>, displayPlot=<function noPrint>)

class Goulib.statemachine.EventLog

Bases: object

log (stateMachine)

class Goulib.statemachine.StateChangeLog (newState)

Bases: Goulib.statemachine.EventLog

__init__ (newState)

class Goulib.statemachine.WaitLog (untilTime, waitForWhat)

Bases: Goulib.statemachine.EventLog

__init__ (untilTime, waitForWhat)

class Goulib.statemachine.TooLateLog (pastTime, missedWhat)

Bases: Goulib.statemachine.EventLog

__init__ (pastTime, missedWhat)

class Goulib.statemachine.TimeMarker (name)

Bases: object

__init__ (name)

set (time)

__call__ ()

__repr__ ()

class Goulib.statemachine.StateMachine (simulation=None, name=None, background_color='#F5ECCE')

Bases: object

__init__ (simulation=None, name=None, background_color='#F5ECCE')

reset ()

where all actuators should be declared and other variables

__reset__ ()

parseDoc (*state*, *f*)

__call__ (*time*)
find the state at time. time must be in seconds

displayGraph ()

checkOnTimeAndWait (*time*, *what*)
checks that the self.time < time if this is not the case an error will be logged with the message

wait (*time*, *cause*=*'unknown cause'*)

run (*start*=0, *stops*=[], *startTime*=None, *maxSteps*=100000, *maxTime*=<Quantity(1000, 'second')>)
runs the behavioral simulation :params start: is the starting state of the simulation :params stops: a list of states that will stop the simulation (after having simulated this last state) :params startTime: a time to start this run if None takes self.time :params maxState: is the number of states being evaluated before the end of simulation :params maxTime: is the virtual time at which the simulation ends_in_comment_or_string :params displayStates: at every new state, display the state in Notebook as well as the time when entered :params displayMove: if True, every actuator.move displays the graph of the move

returns the time when run finishes

hinfo (*args)

hsuccess (*args)

hwarning (*args)

herror (*args)

lastExitTime (*state*)

display (*fromTime*=None, *toTime*=None)

2.21 Goulib.stats module

very basic statistics functions

Goulib.stats.mean_var (*data*)
mean and variance by stable algorithm :param :return: float (mean, variance) of data uses a stable algo by Knuth

Goulib.stats.mean (*data*)

Returns mean of data

Goulib.stats.avg (*data*)

Returns mean of data

Goulib.stats.variance (*data*)

Returns variance of data, faster (?) if mean is already available

Goulib.stats.var (*data*)

Returns variance of data, faster (?) if mean is already available

Goulib.stats.stddev (*data*)

Returns standard deviation of data

Goulib.stats.confidence_interval (*data*, *conf*=0.95)

Returns (low,high) bounds of 95% confidence interval of data

`Goulib.stats.median(data, is_sorted=False)`

Returns median of data

`Goulib.stats.mode(data, is_sorted=False)`

Returns mode (most frequent value) of data

`Goulib.stats.kurtosis(data)`

`Goulib.stats.covariance(data1, data2)`

`Goulib.stats.stats(l)`

Returns min,max,sum,sum2,avg,var of a list

class `Goulib.stats.Stats(data=[], mean=None, var=None)`

Bases: `object`

an object that computes mean, variance and modes of data that is appended to it as in a list (but actual values are not stored)

`__init__(data=[], mean=None, var=None)`

`__repr__()`

`append(x)`

add data x to Stats

`extend(data)`

`remove(data)`

remove data from Stats :param data: value or iterable of values

`sum`

`sum1`

`sum2`

`mean`

`avg`

`average`

`mu`

`variance`

`var`

`stddev`

`sigma`

`__add__(other)`

`__sub__(other)`

`__mul__(other)`

`__neg__()`

`__pow__(n)`

`covariance(other)`

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

class `Goulib.stats.Discrete` (*data*)

Bases: `Goulib.stats.Stats`

discrete probability density function

Parameters **data** – can be:

- list of equiprobable values (uniform distribution)
- dict of x:p values:probability pairs

```

__init__(data)
    Parameters data – can be:
        • list of equiprobable values (uniform distribution)
        • dict of x:p values:probability pairs

__call__(x)

__add__(other)

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__() → list
    default dir() implementation

__eq__
    Return self==value.

__format__()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__mul__(other)

__ne__
    Return self!=value.

__neg__()

__new__()
    Create and return a new object. See help(type) for accurate signature.

__pow__(n)

__reduce__()
    helper for pickle

__reduce_ex__()
    helper for pickle

__repr__()

```

```
__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

__sub__ (other)

append (x)
    add data x to Stats

average

avg

covariance (other)

extend (data)

mean

mu

remove (data)
    remove data from Stats :param data: value or iterable of values

sigma

stddev

sum

sum1

sum2

var

variance

class Goulib.stats.PDF (pdf, data=[])
```

Bases: *Goulib.expr.Expr*, *Goulib.stats.Stats*

probability density function

```
__init__ (pdf, data=[])
```

```
__call__ (x=None, **kwargs)
```

```
__add__ (right)
```

```
__and__ (right)
```

```
__class__
    alias of type
```

```
__delattr__
    Implement delattr(self, name).
```

```
__dir__ () → list
    default dir() implementation
```

```
__div__ (right)
```

```
__eq__ (other)
```

```

__float__()
__format__()
    default object formatter
__ge__(other)
__getattr__
    Return getattr(self, name).
__gt__(other)
__hash__ = None
__invert__()
__le__(other)
__lshift__(dx)
__lt__(other)
__mul__(right)
__ne__(other)
__neg__()
__new__()
    Create and return a new object. See help(type) for accurate signature.
__or__(right)
__pow__(right)
__reduce__()
    helper for pickle
__reduce_ex__()
    helper for pickle
__repr__()
__rmul__(right)
__rshift__(dx)
__setattr__
    Implement setattr(self, name, value).
__sizeof__() → int
    size of object in memory, in bytes
__str__()
__sub__(right)
__truediv__(right)
__xor__(right)
append(x)
    add data x to Stats
applx(f, var='x')
    function composition f o self = self(f(x))

```

apply (*f*, *right=None*)
function composition self o f = f(self(x))

average

avg

complexity ()
measures the complexity of Expr :return: int, sum of the precedence of used ops

covariance (*other*)

extend (*data*)

html (***kwargs*)

isNum

isconstant
Returns True if Expr evaluates to a constant number or bool

latex ()
Returns string LaTeX formula

mean

mu

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

remove (*data*)
remove data from Stats :param data: value or iterable of values

render (*fmt='svg'*, ***kwargs*)

save (*filename*, ***kwargs*)

sigma

stddev

sum

sum1

sum2

svg (***kwargs*)

var

variance

`Goulib.stats.normal_pdf(x, mu, sigma)`
Return the probability density function at x

class `Goulib.stats.Normal` (*data=[], mean=0, var=1*)
Bases: `Goulib.stats.PDF`

represents a normal distributed variable the base class (list) optionally contains data
if data is specified, it it used to fit a normal law

```

__init__(data=[], mean=0, var=1)
    if data is specified, it is used to fit a normal law

sigma

__str__()

latex()

linear(a, b=0)
    Returns a*self+b

__mul__(a)

__div__(a)

__truediv__(a)

__add__(other)

__radd__(other)

__neg__()

__sub__(other)

__rsub__(other)

covariance(other)

cov(other)

pearson(other)

correlation(other)

corr(other)

__and__(right)

__call__(x=None, **kwargs)

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__() → list
    default dir() implementation

__eq__(other)

__float__()

__format__()
    default object formatter

__ge__(other)

__getattr__
    Return getattr(self, name).

__gt__(other)

__hash__ = None

__invert__()

```

`__le__` (*other*)
`__lshift__` (*dx*)
`__lt__` (*other*)
`__ne__` (*other*)
`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.
`__or__` (*right*)
`__pow__` (*right*)
`__reduce__` ()
helper for pickle
`__reduce_ex__` ()
helper for pickle
`__repr__` ()
`__rmul__` (*right*)
`__rshift__` (*dx*)
`__setattr__`
Implement `setattr(self, name, value)`.
`__sizeof__` () → int
size of object in memory, in bytes
`__xor__` (*right*)
append (*x*)
add data *x* to Stats
applx (*f*, *var*=*'x'*)
function composition `f o self = self(f(x))`
apply (*f*, *right*=*None*)
function composition `self o f = f(self(x))`
average
avg
complexity ()
measures the complexity of Expr :return: int, sum of the precedence of used ops
extend (*data*)
html (***kwargs*)
isNum
isconstant
Returns True if Expr evaluates to a constant number or bool
mean
mu
plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

```

png (**kwargs)
remove (data)
    remove data from Stats :param data: value or iterable of values
render (fmt='svg', **kwargs)
save (filename, **kwargs)
stddev
sum
sum1
sum2
svg (**kwargs)
var
variance

```

`Goulib.stats.linear_regression(x, y, conf=None)`

Parameters

- **x, y** – iterable data
- **conf** – float confidence level [0..1]. If None, confidence intervals are not returned

Returns b0,b1,b2, (b0

Return the linear regression parameters and their <prob> confidence intervals.

ex: >>> linear_regression([.1,.2,.3],[10,11,11.5],0.95)

2.22 Goulib.table module

“mini pandas.DataFrame” Table class with Excel + CSV I/O, easy access to columns, HTML output, and much more.

`Goulib.table.attr(args)`

class `Goulib.table.Cell` (*data=None, align=None, fmt=None, tag=None, style={}*)

Bases: `object`

Table cell with HTML attributes

Parameters

- **data** – cell value(s) of any type
- **align** – string for HTML align attribute
- **fmt** – format string applied applied to data
- **tag** – called to build each cell. defaults to ‘td’
- **style** – dict or string for HTML style attribute

`__init__` (*data=None, align=None, fmt=None, tag=None, style={}*)

Parameters

- **data** – cell value(s) of any type

- **align** – string for HTML align attribute
- **fmt** – format string applied applied to data
- **tag** – called to build each cell. defaults to ‘td’
- **style** – dict or string for HTML style attribute

__repr__ ()

static read (*x*)

interprets *x* as int, float, string or None

html (***kwargs*)

Returns string HTML formatted cell:

- if data is int, default align=”right”
- if data is float, default align=”right” and fmt=’%0.2f’
- if data is `timedelta`, align = “right” and formatting is done by `datetime2.strftimedelta()`

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

helper for pickle

```

__reduce_ex__()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__() → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

class `Goulib.table.Row` (*data*, *align=None*, *fmt=None*, *tag=None*, *style={}*)
 Bases: `object`

Table row with HTML attributes

Parameters

- **data** – (list of) cell value(s) of any type
- **align** – (list of) string for HTML align attribute
- **fmt** – (list of) format string applied applied to data
- **tag** – (list of) tags called to build each cell. defaults to ‘td’
- **style** – (list of) dict or string for HTML style attribute

```
__init__(data, align=None, fmt=None, tag=None, style={})
```

Parameters

- **data** – (list of) cell value(s) of any type
- **align** – (list of) string for HTML align attribute
- **fmt** – (list of) format string applied applied to data
- **tag** – (list of) tags called to build each cell. defaults to ‘td’
- **style** – (list of) dict or string for HTML style attribute

```
__repr__()
```

```
html(cell_args={}, **kwargs)
    return in HTML format
```

```
__class__
    alias of type
```

```
__delattr__
    Implement delattr(self, name).
```

```
__dir__() → list
    default dir() implementation
```

```
__eq__
    Return self==value.
```

```
__format__()
    default object formatter
```

```
__ge__
    Return self>=value.
```

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class Goulib.table.**Table** (data=[], **kwargs)
Bases: list

Table class with CSV I/O, easy access to columns, HTML output

inits a table, optionally by reading a Excel, csv or html file :param data: list of list of cells, or string as filename
:param titles: optional list of strings used as column id :param footer: optional list of functions used as column reducers

__init__ (data=[], **kwargs)
inits a table, optionally by reading a Excel, csv or html file :param data: list of list of cells, or string as filename :param titles: optional list of strings used as column id :param footer: optional list of functions used as column reducers

__repr__ ()
Returns repr string of titles+5 first lines

__str__ ()
Returns string of full tables with linefeeds

html (head=None, foot=None, colstyle={}, **kwargs)
HTML representation of table

Parameters

- **head** – optional column headers, .titles by default

- **foot** – optional column footers, .footer by default
- **style** – (list of) dict of style attributes
- **kwargs** – optional parameters passed along to tag('table'... except: * start=optional start row * stop=optional end row used to display a subset of lines. in this case rows with '...' cells are displayed before and/or after the lines

Returns string HTML representation of table

load (*filename*, ***kwargs*)

read_element (*element*, ***kwargs*)

read table from a DOM element. :Warning: drops all formatting

read_html (*filename*, ***kwargs*)

read first table in HTML file

read_json (*filename*, ***kwargs*)

appends a json file made of lines dictionaries

read_xls (*filename*, ***kwargs*)

appends an Excel table

read_csv (*filename*, ***kwargs*)

appends a .csv or similar file to the table

save (*filename*, ***kwargs*)

write_xlsx (*filename*, ***kwargs*)

json (***kwargs*)

Returns string JSON representation of table

write_csv (*filename*, ***kwargs*)

write the table in Excel csv format, optionally transposed

__eq__ (*other*)

compare 2 Tables contents, mainly for tests

ncols ()

Returns number of columns, ignoring title

find_col (*title*)

finds a column from a part of the title

icol (*column*)

iterates a column

col (*column*, *title=False*)

cols (*title=False*)

iterator through columns

ttranspose (*titles_column=0*)

transpose table :param: titles_column :return: Table where rows are made from self's columns and vice-versa

index (*value*, *column=0*)

Returns int row number of first line where column contains value

__getitem__ (*n*)

get (*row*, *col*)

set (*row, col, value*)

setcol (*col, value, i=0*)
set column values :param col: int or string column index :param value: single value assigned to whole column or iterable assigned to each cell :param i: optional int : index of first row to assign

__add__
Return self+value.

__class__
alias of `type`

__contains__
Return key in self.

__delattr__
Implement delattr(self, name).

__delitem__
Delete self[key].

__dir__ () → list
default dir() implementation

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__ = None

__iadd__
Implement self+=value.

__imul__
Implement self*=value.

__iter__
Implement iter(self).

__le__
Return self<=value.

__len__
Return len(self).

__lt__
Return self<value.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__reversed__ ()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__ ()
L.__sizeof__() – size of L in memory, in bytes

append (line)
appends a line to table :param line: can be either: * a list * a dict or column names:values

clear () → None – remove all items from L

copy () → list – a shallow copy of L

count (value) → integer – return number of occurrences of value

extend (iterable) → None – extend list by appending elements from the iterable

insert ()
L.insert(index, object) – insert object before index

pop ([index]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (value) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

addcol (title, val=None, i=0)
add column to the right

sort (by, reverse=False)
sort by column

rowasdict (i)
returns a line as a dict

asdict ()

groupby_gen (by, sort=True, removecol=True)
generates subtables

groupby (by, sort=True, removecol=True)
ordered dictionary of subtables

hierarchy (by='Level', factory=<function Table.<lambda>>, linkfct=<function Table.<lambda>>)
builds a structure from a table containing a “level” column

applyf (*by, f, skiperrors=False*)
apply a function to a column :param by: column name of number :param f: function of the form lambda cell:content :param skiperrors: bool. if True, errors while running f are ignored :return: bool True if ok, False if skiperrors==True and conversion failed

to_datetime (*by, fmt='%Y-%m-%d %H:%M:%S', skiperrors=False*)
convert a column to datetime

to_date (*by, fmt='%Y-%m-%d', skiperrors=False*)
convert a column to date

to_time (*by, fmt='%H:%M:%S', skiperrors=False*)
convert a column to time

to_timedelta (*by, fmt=None, skiperrors=False*)
convert a column to time

total (*funcs*)
build a footer row by applying funcs to all columns

remove_lines_where (*f, value=(None, 0, '')*)
Parameters **f** – function of the form lambda line:bool returning True if line should be removed
Returns int number of lines removed

2.23 Goulib.tests module

utilities for unit tests (using nose)

`Goulib.tests.pprint_gen` (*iterable, indices=[0, 1, 2, -3, -2, -1], sep='...'*)
generates items at specified indices

`Goulib.tests.pprint` (*iterable, indices=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -3, -2, -1], timeout=1*)

class `Goulib.tests.TestCase` (*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None, places=7, delta=None, reltol=None*)

An equality assertion for ordered sequences (like lists and tuples). constraints on `seq1, seq2` from `unittest.TestCase.assertSequenceEqual` are mostly removed

Parameters

- **seq2** (*seq1,*) – iterables to compare for (quasi) equality
- **msg** – optional string message to use on failure instead of a list of differences
- **places** – int number of digits to consider in float comparisons. If `None`, enforces strict equality
- **delta** – optional float absolute tolerance value
- **reltol** – optional float relative tolerance value

base_types = ((`<class 'int'>`), (`<class 'str'>`), `<class 'str'>`, `<class 'bool'>`, `<class 'set'>`, `<class 'dict'>`)

assertEqual (*first, second, places=7, msg=None, delta=None, reltol=None*)

automatically calls `assertAlmostEqual` when needed :param first, second: objects to compare for (quasi) equality :param places: int number of digits to consider in float comparisons.

If None, forces strict equality

Parameters

- **msg** – optional string error message to display in case of failure
- **delta** – optional float absolute tolerance value
- **reltol** – optional float relative tolerance value

assertCountEqual (*seq1, seq2, msg=None*)

compare iterables converted to sets : order has no importance

__call__ (**args, **kwargs*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__ (*other*)

__format__ ()
default object formatter

__ge__
Return `self >= value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self > value`.

__hash__ ()

__init__ (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a `ValueError` if the instance does not have a method with the specified name.

__le__
Return `self <= value`.

__lt__
Return `self < value`.

__ne__
Return `self != value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

addCleanup (*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

addTypeEqualityFunc (*typeobj*, *function*)

Add a type specific assertEquals style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEquals().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

assertAlmostEqual (*first*, *second*, *places=None*, *msg=None*, *delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (**args*, ***kwargs*)

assertDictContainsSubset (*subset*, *dictionary*, *msg=None*)

Checks whether dictionary is a superset of subset.

assertDictEqual (*d1*, *d2*, *msg=None*)

assertEquals (**args*, ***kwargs*)

assertFalse (*expr*, *msg=None*)

Check that the expression is false.

assertGreater (*a*, *b*, *msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a*, *b*, *msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member*, *container*, *msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Args: list1: The first list to compare. list2: The second list to compare. msg: Optional message to use on failure instead of a list of differences.

assertLogs (*logger=None, level=None*)

Fail unless a log message of level *level* or higher is emitted on *logger_name* or its children. If omitted, *level* defaults to INFO and *logger* defaults to the root logger.

This method must be used as a context manager, and will yield a recording object with two attributes: *output* and *records*. At the end of the context manager, the *output* attribute will be a list of the matching formatted log messages and the *records* attribute will be a list of the corresponding LogRecord objects.

Example:

```
with self.assertLogs('foo', level='INFO') as cm:
    logging.getLogger('foo').info('first message')
    logging.getLogger('foo.bar').error('second message')
self.assertEqual(cm.output, ['INFO:foo:first message',
                             'ERROR:foo.bar:second message'])
```

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEquals (**args, **kwargs*)

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (**args, **kwargs*)

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegex (*text, unexpected_regex, msg=None*)

Fail the test if the text matches the regular expression.

assertNotRegexMatches (**args, **kwargs*)

assertRaises (*expected_exception, *args, **kwargs*)

Fail unless an exception of class *expected_exception* is raised by the callable when invoked with specified positional and keyword arguments. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertRaises(SomeException):  
    do_something()
```

An optional keyword argument ‘msg’ can be provided when `assertRaises` is used as a context object.

The context manager keeps a reference to the exception as the ‘exception’ attribute. This allows you to inspect the exception after the assertion:

```
with self.assertRaises(SomeException) as cm:  
    do_something()  
the_exception = cm.exception  
self.assertEqual(the_exception.error_code, 3)
```

assertRaisesRegex (*expected_exception, expected_regex, *args, **kwargs*)

Asserts that the message in a raised exception matches a regex.

Args: *expected_exception*: Exception class expected to be raised. *expected_regex*: Regex (re pattern object or string) expected

to be found in error message.

args: Function to be called and extra positional args. *kwargs*: Extra kwargs. *msg*: Optional message used in case of failure. Can only be used

when `assertRaisesRegex` is used as a context manager.

assertRaisesRegexp (**args, **kwargs*)

assertRegex (*text, expected_regex, msg=None*)

Fail the test unless the text matches the regular expression.

assertRegexMatches (**args, **kwargs*)

assertSetEqual (*set1, set2, msg=None*)

A set-specific equality assertion.

Args: *set1*: The first set to compare. *set2*: The second set to compare. *msg*: Optional message to use on failure instead of a list of

differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (*expr, msg=None*)

Check that the expression is true.

assertTupleEqual (*tuple1, tuple2, msg=None*)

A tuple-specific equality assertion.

Args: tuple1: The first tuple to compare. tuple2: The second tuple to compare. msg: Optional message to use on failure instead of a list of differences.

assertWarns (*expected_warning, *args, **kwargs*)

Fail unless a warning of class warnClass is triggered by the callable when invoked with specified positional and keyword arguments. If a different type of warning is triggered, it will not be handled: depending on the other warning filtering rules in effect, it might be silenced, printed out, or raised as an exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertWarns(SomeWarning):
    do_something()
```

An optional keyword argument 'msg' can be provided when assertWarns is used as a context object.

The context manager keeps a reference to the first matching warning as the 'warning' attribute; similarly, the 'filename' and 'lineno' attributes give you information about the line of Python code from which the warning was triggered. This allows you to inspect the warning after the assertion:

```
with self.assertWarns(SomeWarning) as cm:
    do_something()
the_warning = cm.warning
self.assertEqual(the_warning.some_attribute, 147)
```

assertWarnsRegex (*expected_warning, expected_regex, *args, **kwargs*)

Asserts that the message in a triggered warning matches a regexp. Basic functioning is similar to assertWarns() with the addition that only warnings whose messages also match the regular expression are considered successful matches.

Args: expected_warning: Warning class expected to be triggered. expected_regex: Regex (re pattern object or string) expected to be found in error message.

args: Function to be called and extra positional args. kwargs: Extra kwargs. msg: Optional message used in case of failure. Can only be used

when assertWarnsRegex is used as a context manager.

assert_ (**args, **kwargs*)

countTestCases ()

debug ()

Run the test without collecting errors in a TestResult

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after tearDown.

fail (*msg=None*)

Fail immediately, with the given message.

failIf (**args, **kwargs*)

failIfAlmostEqual (**args, **kwargs*)

failIfEqual (**args, **kwargs*)

failUnless (*args, **kwargs)

failUnlessAlmostEqual (*args, **kwargs)

failUnlessEqual (*args, **kwargs)

failUnlessRaises (*args, **kwargs)

failureException
alias of AssertionError

id()

longMessage = True

maxDiff = 640

run (result=None)

setUp ()
Hook method for setting up the test fixture before exercising it.

setUpClass ()
Hook method for setting up class fixture before running tests in the class.

shortDescription ()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (reason)
Skip this test.

subTest (msg=None, **params)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

tearDown ()
Hook method for deconstructing the test fixture after testing it.

tearDownClass ()
Hook method for deconstructing the class fixture after running all tests in the class.

Goulib.tests.**pep8** (name)

Goulib.tests.**setlog** (level=20, fmt='%(levelname)s: %(filename)s: %(funcName)s: %(message)s')
initializes logging :param level: logging level :param fmt: string

Goulib.tests.**runmodule** (level=20, verbosity=1, argv=[])

Parameters **argv** – optional list of string with additional options passed to nose.run
see <http://nose.readthedocs.org/en/latest/usage.html>

Goulib.tests.**runtests** (level=20, verbosity=1, argv=[])

Parameters **argv** – optional list of string with additional options passed to nose.run
see <http://nose.readthedocs.org/en/latest/usage.html>

2.24 Goulib.workdays module

WorkCalendar class with datetime operations on working hours, handling holidays merges and improves [BusinessHours](#) and [workdays](#) packages

```
class Goulib.workdays.WorkCalendar (worktime=[datetime.time(0, 0), datetime.time(23, 59, 59, 999999)], parent=[], weekends=(5, 6), holidays=set())
```

Bases: `object`

WorkCalendar class with datetime operations on working hours

```
__init__ (worktime=[datetime.time(0, 0), datetime.time(23, 59, 59, 999999)], parent=[], weekends=(5, 6), holidays=set())
```

start

end

setworktime (worktime)

addholidays (days)

add day(s) to to known holidays. dates with year==4 (to allow Feb 29th) apply every year note : holidays set may contain weekends too.

isworkday (day)

@return True if day is a work day

isworktime (time)

@return True if you're supposed to work at that time

nextworkday (day)

@return next work day

prevworkday (day)

@return previous work day

range (start, end)

range of workdays between start (included) and end (not included)

workdays (start_date, ndays)

list of ndays workdays from start

workday (start_date, ndays)

Same as Excel WORKDAY function. Returns a date that is the indicated number of working days before or after the starting date. Working days exclude weekends and any dates identified as holidays. Use WORKDAY to exclude weekends or holidays when you calculate invoice due dates, expected delivery times, or the number of days of work performed.

cast (time, retro=False)

force time to be in workhours

worktime (day)

@return interval of time worked a given day

workdatetime (day)

@return interval of datetime worked a given day

diff (t1, t2)

@return timedelta worktime between t1 and t2 (= t2-t1)

gethours (t1, t2)

@return fractional work hours between t1 and t2 (= t2-t1)

plus (*start*, *t*)
@return start time + t work time (positive or negative)

minus (*start*, *t*)
@return start time - t work time (positive or negative)

networkdays (*start_date*, *end_date*)
Same as Excel NETWORKDAYS function. Returns the number of whole working days between *start_date* and *end_date* (inclusive of both *start_date* and *end_date*). Working days exclude weekends and any dates identified in holidays. Use NETWORKDAYS to calculate employee benefits that accrue based on the number of days worked during a specific term

FRI = 4

MON = 0

SAT = 5

SUN = 6

THU = 3

TUE = 1

WED = 2

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

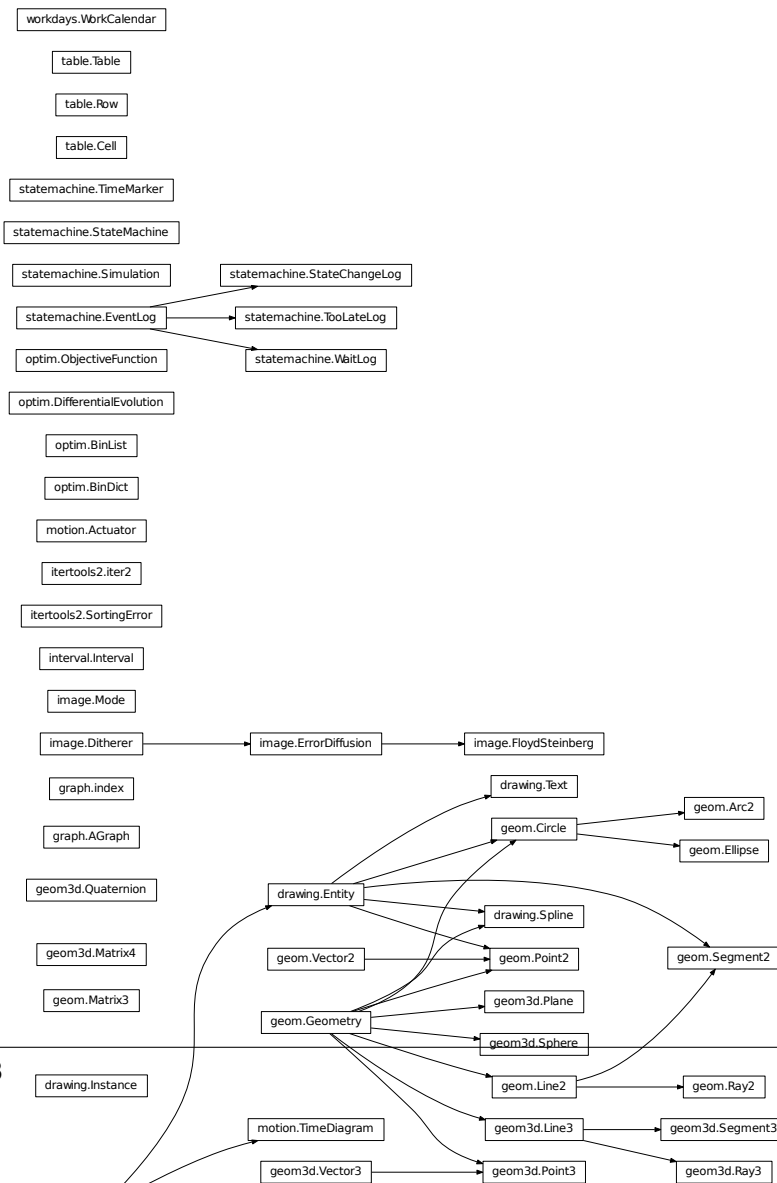
`Goulib.workdays.FullTime = <Goulib.workdays.WorkCalendar object>`
compatibility with <http://pypi.python.org/pypi/BusinessHours>

`Goulib.workdays.workday(start_date, ndays, holidays=[])`
Same as Excel WORKDAY function. Returns a date that is the indicated number of working days before or after the starting date. Working days exclude weekends and any dates identified as holidays. Use WORKDAY to exclude weekends or holidays when you calculate invoice due dates, expected delivery times, or the number of days of work performed.

`Goulib.workdays.networkdays(start_date, end_date, holidays=[])`
Same as Excel NETWORKDAYS function. Returns the number of whole working days between start_date and end_date (inclusive of both start_date and end_date). Working days exclude weekends and any dates identified in holidays. Use NETWORKDAYS to calculate employee benefits that accrue based on the number of days worked during a specific term

CHAPTER 3

Classes



CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`
- `changes`

g

Goulib.colors, 6
Goulib.container, 11
Goulib.datetime2, 14
Goulib.decorators, 24
Goulib.drawing, 31
Goulib.expr, 60
Goulib.geom, 64
Goulib.geom3d, 94
Goulib.graph, 105
Goulib.image, 145
Goulib.interval, 156
Goulib.itertools2, 165
Goulib.markup, 174
Goulib.math2, 190
Goulib.motion, 202
Goulib.optim, 214
Goulib.piecewise, 222
Goulib.plot, 224
Goulib.polynomial, 225
Goulib.statemachine, 228
Goulib.stats, 230
Goulib.table, 239
Goulib.tests, 246
Goulib.workdays, 253

Symbols

- `__abs__` (Goulib.datetime2.timedelta2 attribute), 21
- `__abs__()` (Goulib.geom.Arc2 method), 83
- `__abs__()` (Goulib.geom.Circle method), 80
- `__abs__()` (Goulib.geom.Ellipse method), 86
- `__abs__()` (Goulib.geom.Matrix3 method), 94
- `__abs__()` (Goulib.geom.Point2 method), 71
- `__abs__()` (Goulib.geom.Segment2 method), 77
- `__abs__()` (Goulib.geom.Vector2 method), 69
- `__abs__()` (Goulib.geom3d.Quaternion method), 102
- `__abs__()` (Goulib.geom3d.Segment3 method), 97
- `__abs__()` (Goulib.geom3d.Vector3 method), 95
- `__abs__()` (Goulib.image.Image method), 149
- `__abstractmethods__` (Goulib.drawing.Chain attribute), 29, 46
- `__abstractmethods__` (Goulib.drawing.Drawing attribute), 30, 56
- `__abstractmethods__` (Goulib.drawing.Group attribute), 28, 40
- `__abstractmethods__` (Goulib.drawing.Instance attribute), 28, 43
- `__abstractmethods__` (Goulib.drawing.Rect attribute), 29, 50
- `__abstractmethods__` (Goulib.drawing.Spline attribute), 27, 37
- `__abstractmethods__` (Goulib.geom.Arc2 attribute), 84
- `__abstractmethods__` (Goulib.geom.Circle attribute), 81
- `__abstractmethods__` (Goulib.geom.Ellipse attribute), 86
- `__abstractmethods__` (Goulib.geom.Geometry attribute), 65
- `__abstractmethods__` (Goulib.geom.Line2 attribute), 75
- `__abstractmethods__` (Goulib.geom.Point2 attribute), 71
- `__abstractmethods__` (Goulib.geom.Ray2 attribute), 76
- `__abstractmethods__` (Goulib.geom.Segment2 attribute), 77
- `__abstractmethods__` (Goulib.geom3d.Line3 attribute), 97
- `__abstractmethods__` (Goulib.geom3d.Plane attribute), 99
- `__abstractmethods__` (Goulib.geom3d.Point3 attribute), 96
- `__abstractmethods__` (Goulib.geom3d.Ray3 attribute), 97
- `__abstractmethods__` (Goulib.geom3d.Segment3 attribute), 97
- `__abstractmethods__` (Goulib.geom3d.Sphere attribute), 98
- `__abstractmethods__` (Goulib.interval.Intervals attribute), 159
- `__abstractmethods__` (Goulib.itertools2.keep attribute), 172
- `__add__` (Goulib.datetime2.date2 attribute), 17
- `__add__` (Goulib.datetime2.datetime2 attribute), 14
- `__add__` (Goulib.datetime2.timedelta2 attribute), 21
- `__add__` (Goulib.drawing.Chain attribute), 46
- `__add__` (Goulib.drawing.Drawing attribute), 56
- `__add__` (Goulib.drawing.Group attribute), 40
- `__add__` (Goulib.drawing.Rect attribute), 50
- `__add__` (Goulib.optim.BinList attribute), 217
- `__add__` (Goulib.table.Table attribute), 244
- `__add__()` (Goulib.colors.Color method), 7
- `__add__()` (Goulib.container.Sequence method), 13
- `__add__()` (Goulib.drawing.BBox method), 32
- `__add__()` (Goulib.expr.Expr method), 61
- `__add__()` (Goulib.geom.Point2 method), 71
- `__add__()` (Goulib.geom.Vector2 method), 68
- `__add__()` (Goulib.geom3d.Vector3 method), 95
- `__add__()` (Goulib.image.Image method), 150
- `__add__()` (Goulib.interval.Box method), 163
- `__add__()` (Goulib.interval.Interval method), 157
- `__add__()` (Goulib.interval.Intervals method), 159
- `__add__()` (Goulib.itertools2.iter2 method), 170
- `__add__()` (Goulib.piecewise.Piecewise method), 222
- `__add__()` (Goulib.polynomial.Polynomial method), 226
- `__add__()` (Goulib.stats.Discrete method), 233
- `__add__()` (Goulib.stats.Normal method), 237
- `__add__()` (Goulib.stats.PDF method), 234
- `__add__()` (Goulib.stats.Stats method), 231
- `__and__()` (Goulib.container.Sequence method), 13
- `__and__()` (Goulib.expr.Expr method), 61

`__and__()` (Goulib.piecewise.Piecewise method), 222
`__and__()` (Goulib.polynomial.Polynomial method), 226
`__and__()` (Goulib.stats.Normal method), 237
`__and__()` (Goulib.stats.PDF method), 234
`__bool__` (Goulib.datetime2.timedelta2 attribute), 21
`__bool__()` (Goulib.geom3d.Vector3 method), 95
`__bool__()` (Goulib.graph.DiGraph method), 126
`__bool__()` (Goulib.graph.GeoGraph method), 111
`__call__()` (Goulib.drawing.BBox method), 25, 32
`__call__()` (Goulib.expr.Expr method), 60
`__call__()` (Goulib.geom.Matrix3 method), 93
`__call__()` (Goulib.geom3d.Matrix4 method), 99
`__call__()` (Goulib.image.Ditherer method), 152
`__call__()` (Goulib.image.ErrorDiffusion method), 153
`__call__()` (Goulib.image.FloydSteinberg method), 154
`__call__()` (Goulib.interval.Box method), 163
`__call__()` (Goulib.interval.Intervals method), 159
`__call__()` (Goulib.markup.element method), 174
`__call__()` (Goulib.markup.page method), 176
`__call__()` (Goulib.motion.PVA method), 202
`__call__()` (Goulib.motion.Segment method), 204
`__call__()` (Goulib.motion.SegmentPoly method), 207
`__call__()` (Goulib.motion.Segments method), 205
`__call__()` (Goulib.optim.ObjectiveFunction method), 214
`__call__()` (Goulib.piecewise.Piecewise method), 222
`__call__()` (Goulib.polynomial.Polynomial method), 226
`__call__()` (Goulib.statemachine.StateMachine method), 230
`__call__()` (Goulib.statemachine.TimeMarker method), 229
`__call__()` (Goulib.stats.Discrete method), 233
`__call__()` (Goulib.stats.Normal method), 237
`__call__()` (Goulib.stats.PDF method), 234
`__call__()` (Goulib.tests.TestCase method), 247
`__cause__` (Goulib.itertools2.SortingError attribute), 171
`__cause__` (Goulib.markup.ArgumentError attribute), 184
`__cause__` (Goulib.markup.ClosingError attribute), 182
`__cause__` (Goulib.markup.CustomizationError attribute), 189
`__cause__` (Goulib.markup.DeprecationError attribute), 186
`__cause__` (Goulib.markup.InvalidElementError attribute), 185
`__cause__` (Goulib.markup.MarkupError attribute), 181
`__cause__` (Goulib.markup.ModeError attribute), 188
`__cause__` (Goulib.markup.OpeningError attribute), 183
`__class__` (Goulib.colors.Color attribute), 7
`__class__` (Goulib.colors.Palette attribute), 9
`__class__` (Goulib.container.Record attribute), 11
`__class__` (Goulib.container.Sequence attribute), 13
`__class__` (Goulib.datetime2.date2 attribute), 17
`__class__` (Goulib.datetime2.datetime2 attribute), 15
`__class__` (Goulib.datetime2.time2 attribute), 19
`__class__` (Goulib.datetime2.timedelta2 attribute), 21
`__class__` (Goulib.drawing.BBox attribute), 32
`__class__` (Goulib.drawing.Chain attribute), 46
`__class__` (Goulib.drawing.Drawing attribute), 56
`__class__` (Goulib.drawing.Entity attribute), 35
`__class__` (Goulib.drawing.Group attribute), 40
`__class__` (Goulib.drawing.Instance attribute), 43
`__class__` (Goulib.drawing.Rect attribute), 50
`__class__` (Goulib.drawing.Spline attribute), 37
`__class__` (Goulib.drawing.Text attribute), 54
`__class__` (Goulib.expr.Expr attribute), 61
`__class__` (Goulib.expr.TextVisitor attribute), 62
`__class__` (Goulib.geom.Arc2 attribute), 84
`__class__` (Goulib.geom.Circle attribute), 81
`__class__` (Goulib.geom.Ellipse attribute), 86
`__class__` (Goulib.geom.Geometry attribute), 65
`__class__` (Goulib.geom.Line2 attribute), 75
`__class__` (Goulib.geom.Matrix3 attribute), 93
`__class__` (Goulib.geom.Point2 attribute), 71
`__class__` (Goulib.geom.Ray2 attribute), 76
`__class__` (Goulib.geom.Segment2 attribute), 77
`__class__` (Goulib.geom.Vector2 attribute), 69
`__class__` (Goulib.graph.AGraph attribute), 109
`__class__` (Goulib.graph.DiGraph attribute), 126
`__class__` (Goulib.graph.GeoGraph attribute), 111
`__class__` (Goulib.graph.index attribute), 108
`__class__` (Goulib.graph.index.Index attribute), 107
`__class__` (Goulib.graph.index.Property attribute), 105
`__class__` (Goulib.image.Ditherer attribute), 152
`__class__` (Goulib.image.ErrorDiffusion attribute), 153
`__class__` (Goulib.image.FloydSteinberg attribute), 154
`__class__` (Goulib.image.Image attribute), 150
`__class__` (Goulib.image.Mode attribute), 145
`__class__` (Goulib.interval.Box attribute), 163
`__class__` (Goulib.interval.Interval attribute), 157
`__class__` (Goulib.interval.Intervals attribute), 159
`__class__` (Goulib.itertools2.SortingError attribute), 171
`__class__` (Goulib.itertools2.iter2 attribute), 170
`__class__` (Goulib.itertools2.keep attribute), 172
`__class__` (Goulib.markup.ArgumentError attribute), 184
`__class__` (Goulib.markup.ClosingError attribute), 182
`__class__` (Goulib.markup.CustomizationError attribute), 189
`__class__` (Goulib.markup.DeprecationError attribute), 186
`__class__` (Goulib.markup.InvalidElementError attribute), 185
`__class__` (Goulib.markup.MarkupError attribute), 181
`__class__` (Goulib.markup.ModeError attribute), 188
`__class__` (Goulib.markup.OpeningError attribute), 183
`__class__` (Goulib.markup.dummy attribute), 178
`__class__` (Goulib.markup.element attribute), 174
`__class__` (Goulib.markup.page attribute), 177
`__class__` (Goulib.markup.russell attribute), 180

- `__class__` (Goulib.motion.Actuator attribute), 210
- `__class__` (Goulib.motion.PVA attribute), 202
- `__class__` (Goulib.motion.Segment attribute), 204
- `__class__` (Goulib.motion.SegmentPoly attribute), 207
- `__class__` (Goulib.motion.Segments attribute), 205
- `__class__` (Goulib.motion.TimeDiagram attribute), 211
- `__class__` (Goulib.optim.BinDict attribute), 215
- `__class__` (Goulib.optim.BinList attribute), 217
- `__class__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__class__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__class__` (Goulib.piecewise.Piecewise attribute), 222
- `__class__` (Goulib.plot.Plot attribute), 224
- `__class__` (Goulib.polynomial.Polynomial attribute), 226
- `__class__` (Goulib.stats.Discrete attribute), 233
- `__class__` (Goulib.stats.Normal attribute), 237
- `__class__` (Goulib.stats.PDF attribute), 234
- `__class__` (Goulib.stats.Stats attribute), 231
- `__class__` (Goulib.table.Cell attribute), 240
- `__class__` (Goulib.table.Row attribute), 241
- `__class__` (Goulib.table.Table attribute), 244
- `__class__` (Goulib.tests.TestCase attribute), 247
- `__class__` (Goulib.workdays.WorkCalendar attribute), 254
- `__contains__` (Goulib.drawing.Chain attribute), 46
- `__contains__` (Goulib.drawing.Drawing attribute), 56
- `__contains__` (Goulib.drawing.Group attribute), 40
- `__contains__` (Goulib.drawing.Rect attribute), 50
- `__contains__` (Goulib.optim.BinList attribute), 218
- `__contains__` (Goulib.table.Table attribute), 244
- `__contains__()` (Goulib.colors.Palette method), 9
- `__contains__()` (Goulib.container.Record method), 11
- `__contains__()` (Goulib.container.Sequence method), 13
- `__contains__()` (Goulib.drawing.BBox method), 25, 32
- `__contains__()` (Goulib.drawing.Instance method), 43
- `__contains__()` (Goulib.drawing.Spline method), 37
- `__contains__()` (Goulib.geom.Arc2 method), 84
- `__contains__()` (Goulib.geom.Circle method), 81
- `__contains__()` (Goulib.geom.Ellipse method), 86
- `__contains__()` (Goulib.geom.Geometry method), 65
- `__contains__()` (Goulib.geom.Line2 method), 75
- `__contains__()` (Goulib.geom.Point2 method), 70
- `__contains__()` (Goulib.geom.Ray2 method), 76
- `__contains__()` (Goulib.geom.Segment2 method), 77
- `__contains__()` (Goulib.geom3d.Sphere method), 98
- `__contains__()` (Goulib.graph.DiGraph method), 126
- `__contains__()` (Goulib.graph.GeoGraph method), 111
- `__contains__()` (Goulib.graph.index.Index method), 107
- `__contains__()` (Goulib.interval.Box method), 163
- `__contains__()` (Goulib.interval.Interval method), 157
- `__contains__()` (Goulib.interval.Intervals method), 159
- `__contains__()` (Goulib.markup.russell method), 179
- `__contains__()` (Goulib.optim.BinDict method), 216
- `__context__` (Goulib.itertools2.SortingError attribute), 171
- `__context__` (Goulib.markup.ArgumentError attribute), 184
- `__context__` (Goulib.markup.ClosingError attribute), 182
- `__context__` (Goulib.markup.CustomizationError attribute), 189
- `__context__` (Goulib.markup.DeprecationError attribute), 187
- `__context__` (Goulib.markup.InvalidElementError attribute), 185
- `__context__` (Goulib.markup.MarkupError attribute), 181
- `__context__` (Goulib.markup.ModeError attribute), 188
- `__context__` (Goulib.markup.OpeningError attribute), 183
- `__copy__()` (Goulib.drawing.Chain method), 47
- `__copy__()` (Goulib.drawing.Drawing method), 56
- `__copy__()` (Goulib.drawing.Group method), 27, 39
- `__copy__()` (Goulib.drawing.Rect method), 50
- `__copy__()` (Goulib.interval.Intervals method), 159
- `__delattr__` (Goulib.colors.Color attribute), 7
- `__delattr__` (Goulib.colors.Palette attribute), 9
- `__delattr__` (Goulib.container.Record attribute), 11
- `__delattr__` (Goulib.container.Sequence attribute), 13
- `__delattr__` (Goulib.datetime2.date2 attribute), 17
- `__delattr__` (Goulib.datetime2.datetime2 attribute), 15
- `__delattr__` (Goulib.datetime2.time2 attribute), 19
- `__delattr__` (Goulib.datetime2.timedelta2 attribute), 21
- `__delattr__` (Goulib.drawing.BBox attribute), 32
- `__delattr__` (Goulib.drawing.Chain attribute), 47
- `__delattr__` (Goulib.drawing.Drawing attribute), 56
- `__delattr__` (Goulib.drawing.Entity attribute), 35
- `__delattr__` (Goulib.drawing.Group attribute), 40
- `__delattr__` (Goulib.drawing.Instance attribute), 43
- `__delattr__` (Goulib.drawing.Rect attribute), 50
- `__delattr__` (Goulib.drawing.Spline attribute), 37
- `__delattr__` (Goulib.drawing.Text attribute), 54
- `__delattr__` (Goulib.expr.Expr attribute), 61
- `__delattr__` (Goulib.expr.TextVisitor attribute), 62
- `__delattr__` (Goulib.geom.Arc2 attribute), 84
- `__delattr__` (Goulib.geom.Circle attribute), 81
- `__delattr__` (Goulib.geom.Ellipse attribute), 86
- `__delattr__` (Goulib.geom.Geometry attribute), 65
- `__delattr__` (Goulib.geom.Line2 attribute), 75
- `__delattr__` (Goulib.geom.Matrix3 attribute), 93
- `__delattr__` (Goulib.geom.Point2 attribute), 71
- `__delattr__` (Goulib.geom.Ray2 attribute), 76
- `__delattr__` (Goulib.geom.Segment2 attribute), 78
- `__delattr__` (Goulib.geom.Vector2 attribute), 69
- `__delattr__` (Goulib.graph.AGraph attribute), 109
- `__delattr__` (Goulib.graph.DiGraph attribute), 126
- `__delattr__` (Goulib.graph.GeoGraph attribute), 111
- `__delattr__` (Goulib.graph.index attribute), 108
- `__delattr__` (Goulib.graph.index.Index attribute), 107

`__delattr__` (Goulib.graph.index.Property attribute), 106
`__delattr__` (Goulib.image.Ditherer attribute), 152
`__delattr__` (Goulib.image.ErrorDiffusion attribute), 153
`__delattr__` (Goulib.image.FloydSteinberg attribute), 154
`__delattr__` (Goulib.image.Image attribute), 150
`__delattr__` (Goulib.image.Mode attribute), 145
`__delattr__` (Goulib.interval.Box attribute), 163
`__delattr__` (Goulib.interval.Interval attribute), 157
`__delattr__` (Goulib.interval.Intervals attribute), 160
`__delattr__` (Goulib.itertools2.SortingError attribute), 171
`__delattr__` (Goulib.itertools2.iter2 attribute), 170
`__delattr__` (Goulib.itertools2.keep attribute), 173
`__delattr__` (Goulib.markup.ArgumentError attribute), 184
`__delattr__` (Goulib.markup.ClosingError attribute), 182
`__delattr__` (Goulib.markup.CustomizationError attribute), 189
`__delattr__` (Goulib.markup.DeprecationError attribute), 187
`__delattr__` (Goulib.markup.InvalidElementError attribute), 185
`__delattr__` (Goulib.markup.MarkupError attribute), 181
`__delattr__` (Goulib.markup.ModeError attribute), 188
`__delattr__` (Goulib.markup.OpeningError attribute), 183
`__delattr__` (Goulib.markup.dummy attribute), 179
`__delattr__` (Goulib.markup.element attribute), 174
`__delattr__` (Goulib.markup.page attribute), 178
`__delattr__` (Goulib.markup.russell attribute), 180
`__delattr__` (Goulib.motion.Actuator attribute), 210
`__delattr__` (Goulib.motion.PVA attribute), 202
`__delattr__` (Goulib.motion.Segment attribute), 204
`__delattr__` (Goulib.motion.SegmentPoly attribute), 207
`__delattr__` (Goulib.motion.Segments attribute), 205
`__delattr__` (Goulib.motion.TimeDiagram attribute), 211
`__delattr__` (Goulib.optim.BinDict attribute), 216
`__delattr__` (Goulib.optim.BinList attribute), 218
`__delattr__` (Goulib.optim.DifferentialEvolution attribute), 221
`__delattr__` (Goulib.optim.ObjectiveFunction attribute), 214
`__delattr__` (Goulib.piecewise.Piecewise attribute), 222
`__delattr__` (Goulib.plot.Plot attribute), 224
`__delattr__` (Goulib.polynomial.Polynomial attribute), 226
`__delattr__` (Goulib.stats.Discrete attribute), 233
`__delattr__` (Goulib.stats.Normal attribute), 237
`__delattr__` (Goulib.stats.PDF attribute), 234
`__delattr__` (Goulib.stats.Stats attribute), 232
`__delattr__` (Goulib.table.Cell attribute), 240
`__delattr__` (Goulib.table.Row attribute), 241
`__delattr__` (Goulib.table.Table attribute), 244
`__delattr__` (Goulib.tests.TestCase attribute), 247
`__delattr__` (Goulib.workdays.WorkCalendar attribute), 254
`__delitem__` (Goulib.colors.Palette attribute), 9
`__delitem__` (Goulib.container.Record attribute), 11
`__delitem__` (Goulib.drawing.BBox attribute), 32
`__delitem__` (Goulib.drawing.Chain attribute), 47
`__delitem__` (Goulib.drawing.Drawing attribute), 56
`__delitem__` (Goulib.drawing.Group attribute), 40
`__delitem__` (Goulib.drawing.Rect attribute), 50
`__delitem__` (Goulib.graph.index.Index attribute), 107
`__delitem__` (Goulib.interval.Box attribute), 163
`__delitem__` (Goulib.interval.Interval attribute), 157
`__delitem__` (Goulib.optim.BinList attribute), 218
`__delitem__` (Goulib.table.Table attribute), 244
`__delitem__` () (Goulib.interval.Intervals method), 160
`__delitem__` () (Goulib.optim.BinDict method), 215
`__dir__` () (Goulib.colors.Color method), 7
`__dir__` () (Goulib.colors.Palette method), 9
`__dir__` () (Goulib.container.Record method), 11
`__dir__` () (Goulib.container.Sequence method), 13
`__dir__` () (Goulib.datetime2.date2 method), 17
`__dir__` () (Goulib.datetime2.datetime2 method), 15
`__dir__` () (Goulib.datetime2.time2 method), 19
`__dir__` () (Goulib.datetime2.timedelta2 method), 21
`__dir__` () (Goulib.drawing.BBox method), 32
`__dir__` () (Goulib.drawing.Chain method), 47
`__dir__` () (Goulib.drawing.Drawing method), 56
`__dir__` () (Goulib.drawing.Entity method), 35
`__dir__` () (Goulib.drawing.Group method), 40
`__dir__` () (Goulib.drawing.Instance method), 43
`__dir__` () (Goulib.drawing.Rect method), 50
`__dir__` () (Goulib.drawing.Spline method), 37
`__dir__` () (Goulib.drawing.Text method), 54
`__dir__` () (Goulib.expr.Expr method), 61
`__dir__` () (Goulib.expr.TextVisitor method), 62
`__dir__` () (Goulib.geom.Arc2 method), 84
`__dir__` () (Goulib.geom.Circle method), 81
`__dir__` () (Goulib.geom.Ellipse method), 86
`__dir__` () (Goulib.geom.Geometry method), 65
`__dir__` () (Goulib.geom.Line2 method), 75
`__dir__` () (Goulib.geom.Matrix3 method), 93
`__dir__` () (Goulib.geom.Point2 method), 71
`__dir__` () (Goulib.geom.Ray2 method), 76
`__dir__` () (Goulib.geom.Segment2 method), 78
`__dir__` () (Goulib.geom.Vector2 method), 69
`__dir__` () (Goulib.graph.AGraph method), 109
`__dir__` () (Goulib.graph.DiGraph method), 126
`__dir__` () (Goulib.graph.GeoGraph method), 111
`__dir__` () (Goulib.graph.index method), 108
`__dir__` () (Goulib.graph.index.Index method), 107
`__dir__` () (Goulib.graph.index.Property method), 106
`__dir__` () (Goulib.image.Ditherer method), 152
`__dir__` () (Goulib.image.ErrorDiffusion method), 153
`__dir__` () (Goulib.image.FloydSteinberg method), 154
`__dir__` () (Goulib.image.Image method), 150
`__dir__` () (Goulib.image.Mode method), 145

- `__dir__` (Goulib.interval.Box method), 163
- `__dir__` (Goulib.interval.Interval method), 158
- `__dir__` (Goulib.interval.Intervals method), 160
- `__dir__` (Goulib.itertools2.SortingError method), 171
- `__dir__` (Goulib.itertools2.iter2 method), 170
- `__dir__` (Goulib.itertools2.keep method), 173
- `__dir__` (Goulib.markup.ArgumentError method), 184
- `__dir__` (Goulib.markup.ClosingError method), 182
- `__dir__` (Goulib.markup.CustomizationError method), 189
- `__dir__` (Goulib.markup.DeprecationError method), 187
- `__dir__` (Goulib.markup.InvalidElementError method), 185
- `__dir__` (Goulib.markup.MarkupError method), 181
- `__dir__` (Goulib.markup.ModeError method), 188
- `__dir__` (Goulib.markup.OpeningError method), 183
- `__dir__` (Goulib.markup.dummy method), 179
- `__dir__` (Goulib.markup.element method), 174
- `__dir__` (Goulib.markup.page method), 178
- `__dir__` (Goulib.markup.russell method), 180
- `__dir__` (Goulib.motion.Actuator method), 210
- `__dir__` (Goulib.motion.PVA method), 202
- `__dir__` (Goulib.motion.Segment method), 204
- `__dir__` (Goulib.motion.SegmentPoly method), 207
- `__dir__` (Goulib.motion.Segments method), 206
- `__dir__` (Goulib.motion.TimeDiagram method), 211
- `__dir__` (Goulib.optim.BinDict method), 216
- `__dir__` (Goulib.optim.BinList method), 218
- `__dir__` (Goulib.optim.DifferentialEvolution method), 221
- `__dir__` (Goulib.optim.ObjectiveFunction method), 214
- `__dir__` (Goulib.piecewise.Piecewise method), 223
- `__dir__` (Goulib.plot.Plot method), 224
- `__dir__` (Goulib.polynomial.Polynomial method), 226
- `__dir__` (Goulib.stats.Discrete method), 233
- `__dir__` (Goulib.stats.Normal method), 237
- `__dir__` (Goulib.stats.PDF method), 234
- `__dir__` (Goulib.stats.Stats method), 232
- `__dir__` (Goulib.table.Cell method), 240
- `__dir__` (Goulib.table.Row method), 241
- `__dir__` (Goulib.table.Table method), 244
- `__dir__` (Goulib.tests.TestCase method), 247
- `__dir__` (Goulib.workdays.WorkCalendar method), 254
- `__div__` (Goulib.expr.Expr method), 61
- `__div__` (Goulib.geom.Point2 method), 71
- `__div__` (Goulib.geom.Vector2 method), 69
- `__div__` (Goulib.geom3d.Vector3 method), 95
- `__div__` (Goulib.image.Image method), 150
- `__div__` (Goulib.piecewise.Piecewise method), 223
- `__div__` (Goulib.polynomial.Polynomial method), 226
- `__div__` (Goulib.stats.Normal method), 237
- `__div__` (Goulib.stats.PDF method), 234
- `__divmod__` (Goulib.datetime2.timedelta2 attribute), 21
- `__eq__` (Goulib.colors.Palette attribute), 9
- `__eq__` (Goulib.container.Record attribute), 11
- `__eq__` (Goulib.container.Sequence attribute), 13
- `__eq__` (Goulib.datetime2.date2 attribute), 17
- `__eq__` (Goulib.datetime2.datetime2 attribute), 15
- `__eq__` (Goulib.datetime2.time2 attribute), 19
- `__eq__` (Goulib.datetime2.timedelta2 attribute), 21
- `__eq__` (Goulib.drawing.BBox attribute), 32
- `__eq__` (Goulib.drawing.Chain attribute), 47
- `__eq__` (Goulib.drawing.Drawing attribute), 56
- `__eq__` (Goulib.drawing.Entity attribute), 35
- `__eq__` (Goulib.drawing.Group attribute), 40
- `__eq__` (Goulib.drawing.Instance attribute), 43
- `__eq__` (Goulib.drawing.Rect attribute), 50
- `__eq__` (Goulib.drawing.Spline attribute), 37
- `__eq__` (Goulib.drawing.Text attribute), 54
- `__eq__` (Goulib.expr.TextVisitor attribute), 62
- `__eq__` (Goulib.geom.Geometry attribute), 65
- `__eq__` (Goulib.graph.AGraph attribute), 109
- `__eq__` (Goulib.graph.index attribute), 108
- `__eq__` (Goulib.graph.index.Index attribute), 107
- `__eq__` (Goulib.graph.index.Property attribute), 106
- `__eq__` (Goulib.image.Ditherer attribute), 152
- `__eq__` (Goulib.image.ErrorDiffusion attribute), 153
- `__eq__` (Goulib.image.FloydSteinberg attribute), 154
- `__eq__` (Goulib.image.Image attribute), 151
- `__eq__` (Goulib.image.Mode attribute), 145
- `__eq__` (Goulib.interval.Box attribute), 163
- `__eq__` (Goulib.itertools2.SortingError attribute), 171
- `__eq__` (Goulib.itertools2.iter2 attribute), 170
- `__eq__` (Goulib.itertools2.keep attribute), 173
- `__eq__` (Goulib.markup.ArgumentError attribute), 184
- `__eq__` (Goulib.markup.ClosingError attribute), 182
- `__eq__` (Goulib.markup.CustomizationError attribute), 189
- `__eq__` (Goulib.markup.DeprecationError attribute), 187
- `__eq__` (Goulib.markup.InvalidElementError attribute), 185
- `__eq__` (Goulib.markup.MarkupError attribute), 181
- `__eq__` (Goulib.markup.ModeError attribute), 188
- `__eq__` (Goulib.markup.OpeningError attribute), 183
- `__eq__` (Goulib.markup.dummy attribute), 179
- `__eq__` (Goulib.markup.element attribute), 174
- `__eq__` (Goulib.markup.page attribute), 178
- `__eq__` (Goulib.markup.russell attribute), 180
- `__eq__` (Goulib.motion.Actuator attribute), 210
- `__eq__` (Goulib.motion.PVA attribute), 202
- `__eq__` (Goulib.motion.Segment attribute), 204
- `__eq__` (Goulib.motion.SegmentPoly attribute), 207
- `__eq__` (Goulib.motion.Segments attribute), 206
- `__eq__` (Goulib.motion.TimeDiagram attribute), 211
- `__eq__` (Goulib.optim.BinDict attribute), 216
- `__eq__` (Goulib.optim.BinList attribute), 218

`__eq__` (Goulib.optim.DifferentialEvolution attribute), 221

`__eq__` (Goulib.optim.ObjectiveFunction attribute), 214

`__eq__` (Goulib.plot.Plot attribute), 224

`__eq__` (Goulib.stats.Discrete attribute), 233

`__eq__` (Goulib.stats.Stats attribute), 232

`__eq__` (Goulib.table.Cell attribute), 240

`__eq__` (Goulib.table.Row attribute), 241

`__eq__` (Goulib.workdays.WorkCalendar attribute), 254

`__eq__()` (Goulib.colors.Color method), 7

`__eq__()` (Goulib.expr.Expr method), 61

`__eq__()` (Goulib.geom.Arc2 method), 84

`__eq__()` (Goulib.geom.Circle method), 80

`__eq__()` (Goulib.geom.Ellipse method), 86

`__eq__()` (Goulib.geom.Line2 method), 75

`__eq__()` (Goulib.geom.Matrix3 method), 93

`__eq__()` (Goulib.geom.Point2 method), 71

`__eq__()` (Goulib.geom.Ray2 method), 76

`__eq__()` (Goulib.geom.Segment2 method), 78

`__eq__()` (Goulib.geom.Vector2 method), 68

`__eq__()` (Goulib.geom3d.Vector3 method), 94

`__eq__()` (Goulib.graph.DiGraph method), 126

`__eq__()` (Goulib.graph.GeoGraph method), 111

`__eq__()` (Goulib.interval.Interval method), 157

`__eq__()` (Goulib.interval.Intervals method), 160

`__eq__()` (Goulib.piecewise.Piecewise method), 223

`__eq__()` (Goulib.polynomial.Polynomial method), 226

`__eq__()` (Goulib.stats.Normal method), 237

`__eq__()` (Goulib.stats.PDF method), 234

`__eq__()` (Goulib.table.Table method), 243

`__eq__()` (Goulib.tests.TestCase method), 247

`__float__()` (Goulib.expr.Expr method), 60

`__float__()` (Goulib.piecewise.Piecewise method), 223

`__float__()` (Goulib.polynomial.Polynomial method), 226

`__float__()` (Goulib.stats.Normal method), 237

`__float__()` (Goulib.stats.PDF method), 234

`__floordiv__` (Goulib.datetime2.timedelta2 attribute), 21

`__floordiv__()` (Goulib.geom.Point2 method), 71

`__floordiv__()` (Goulib.geom.Vector2 method), 69

`__floordiv__()` (Goulib.geom3d.Vector3 method), 95

`__format__()` (Goulib.colors.Color method), 8

`__format__()` (Goulib.colors.Palette method), 9

`__format__()` (Goulib.container.Record method), 11

`__format__()` (Goulib.container.Sequence method), 13

`__format__()` (Goulib.datetime2.date2 method), 17

`__format__()` (Goulib.datetime2.datetime2 method), 15

`__format__()` (Goulib.datetime2.time2 method), 19

`__format__()` (Goulib.datetime2.timedelta2 method), 21

`__format__()` (Goulib.drawing.BBox method), 32

`__format__()` (Goulib.drawing.Chain method), 47

`__format__()` (Goulib.drawing.Drawing method), 56

`__format__()` (Goulib.drawing.Entity method), 36

`__format__()` (Goulib.drawing.Group method), 40

`__format__()` (Goulib.drawing.Instance method), 43

`__format__()` (Goulib.drawing.Rect method), 50

`__format__()` (Goulib.drawing.Spline method), 37

`__format__()` (Goulib.drawing.Text method), 54

`__format__()` (Goulib.expr.Expr method), 61

`__format__()` (Goulib.expr.TextVisitor method), 62

`__format__()` (Goulib.geom.Arc2 method), 84

`__format__()` (Goulib.geom.Circle method), 81

`__format__()` (Goulib.geom.Ellipse method), 86

`__format__()` (Goulib.geom.Geometry method), 65

`__format__()` (Goulib.geom.Line2 method), 75

`__format__()` (Goulib.geom.Matrix3 method), 93

`__format__()` (Goulib.geom.Point2 method), 71

`__format__()` (Goulib.geom.Ray2 method), 76

`__format__()` (Goulib.geom.Segment2 method), 78

`__format__()` (Goulib.geom.Vector2 method), 69

`__format__()` (Goulib.graph.AGraph method), 109

`__format__()` (Goulib.graph.DiGraph method), 126

`__format__()` (Goulib.graph.GeoGraph method), 111

`__format__()` (Goulib.graph.index method), 108

`__format__()` (Goulib.graph.index.Index method), 107

`__format__()` (Goulib.graph.index.Property method), 106

`__format__()` (Goulib.image.Ditherer method), 152

`__format__()` (Goulib.image.ErrorDiffusion method), 153

`__format__()` (Goulib.image.FloydSteinberg method), 154

`__format__()` (Goulib.image.Image method), 151

`__format__()` (Goulib.image.Mode method), 145

`__format__()` (Goulib.interval.Box method), 163

`__format__()` (Goulib.interval.Interval method), 158

`__format__()` (Goulib.interval.Intervals method), 160

`__format__()` (Goulib.itertools2.SortingError method), 171

`__format__()` (Goulib.itertools2.iter2 method), 170

`__format__()` (Goulib.itertools2.keep method), 173

`__format__()` (Goulib.markup.ArgumentError method), 184

`__format__()` (Goulib.markup.ClosingError method), 182

`__format__()` (Goulib.markup.CustomizationError method), 189

`__format__()` (Goulib.markup.DeprecationError method), 187

`__format__()` (Goulib.markup.InvalidElementError method), 186

`__format__()` (Goulib.markup.MarkupError method), 181

`__format__()` (Goulib.markup.ModeError method), 188

`__format__()` (Goulib.markup.OpeningError method), 183

`__format__()` (Goulib.markup.dummy method), 179

`__format__()` (Goulib.markup.element method), 174

`__format__()` (Goulib.markup.page method), 178

`__format__()` (Goulib.markup.russell method), 180

`__format__()` (Goulib.motion.Actuator method), 210

`__format__()` (Goulib.motion.PVA method), 202

`__format__()` (Goulib.motion.Segment method), 204

- `__format__()` (Goulib.motion.SegmentPoly method), 207
- `__format__()` (Goulib.motion.Segments method), 206
- `__format__()` (Goulib.motion.TimeDiagram method), 211
- `__format__()` (Goulib.optim.BinDict method), 216
- `__format__()` (Goulib.optim.BinList method), 218
- `__format__()` (Goulib.optim.DifferentialEvolution method), 221
- `__format__()` (Goulib.optim.ObjectiveFunction method), 214
- `__format__()` (Goulib.piecewise.Piecewise method), 223
- `__format__()` (Goulib.plot.Plot method), 224
- `__format__()` (Goulib.polynomial.Polynomial method), 226
- `__format__()` (Goulib.stats.Discrete method), 233
- `__format__()` (Goulib.stats.Normal method), 237
- `__format__()` (Goulib.stats.PDF method), 235
- `__format__()` (Goulib.stats.Stats method), 232
- `__format__()` (Goulib.table.Cell method), 240
- `__format__()` (Goulib.table.Row method), 241
- `__format__()` (Goulib.table.Table method), 244
- `__format__()` (Goulib.tests.TestCase method), 247
- `__format__()` (Goulib.workdays.WorkCalendar method), 254
- `__ge__` (Goulib.colors.Color attribute), 8
- `__ge__` (Goulib.colors.Palette attribute), 9
- `__ge__` (Goulib.container.Record attribute), 11
- `__ge__` (Goulib.container.Sequence attribute), 13
- `__ge__` (Goulib.datetime2.date2 attribute), 17
- `__ge__` (Goulib.datetime2.datetime2 attribute), 15
- `__ge__` (Goulib.datetime2.time2 attribute), 19
- `__ge__` (Goulib.datetime2.timedelta2 attribute), 21
- `__ge__` (Goulib.drawing.BBox attribute), 33
- `__ge__` (Goulib.drawing.Chain attribute), 47
- `__ge__` (Goulib.drawing.Drawing attribute), 56
- `__ge__` (Goulib.drawing.Entity attribute), 36
- `__ge__` (Goulib.drawing.Group attribute), 40
- `__ge__` (Goulib.drawing.Instance attribute), 44
- `__ge__` (Goulib.drawing.Rect attribute), 50
- `__ge__` (Goulib.drawing.Spline attribute), 37
- `__ge__` (Goulib.drawing.Text attribute), 54
- `__ge__` (Goulib.expr.TextVisitor attribute), 62
- `__ge__` (Goulib.geom.Arc2 attribute), 84
- `__ge__` (Goulib.geom.Circle attribute), 81
- `__ge__` (Goulib.geom.Ellipse attribute), 87
- `__ge__` (Goulib.geom.Geometry attribute), 65
- `__ge__` (Goulib.geom.Line2 attribute), 75
- `__ge__` (Goulib.geom.Matrix3 attribute), 93
- `__ge__` (Goulib.geom.Point2 attribute), 71
- `__ge__` (Goulib.geom.Ray2 attribute), 76
- `__ge__` (Goulib.geom.Segment2 attribute), 78
- `__ge__` (Goulib.geom.Vector2 attribute), 69
- `__ge__` (Goulib.graph.AGraph attribute), 109
- `__ge__` (Goulib.graph.DiGraph attribute), 126
- `__ge__` (Goulib.graph.GeoGraph attribute), 111
- `__ge__` (Goulib.graph.index attribute), 108
- `__ge__` (Goulib.graph.index.Index attribute), 107
- `__ge__` (Goulib.graph.index.Property attribute), 106
- `__ge__` (Goulib.image.Ditherer attribute), 152
- `__ge__` (Goulib.image.ErrorDiffusion attribute), 153
- `__ge__` (Goulib.image.FloydSteinberg attribute), 154
- `__ge__` (Goulib.image.Image attribute), 151
- `__ge__` (Goulib.image.Mode attribute), 145
- `__ge__` (Goulib.interval.Box attribute), 163
- `__ge__` (Goulib.interval.Interval attribute), 158
- `__ge__` (Goulib.itertools2.SortingError attribute), 171
- `__ge__` (Goulib.itertools2.iter2 attribute), 170
- `__ge__` (Goulib.itertools2.keep attribute), 173
- `__ge__` (Goulib.markup.ArgumentError attribute), 184
- `__ge__` (Goulib.markup.ClosingError attribute), 182
- `__ge__` (Goulib.markup.CustomizationError attribute), 189
- `__ge__` (Goulib.markup.DeprecationError attribute), 187
- `__ge__` (Goulib.markup.InvalidElementError attribute), 186
- `__ge__` (Goulib.markup.MarkupError attribute), 181
- `__ge__` (Goulib.markup.ModeError attribute), 188
- `__ge__` (Goulib.markup.OpeningError attribute), 183
- `__ge__` (Goulib.markup.dummy attribute), 179
- `__ge__` (Goulib.markup.element attribute), 175
- `__ge__` (Goulib.markup.page attribute), 178
- `__ge__` (Goulib.markup.russell attribute), 180
- `__ge__` (Goulib.motion.Actuator attribute), 210
- `__ge__` (Goulib.motion.PVA attribute), 202
- `__ge__` (Goulib.motion.Segment attribute), 204
- `__ge__` (Goulib.motion.SegmentPoly attribute), 207
- `__ge__` (Goulib.motion.Segments attribute), 206
- `__ge__` (Goulib.motion.TimeDiagram attribute), 211
- `__ge__` (Goulib.optim.BinDict attribute), 216
- `__ge__` (Goulib.optim.BinList attribute), 218
- `__ge__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__ge__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__ge__` (Goulib.plot.Plot attribute), 224
- `__ge__` (Goulib.stats.Discrete attribute), 233
- `__ge__` (Goulib.stats.Stats attribute), 232
- `__ge__` (Goulib.table.Cell attribute), 240
- `__ge__` (Goulib.table.Row attribute), 241
- `__ge__` (Goulib.table.Table attribute), 244
- `__ge__` (Goulib.tests.TestCase attribute), 247
- `__ge__` (Goulib.workdays.WorkCalendar attribute), 254
- `__ge__()` (Goulib.expr.Expr method), 61
- `__ge__()` (Goulib.interval.Intervals method), 160
- `__ge__()` (Goulib.piecewise.Piecewise method), 223
- `__ge__()` (Goulib.polynomial.Polynomial method), 226
- `__ge__()` (Goulib.stats.Normal method), 237
- `__ge__()` (Goulib.stats.PDF method), 235
- `__getattr__()` (Goulib.container.Record method), 11
- `__getattr__()` (Goulib.markup.page method), 176

- `__getattrattribute__` (Goulib.colors.Color attribute), 8
- `__getattrattribute__` (Goulib.colors.Palette attribute), 9
- `__getattrattribute__` (Goulib.container.Record attribute), 11
- `__getattrattribute__` (Goulib.container.Sequence attribute), 14
- `__getattrattribute__` (Goulib.datetime2.date2 attribute), 17
- `__getattrattribute__` (Goulib.datetime2.datetime2 attribute), 15
- `__getattrattribute__` (Goulib.datetime2.time2 attribute), 19
- `__getattrattribute__` (Goulib.datetime2.timedelta2 attribute), 21
- `__getattrattribute__` (Goulib.drawing.BBox attribute), 33
- `__getattrattribute__` (Goulib.drawing.Chain attribute), 47
- `__getattrattribute__` (Goulib.drawing.Drawing attribute), 56
- `__getattrattribute__` (Goulib.drawing.Entity attribute), 36
- `__getattrattribute__` (Goulib.drawing.Group attribute), 40
- `__getattrattribute__` (Goulib.drawing.Instance attribute), 44
- `__getattrattribute__` (Goulib.drawing.Rect attribute), 50
- `__getattrattribute__` (Goulib.drawing.Spline attribute), 37
- `__getattrattribute__` (Goulib.drawing.Text attribute), 54
- `__getattrattribute__` (Goulib.expr.Expr attribute), 61
- `__getattrattribute__` (Goulib.expr.TextVisitor attribute), 62
- `__getattrattribute__` (Goulib.geom.Arc2 attribute), 84
- `__getattrattribute__` (Goulib.geom.Circle attribute), 81
- `__getattrattribute__` (Goulib.geom.Ellipse attribute), 87
- `__getattrattribute__` (Goulib.geom.Geometry attribute), 65
- `__getattrattribute__` (Goulib.geom.Line2 attribute), 75
- `__getattrattribute__` (Goulib.geom.Matrix3 attribute), 93
- `__getattrattribute__` (Goulib.geom.Point2 attribute), 71
- `__getattrattribute__` (Goulib.geom.Ray2 attribute), 76
- `__getattrattribute__` (Goulib.geom.Segment2 attribute), 78
- `__getattrattribute__` (Goulib.geom.Vector2 attribute), 70
- `__getattrattribute__` (Goulib.graph.AGraph attribute), 109
- `__getattrattribute__` (Goulib.graph.DiGraph attribute), 127
- `__getattrattribute__` (Goulib.graph.GeoGraph attribute), 111
- `__getattrattribute__` (Goulib.graph.index attribute), 108
- `__getattrattribute__` (Goulib.graph.index.Index attribute), 107
- `__getattrattribute__` (Goulib.graph.index.Property attribute), 106
- `__getattrattribute__` (Goulib.image.Ditherer attribute), 153
- `__getattrattribute__` (Goulib.image.ErrorDiffusion attribute), 153
- `__getattrattribute__` (Goulib.image.FloydSteinberg attribute), 154
- `__getattrattribute__` (Goulib.image.Image attribute), 151
- `__getattrattribute__` (Goulib.image.Mode attribute), 145
- `__getattrattribute__` (Goulib.interval.Box attribute), 163
- `__getattrattribute__` (Goulib.interval.Interval attribute), 158
- `__getattrattribute__` (Goulib.interval.Intervals attribute), 160
- `__getattrattribute__` (Goulib.itertools2.SortingError attribute), 171
- `__getattrattribute__` (Goulib.itertools2.iter2 attribute), 170
- `__getattrattribute__` (Goulib.itertools2.keep attribute), 173
- `__getattrattribute__` (Goulib.markup.ArgumentError attribute), 184
- `__getattrattribute__` (Goulib.markup.ClosingError attribute), 182
- `__getattrattribute__` (Goulib.markup.CustomizationError attribute), 189
- `__getattrattribute__` (Goulib.markup.DeprecationError attribute), 187
- `__getattrattribute__` (Goulib.markup.InvalidElementError attribute), 186
- `__getattrattribute__` (Goulib.markup.MarkupError attribute), 181
- `__getattrattribute__` (Goulib.markup.ModeError attribute), 188
- `__getattrattribute__` (Goulib.markup.OpeningError attribute), 183
- `__getattrattribute__` (Goulib.markup.dummy attribute), 179
- `__getattrattribute__` (Goulib.markup.element attribute), 175
- `__getattrattribute__` (Goulib.markup.page attribute), 178
- `__getattrattribute__` (Goulib.markup.russell attribute), 180
- `__getattrattribute__` (Goulib.motion.Actuator attribute), 210
- `__getattrattribute__` (Goulib.motion.PVA attribute), 202
- `__getattrattribute__` (Goulib.motion.Segment attribute), 204
- `__getattrattribute__` (Goulib.motion.SegmentPoly attribute), 207
- `__getattrattribute__` (Goulib.motion.Segments attribute), 206
- `__getattrattribute__` (Goulib.motion.TimeDiagram attribute), 211
- `__getattrattribute__` (Goulib.optim.BinDict attribute), 216
- `__getattrattribute__` (Goulib.optim.BinList attribute), 218
- `__getattrattribute__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__getattrattribute__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__getattrattribute__` (Goulib.piecewise.Piecewise attribute), 223
- `__getattrattribute__` (Goulib.plot.Plot attribute), 225
- `__getattrattribute__` (Goulib.polynomial.Polynomial attribute), 227
- `__getattrattribute__` (Goulib.stats.Discrete attribute), 233
- `__getattrattribute__` (Goulib.stats.Normal attribute), 237
- `__getattrattribute__` (Goulib.stats.PDF attribute), 235
- `__getattrattribute__` (Goulib.stats.Stats attribute), 232
- `__getattrattribute__` (Goulib.table.Cell attribute), 240
- `__getattrattribute__` (Goulib.table.Row attribute), 241
- `__getattrattribute__` (Goulib.table.Table attribute), 244
- `__getattrattribute__` (Goulib.tests.TestCase attribute), 247
- `__getattrattribute__` (Goulib.workdays.WorkCalendar attribute), 254
- `__getitem__` () (Goulib.colors.Palette method), 9
- `__getitem__` () (Goulib.container.Record method), 11
- `__getitem__` () (Goulib.container.Sequence method), 13
- `__getitem__` () (Goulib.drawing.BBox method), 33
- `__getitem__` () (Goulib.drawing.Chain method), 47

- `__getitem__()` (Goulib.drawing.Drawing method), 56
- `__getitem__()` (Goulib.drawing.Group method), 40
- `__getitem__()` (Goulib.drawing.Rect method), 50
- `__getitem__()` (Goulib.geom.Matrix3 method), 93
- `__getitem__()` (Goulib.geom3d.Matrix4 method), 99
- `__getitem__()` (Goulib.graph.DiGraph method), 127
- `__getitem__()` (Goulib.graph.GeoGraph method), 111
- `__getitem__()` (Goulib.graph.index.Index method), 107
- `__getitem__()` (Goulib.image.Image method), 148
- `__getitem__()` (Goulib.interval.Box method), 164
- `__getitem__()` (Goulib.interval.Interval method), 158
- `__getitem__()` (Goulib.interval.Intervals method), 160
- `__getitem__()` (Goulib.optim.BinDict method), 216
- `__getitem__()` (Goulib.optim.BinList method), 218
- `__getitem__()` (Goulib.piecewise.Piecewise method), 222
- `__getitem__()` (Goulib.table.Table method), 243
- `__getstate__()` (Goulib.graph.DiGraph method), 127
- `__getstate__()` (Goulib.graph.GeoGraph method), 111
- `__gt__` (Goulib.colors.Color attribute), 8
- `__gt__` (Goulib.colors.Palette attribute), 9
- `__gt__` (Goulib.container.Record attribute), 11
- `__gt__` (Goulib.container.Sequence attribute), 14
- `__gt__` (Goulib.datetime2.date2 attribute), 18
- `__gt__` (Goulib.datetime2.datetime2 attribute), 15
- `__gt__` (Goulib.datetime2.time2 attribute), 19
- `__gt__` (Goulib.datetime2.timedelta2 attribute), 21
- `__gt__` (Goulib.drawing.BBox attribute), 33
- `__gt__` (Goulib.drawing.Chain attribute), 47
- `__gt__` (Goulib.drawing.Drawing attribute), 56
- `__gt__` (Goulib.drawing.Entity attribute), 36
- `__gt__` (Goulib.drawing.Group attribute), 40
- `__gt__` (Goulib.drawing.Instance attribute), 44
- `__gt__` (Goulib.drawing.Rect attribute), 50
- `__gt__` (Goulib.drawing.Spline attribute), 37
- `__gt__` (Goulib.drawing.Text attribute), 54
- `__gt__` (Goulib.expr.TextVisitor attribute), 62
- `__gt__` (Goulib.geom.Arc2 attribute), 84
- `__gt__` (Goulib.geom.Circle attribute), 81
- `__gt__` (Goulib.geom.Ellipse attribute), 87
- `__gt__` (Goulib.geom.Geometry attribute), 65
- `__gt__` (Goulib.geom.Line2 attribute), 75
- `__gt__` (Goulib.geom.Matrix3 attribute), 93
- `__gt__` (Goulib.geom.Point2 attribute), 71
- `__gt__` (Goulib.geom.Ray2 attribute), 76
- `__gt__` (Goulib.geom.Segment2 attribute), 78
- `__gt__` (Goulib.geom.Vector2 attribute), 70
- `__gt__` (Goulib.graph.AGraph attribute), 110
- `__gt__` (Goulib.graph.DiGraph attribute), 127
- `__gt__` (Goulib.graph.GeoGraph attribute), 111
- `__gt__` (Goulib.graph.index attribute), 109
- `__gt__` (Goulib.graph.index.Index attribute), 107
- `__gt__` (Goulib.graph.index.Property attribute), 106
- `__gt__` (Goulib.image.Ditherer attribute), 153
- `__gt__` (Goulib.image.ErrorDiffusion attribute), 154
- `__gt__` (Goulib.image.FloydSteinberg attribute), 154
- `__gt__` (Goulib.image.Image attribute), 151
- `__gt__` (Goulib.image.Mode attribute), 146
- `__gt__` (Goulib.interval.Box attribute), 164
- `__gt__` (Goulib.interval.Interval attribute), 158
- `__gt__` (Goulib.itertools2.SortingError attribute), 171
- `__gt__` (Goulib.itertools2.iter2 attribute), 170
- `__gt__` (Goulib.itertools2.keep attribute), 173
- `__gt__` (Goulib.markup.ArgumentError attribute), 185
- `__gt__` (Goulib.markup.ClosingError attribute), 182
- `__gt__` (Goulib.markup.CustomizationError attribute), 189
- `__gt__` (Goulib.markup.DeprecationError attribute), 187
- `__gt__` (Goulib.markup.InvalidElementError attribute), 186
- `__gt__` (Goulib.markup.MarkupError attribute), 181
- `__gt__` (Goulib.markup.ModeError attribute), 188
- `__gt__` (Goulib.markup.OpeningError attribute), 183
- `__gt__` (Goulib.markup.dummy attribute), 179
- `__gt__` (Goulib.markup.element attribute), 175
- `__gt__` (Goulib.markup.page attribute), 178
- `__gt__` (Goulib.markup.russell attribute), 180
- `__gt__` (Goulib.motion.Actuator attribute), 210
- `__gt__` (Goulib.motion.PVA attribute), 203
- `__gt__` (Goulib.motion.Segment attribute), 204
- `__gt__` (Goulib.motion.SegmentPoly attribute), 207
- `__gt__` (Goulib.motion.Segments attribute), 206
- `__gt__` (Goulib.motion.TimeDiagram attribute), 211
- `__gt__` (Goulib.optim.BinDict attribute), 216
- `__gt__` (Goulib.optim.BinList attribute), 218
- `__gt__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__gt__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__gt__` (Goulib.plot.Plot attribute), 225
- `__gt__` (Goulib.stats.Discrete attribute), 233
- `__gt__` (Goulib.stats.Stats attribute), 232
- `__gt__` (Goulib.table.Cell attribute), 240
- `__gt__` (Goulib.table.Row attribute), 242
- `__gt__` (Goulib.table.Table attribute), 244
- `__gt__` (Goulib.tests.TestCase attribute), 247
- `__gt__` (Goulib.workdays.WorkCalendar attribute), 254
- `__gt__()` (Goulib.expr.Expr method), 61
- `__gt__()` (Goulib.interval.Intervals method), 160
- `__gt__()` (Goulib.piecewise.Piecewise method), 223
- `__gt__()` (Goulib.polynomial.Polynomial method), 227
- `__gt__()` (Goulib.stats.Normal method), 237
- `__gt__()` (Goulib.stats.PDF method), 235
- `__hash__` (Goulib.colors.Palette attribute), 9
- `__hash__` (Goulib.container.Record attribute), 11
- `__hash__` (Goulib.container.Sequence attribute), 14
- `__hash__` (Goulib.datetime2.date2 attribute), 18
- `__hash__` (Goulib.datetime2.datetime2 attribute), 15
- `__hash__` (Goulib.datetime2.time2 attribute), 19
- `__hash__` (Goulib.datetime2.timedelta2 attribute), 21

`__hash__` (Goulib.drawing.BBox attribute), 33
`__hash__` (Goulib.drawing.Chain attribute), 47
`__hash__` (Goulib.drawing.Drawing attribute), 56
`__hash__` (Goulib.drawing.Entity attribute), 36
`__hash__` (Goulib.drawing.Group attribute), 40
`__hash__` (Goulib.drawing.Instance attribute), 44
`__hash__` (Goulib.drawing.Rect attribute), 50
`__hash__` (Goulib.drawing.Spline attribute), 37
`__hash__` (Goulib.drawing.Text attribute), 54
`__hash__` (Goulib.expr.Expr attribute), 62
`__hash__` (Goulib.expr.TextVisitor attribute), 63
`__hash__` (Goulib.geom.Arc2 attribute), 84
`__hash__` (Goulib.geom.Circle attribute), 81
`__hash__` (Goulib.geom.Ellipse attribute), 87
`__hash__` (Goulib.geom.Geometry attribute), 65
`__hash__` (Goulib.geom.Line2 attribute), 75
`__hash__` (Goulib.geom.Matrix3 attribute), 93
`__hash__` (Goulib.geom.Ray2 attribute), 76
`__hash__` (Goulib.geom.Segment2 attribute), 78
`__hash__` (Goulib.geom3d.Vector3 attribute), 95
`__hash__` (Goulib.graph.AGraph attribute), 110
`__hash__` (Goulib.graph.DiGraph attribute), 127
`__hash__` (Goulib.graph.GeoGraph attribute), 111
`__hash__` (Goulib.graph.index attribute), 109
`__hash__` (Goulib.graph.index.Index attribute), 107
`__hash__` (Goulib.graph.index.Property attribute), 106
`__hash__` (Goulib.image.Ditherer attribute), 153
`__hash__` (Goulib.image.ErrorDiffusion attribute), 154
`__hash__` (Goulib.image.FloydSteinberg attribute), 155
`__hash__` (Goulib.image.Mode attribute), 146
`__hash__` (Goulib.interval.Box attribute), 164
`__hash__` (Goulib.interval.Intervals attribute), 160
`__hash__` (Goulib.itertools2.SortingError attribute), 171
`__hash__` (Goulib.itertools2.iter2 attribute), 170
`__hash__` (Goulib.itertools2.keep attribute), 173
`__hash__` (Goulib.markup.ArgumentError attribute), 185
`__hash__` (Goulib.markup.ClosingError attribute), 182
`__hash__` (Goulib.markup.CustomizationError attribute), 189
`__hash__` (Goulib.markup.DeprecationError attribute), 187
`__hash__` (Goulib.markup.InvalidElementError attribute), 186
`__hash__` (Goulib.markup.MarkupError attribute), 181
`__hash__` (Goulib.markup.ModeError attribute), 188
`__hash__` (Goulib.markup.OpeningError attribute), 183
`__hash__` (Goulib.markup.dummy attribute), 179
`__hash__` (Goulib.markup.element attribute), 175
`__hash__` (Goulib.markup.page attribute), 178
`__hash__` (Goulib.markup.russell attribute), 180
`__hash__` (Goulib.motion.Actuator attribute), 210
`__hash__` (Goulib.motion.PVA attribute), 203
`__hash__` (Goulib.motion.Segment attribute), 204
`__hash__` (Goulib.motion.SegmentPoly attribute), 207
`__hash__` (Goulib.motion.Segments attribute), 206
`__hash__` (Goulib.motion.TimeDiagram attribute), 211
`__hash__` (Goulib.optim.BinDict attribute), 216
`__hash__` (Goulib.optim.BinList attribute), 218
`__hash__` (Goulib.optim.DifferentialEvolution attribute), 221
`__hash__` (Goulib.optim.ObjectiveFunction attribute), 214
`__hash__` (Goulib.piecewise.Piecewise attribute), 223
`__hash__` (Goulib.plot.Plot attribute), 225
`__hash__` (Goulib.polynomial.Polynomial attribute), 227
`__hash__` (Goulib.stats.Discrete attribute), 233
`__hash__` (Goulib.stats.Normal attribute), 237
`__hash__` (Goulib.stats.PDF attribute), 235
`__hash__` (Goulib.stats.Stats attribute), 232
`__hash__` (Goulib.table.Cell attribute), 240
`__hash__` (Goulib.table.Row attribute), 242
`__hash__` (Goulib.table.Table attribute), 244
`__hash__` (Goulib.workdays.WorkCalendar attribute), 254
`__hash__()` (Goulib.colors.Color method), 7
`__hash__()` (Goulib.geom.Point2 method), 71
`__hash__()` (Goulib.geom.Vector2 method), 68
`__hash__()` (Goulib.image.Image method), 149
`__hash__()` (Goulib.interval.Interval method), 157
`__hash__()` (Goulib.tests.TestCase method), 247
`__iadd__` (Goulib.drawing.Chain attribute), 47
`__iadd__` (Goulib.drawing.Drawing attribute), 56
`__iadd__` (Goulib.drawing.Group attribute), 40
`__iadd__` (Goulib.drawing.Rect attribute), 50
`__iadd__` (Goulib.table.Table attribute), 244
`__iadd__()` (Goulib.drawing.BBox method), 25, 32
`__iadd__()` (Goulib.geom.Point2 method), 71
`__iadd__()` (Goulib.geom.Vector2 method), 69
`__iadd__()` (Goulib.geom3d.Vector3 method), 95
`__iadd__()` (Goulib.interval.Box method), 163
`__iadd__()` (Goulib.interval.Interval method), 157
`__iadd__()` (Goulib.interval.Intervals method), 159
`__iadd__()` (Goulib.optim.BinDict method), 215
`__iadd__()` (Goulib.optim.BinList method), 217
`__imul__` (Goulib.drawing.BBox attribute), 33
`__imul__` (Goulib.drawing.Chain attribute), 47
`__imul__` (Goulib.drawing.Drawing attribute), 56
`__imul__` (Goulib.drawing.Group attribute), 40
`__imul__` (Goulib.drawing.Rect attribute), 50
`__imul__` (Goulib.interval.Box attribute), 164
`__imul__` (Goulib.interval.Interval attribute), 158
`__imul__` (Goulib.optim.BinList attribute), 218
`__imul__` (Goulib.table.Table attribute), 244
`__imul__()` (Goulib.geom.Matrix3 method), 93
`__imul__()` (Goulib.geom.Point2 method), 71
`__imul__()` (Goulib.geom.Vector2 method), 69
`__imul__()` (Goulib.geom3d.Matrix4 method), 99
`__imul__()` (Goulib.geom3d.Quaternion method), 102

- `__imul__()` (Goulib.geom3d.Vector3 method), 95
- `__imul__()` (Goulib.interval.Intervals method), 160
- `__init__` (Goulib.datetime2.date2 attribute), 18
- `__init__` (Goulib.drawing.Entity attribute), 36
- `__init__` (Goulib.drawing.Group attribute), 40
- `__init__` (Goulib.graph.AGraph attribute), 110
- `__init__` (Goulib.graph.index attribute), 109
- `__init__` (Goulib.graph.index.Property attribute), 106
- `__init__` (Goulib.itertools2.SortingError attribute), 171
- `__init__` (Goulib.markup.MarkupError attribute), 181
- `__init__` (Goulib.markup.dummy attribute), 179
- `__init__` (Goulib.markup.russell attribute), 180
- `__init__` (Goulib.plot.Plot attribute), 225
- `__init__()` (Goulib.colors.Color method), 6
- `__init__()` (Goulib.colors.Palette method), 8
- `__init__()` (Goulib.container.Record method), 11
- `__init__()` (Goulib.container.Sequence method), 13
- `__init__()` (Goulib.datetime2.datetime2 method), 14
- `__init__()` (Goulib.datetime2.time2 method), 19
- `__init__()` (Goulib.datetime2.timedelta2 method), 20
- `__init__()` (Goulib.drawing.BBox method), 25, 31
- `__init__()` (Goulib.drawing.Chain method), 28, 46
- `__init__()` (Goulib.drawing.Drawing method), 30, 56
- `__init__()` (Goulib.drawing.Instance method), 28, 43
- `__init__()` (Goulib.drawing.Rect method), 29, 49
- `__init__()` (Goulib.drawing.Spline method), 27, 36
- `__init__()` (Goulib.drawing.Text method), 30, 53
- `__init__()` (Goulib.expr.Expr method), 60
- `__init__()` (Goulib.expr.TextVisitor method), 62
- `__init__()` (Goulib.geom.Arc2 method), 83
- `__init__()` (Goulib.geom.Circle method), 80
- `__init__()` (Goulib.geom.Ellipse method), 86
- `__init__()` (Goulib.geom.Geometry method), 64
- `__init__()` (Goulib.geom.Line2 method), 75
- `__init__()` (Goulib.geom.Matrix3 method), 92
- `__init__()` (Goulib.geom.Point2 method), 71
- `__init__()` (Goulib.geom.Ray2 method), 76
- `__init__()` (Goulib.geom.Segment2 method), 78
- `__init__()` (Goulib.geom.Vector2 method), 68
- `__init__()` (Goulib.geom3d.Line3 method), 97
- `__init__()` (Goulib.geom3d.Matrix4 method), 99
- `__init__()` (Goulib.geom3d.Plane method), 99
- `__init__()` (Goulib.geom3d.Quaternion method), 102
- `__init__()` (Goulib.geom3d.Sphere method), 98
- `__init__()` (Goulib.geom3d.Vector3 method), 94
- `__init__()` (Goulib.graph.DiGraph method), 104, 126
- `__init__()` (Goulib.graph.GeoGraph method), 104, 110
- `__init__()` (Goulib.graph.index.Index method), 103, 106
- `__init__()` (Goulib.image.Ditherer method), 152
- `__init__()` (Goulib.image.ErrorDiffusion method), 153
- `__init__()` (Goulib.image.FloydSteinberg method), 154
- `__init__()` (Goulib.image.Image method), 146
- `__init__()` (Goulib.image.Mode method), 145
- `__init__()` (Goulib.interval.Box method), 163
- `__init__()` (Goulib.interval.Interval method), 156
- `__init__()` (Goulib.interval.Intervals method), 160
- `__init__()` (Goulib.itertools2.iter2 method), 169
- `__init__()` (Goulib.itertools2.keep method), 172
- `__init__()` (Goulib.markup.ArgumentError method), 184
- `__init__()` (Goulib.markup.ClosingError method), 182
- `__init__()` (Goulib.markup.CustomizationError method), 190
- `__init__()` (Goulib.markup.DeprecationError method), 188
- `__init__()` (Goulib.markup.InvalidElementError method), 186
- `__init__()` (Goulib.markup.ModeError method), 189
- `__init__()` (Goulib.markup.OpeningError method), 183
- `__init__()` (Goulib.markup.element method), 174
- `__init__()` (Goulib.markup.page method), 176
- `__init__()` (Goulib.motion.Actuator method), 209
- `__init__()` (Goulib.motion.PVA method), 202
- `__init__()` (Goulib.motion.Segment method), 203
- `__init__()` (Goulib.motion.SegmentPoly method), 207
- `__init__()` (Goulib.motion.Segments method), 205
- `__init__()` (Goulib.motion.TimeDiagram method), 211
- `__init__()` (Goulib.optim.BinDict method), 216
- `__init__()` (Goulib.optim.BinList method), 218
- `__init__()` (Goulib.optim.DifferentialEvolution method), 221
- `__init__()` (Goulib.optim.ObjectiveFunction method), 214
- `__init__()` (Goulib.piecewise.Piecewise method), 222
- `__init__()` (Goulib.polynomial.Polynomial method), 226
- `__init__()` (Goulib.statemachine.Simulation method), 229
- `__init__()` (Goulib.statemachine.StateChangeLog method), 229
- `__init__()` (Goulib.statemachine.StateMachine method), 229
- `__init__()` (Goulib.statemachine.TimeMarker method), 229
- `__init__()` (Goulib.statemachine.TooLateLog method), 229
- `__init__()` (Goulib.statemachine.WaitLog method), 229
- `__init__()` (Goulib.stats.Discrete method), 232
- `__init__()` (Goulib.stats.Normal method), 236
- `__init__()` (Goulib.stats.PDF method), 234
- `__init__()` (Goulib.stats.Stats method), 231
- `__init__()` (Goulib.table.Cell method), 239
- `__init__()` (Goulib.table.Row method), 241
- `__init__()` (Goulib.table.Table method), 242
- `__init__()` (Goulib.tests.TestCase method), 247
- `__init__()` (Goulib.workdays.WorkCalendar method), 253
- `__inv__()` (Goulib.image.Image method), 149
- `__invert__()` (Goulib.expr.Expr method), 61
- `__invert__()` (Goulib.piecewise.Piecewise method), 223
- `__invert__()` (Goulib.polynomial.Polynomial method), 227
- `__invert__()` (Goulib.stats.Normal method), 237

`__invert__()` (Goulib.stats.PDF method), 235
`__isub__()` (Goulib.optim.BinDict method), 215
`__isub__()` (Goulib.optim.BinList method), 217
`__iter__` (Goulib.colors.Palette attribute), 9
`__iter__` (Goulib.container.Record attribute), 11
`__iter__` (Goulib.drawing.BBox attribute), 33
`__iter__` (Goulib.drawing.Chain attribute), 47
`__iter__` (Goulib.drawing.Drawing attribute), 56
`__iter__` (Goulib.drawing.Group attribute), 40
`__iter__` (Goulib.drawing.Rect attribute), 50
`__iter__` (Goulib.graph.index.Index attribute), 107
`__iter__` (Goulib.interval.Box attribute), 164
`__iter__` (Goulib.interval.Interval attribute), 158
`__iter__` (Goulib.optim.BinDict attribute), 216
`__iter__` (Goulib.optim.BinList attribute), 218
`__iter__` (Goulib.table.Table attribute), 244
`__iter__()` (Goulib.container.Sequence method), 13
`__iter__()` (Goulib.drawing.Instance method), 28, 43
`__iter__()` (Goulib.geom.Matrix3 method), 92
`__iter__()` (Goulib.geom.Point2 method), 71
`__iter__()` (Goulib.geom.Vector2 method), 68
`__iter__()` (Goulib.geom3d.Matrix4 method), 99
`__iter__()` (Goulib.geom3d.Vector3 method), 95
`__iter__()` (Goulib.graph.DiGraph method), 127
`__iter__()` (Goulib.graph.GeoGraph method), 111
`__iter__()` (Goulib.interval.Intervals method), 160
`__iter__()` (Goulib.itertools2.iter2 method), 170
`__iter__()` (Goulib.itertools2.keep method), 173
`__iter__()` (Goulib.piecewise.Piecewise method), 222
`__le__` (Goulib.colors.Color attribute), 8
`__le__` (Goulib.colors.Palette attribute), 9
`__le__` (Goulib.container.Record attribute), 11
`__le__` (Goulib.container.Sequence attribute), 14
`__le__` (Goulib.datetime2.date2 attribute), 18
`__le__` (Goulib.datetime2.datetime2 attribute), 15
`__le__` (Goulib.datetime2.time2 attribute), 19
`__le__` (Goulib.datetime2.timedelta2 attribute), 21
`__le__` (Goulib.drawing.BBox attribute), 33
`__le__` (Goulib.drawing.Chain attribute), 47
`__le__` (Goulib.drawing.Drawing attribute), 57
`__le__` (Goulib.drawing.Entity attribute), 36
`__le__` (Goulib.drawing.Group attribute), 40
`__le__` (Goulib.drawing.Instance attribute), 44
`__le__` (Goulib.drawing.Rect attribute), 50
`__le__` (Goulib.drawing.Spline attribute), 37
`__le__` (Goulib.drawing.Text attribute), 54
`__le__` (Goulib.expr.TextVisitor attribute), 63
`__le__` (Goulib.geom.Arc2 attribute), 84
`__le__` (Goulib.geom.Circle attribute), 81
`__le__` (Goulib.geom.Ellipse attribute), 87
`__le__` (Goulib.geom.Geometry attribute), 65
`__le__` (Goulib.geom.Line2 attribute), 75
`__le__` (Goulib.geom.Matrix3 attribute), 93
`__le__` (Goulib.geom.Point2 attribute), 72
`__le__` (Goulib.geom.Ray2 attribute), 76
`__le__` (Goulib.geom.Segment2 attribute), 78
`__le__` (Goulib.geom.Vector2 attribute), 70
`__le__` (Goulib.graph.AGraph attribute), 110
`__le__` (Goulib.graph.DiGraph attribute), 127
`__le__` (Goulib.graph.GeoGraph attribute), 112
`__le__` (Goulib.graph.index attribute), 109
`__le__` (Goulib.graph.index.Index attribute), 107
`__le__` (Goulib.graph.index.Property attribute), 106
`__le__` (Goulib.image.Ditherer attribute), 153
`__le__` (Goulib.image.ErrorDiffusion attribute), 154
`__le__` (Goulib.image.FloydSteinberg attribute), 155
`__le__` (Goulib.image.Image attribute), 151
`__le__` (Goulib.image.Mode attribute), 146
`__le__` (Goulib.interval.Box attribute), 164
`__le__` (Goulib.interval.Interval attribute), 158
`__le__` (Goulib.itertools2.SortingError attribute), 171
`__le__` (Goulib.itertools2.iter2 attribute), 170
`__le__` (Goulib.itertools2.keep attribute), 173
`__le__` (Goulib.markup.ArgumentError attribute), 185
`__le__` (Goulib.markup.ClosingError attribute), 182
`__le__` (Goulib.markup.CustomizationError attribute), 189
`__le__` (Goulib.markup.DeprecationError attribute), 187
`__le__` (Goulib.markup.InvalidElementError attribute), 186
`__le__` (Goulib.markup.MarkupError attribute), 181
`__le__` (Goulib.markup.ModeError attribute), 188
`__le__` (Goulib.markup.OpeningError attribute), 183
`__le__` (Goulib.markup.dummy attribute), 179
`__le__` (Goulib.markup.element attribute), 175
`__le__` (Goulib.markup.page attribute), 178
`__le__` (Goulib.markup.russell attribute), 180
`__le__` (Goulib.motion.Actuator attribute), 210
`__le__` (Goulib.motion.PVA attribute), 203
`__le__` (Goulib.motion.Segment attribute), 204
`__le__` (Goulib.motion.SegmentPoly attribute), 207
`__le__` (Goulib.motion.Segments attribute), 206
`__le__` (Goulib.motion.TimeDiagram attribute), 212
`__le__` (Goulib.optim.BinDict attribute), 216
`__le__` (Goulib.optim.BinList attribute), 218
`__le__` (Goulib.optim.DifferentialEvolution attribute), 221
`__le__` (Goulib.optim.ObjectiveFunction attribute), 214
`__le__` (Goulib.plot.Plot attribute), 225
`__le__` (Goulib.stats.Discrete attribute), 233
`__le__` (Goulib.stats.Stats attribute), 232
`__le__` (Goulib.table.Cell attribute), 240
`__le__` (Goulib.table.Row attribute), 242
`__le__` (Goulib.table.Table attribute), 244
`__le__` (Goulib.tests.TestCase attribute), 247
`__le__` (Goulib.workdays.WorkCalendar attribute), 254
`__le__()` (Goulib.expr.Expr method), 61
`__le__()` (Goulib.interval.Intervals method), 160

- `__le__()` (Goulib.piecewise.Piecewise method), 223
- `__le__()` (Goulib.polynomial.Polynomial method), 227
- `__le__()` (Goulib.stats.Normal method), 237
- `__le__()` (Goulib.stats.PDF method), 235
- `__len__` (Goulib.colors.Palette attribute), 9
- `__len__` (Goulib.container.Record attribute), 11
- `__len__` (Goulib.drawing.BBox attribute), 33
- `__len__` (Goulib.drawing.Chain attribute), 47
- `__len__` (Goulib.drawing.Drawing attribute), 57
- `__len__` (Goulib.drawing.Group attribute), 40
- `__len__` (Goulib.drawing.Rect attribute), 50
- `__len__` (Goulib.graph.index.Index attribute), 107
- `__len__` (Goulib.interval.Box attribute), 164
- `__len__` (Goulib.interval.Interval attribute), 158
- `__len__` (Goulib.optim.BinDict attribute), 216
- `__len__` (Goulib.optim.BinList attribute), 218
- `__len__` (Goulib.table.Table attribute), 244
- `__len__()` (Goulib.geom.Point2 method), 72
- `__len__()` (Goulib.geom.Vector2 method), 68
- `__len__()` (Goulib.geom3d.Vector3 method), 95
- `__len__()` (Goulib.graph.DiGraph method), 127
- `__len__()` (Goulib.graph.GeoGraph method), 112
- `__len__()` (Goulib.interval.Intervals method), 160
- `__len__()` (Goulib.piecewise.Piecewise method), 222
- `__lshift__()` (Goulib.expr.Expr method), 61
- `__lshift__()` (Goulib.piecewise.Piecewise method), 222
- `__lshift__()` (Goulib.polynomial.Polynomial method), 227
- `__lshift__()` (Goulib.stats.Normal method), 238
- `__lshift__()` (Goulib.stats.PDF method), 235
- `__lt__` (Goulib.colors.Color attribute), 8
- `__lt__` (Goulib.colors.Palette attribute), 9
- `__lt__` (Goulib.container.Record attribute), 12
- `__lt__` (Goulib.container.Sequence attribute), 14
- `__lt__` (Goulib.datetime2.date2 attribute), 18
- `__lt__` (Goulib.datetime2.datetime2 attribute), 15
- `__lt__` (Goulib.datetime2.time2 attribute), 20
- `__lt__` (Goulib.datetime2.timedelta2 attribute), 21
- `__lt__` (Goulib.drawing.BBox attribute), 33
- `__lt__` (Goulib.drawing.Chain attribute), 47
- `__lt__` (Goulib.drawing.Drawing attribute), 57
- `__lt__` (Goulib.drawing.Entity attribute), 36
- `__lt__` (Goulib.drawing.Group attribute), 41
- `__lt__` (Goulib.drawing.Instance attribute), 44
- `__lt__` (Goulib.drawing.Rect attribute), 50
- `__lt__` (Goulib.drawing.Spline attribute), 37
- `__lt__` (Goulib.drawing.Text attribute), 54
- `__lt__` (Goulib.expr.TextVisitor attribute), 63
- `__lt__` (Goulib.geom.Arc2 attribute), 84
- `__lt__` (Goulib.geom.Circle attribute), 81
- `__lt__` (Goulib.geom.Ellipse attribute), 87
- `__lt__` (Goulib.geom.Geometry attribute), 65
- `__lt__` (Goulib.geom.Line2 attribute), 76
- `__lt__` (Goulib.geom.Matrix3 attribute), 93
- `__lt__` (Goulib.geom.Point2 attribute), 72
- `__lt__` (Goulib.geom.Ray2 attribute), 77
- `__lt__` (Goulib.geom.Segment2 attribute), 78
- `__lt__` (Goulib.geom.Vector2 attribute), 70
- `__lt__` (Goulib.graph.AGraph attribute), 110
- `__lt__` (Goulib.graph.DiGraph attribute), 127
- `__lt__` (Goulib.graph.GeoGraph attribute), 112
- `__lt__` (Goulib.graph.index attribute), 109
- `__lt__` (Goulib.graph.index.Index attribute), 107
- `__lt__` (Goulib.graph.index.Property attribute), 106
- `__lt__` (Goulib.image.Ditherer attribute), 153
- `__lt__` (Goulib.image.ErrorDiffusion attribute), 154
- `__lt__` (Goulib.image.FloydSteinberg attribute), 155
- `__lt__` (Goulib.image.Mode attribute), 146
- `__lt__` (Goulib.interval.Box attribute), 164
- `__lt__` (Goulib.itertools2.SortingError attribute), 171
- `__lt__` (Goulib.itertools2.iter2 attribute), 170
- `__lt__` (Goulib.itertools2.keep attribute), 173
- `__lt__` (Goulib.markup.ArgumentError attribute), 185
- `__lt__` (Goulib.markup.ClosingError attribute), 182
- `__lt__` (Goulib.markup.CustomizationError attribute), 189
- `__lt__` (Goulib.markup.DeprecationError attribute), 187
- `__lt__` (Goulib.markup.InvalidElementError attribute), 186
- `__lt__` (Goulib.markup.MarkupError attribute), 181
- `__lt__` (Goulib.markup.ModeError attribute), 188
- `__lt__` (Goulib.markup.OpeningError attribute), 183
- `__lt__` (Goulib.markup.dummy attribute), 179
- `__lt__` (Goulib.markup.element attribute), 175
- `__lt__` (Goulib.markup.page attribute), 178
- `__lt__` (Goulib.markup.russell attribute), 180
- `__lt__` (Goulib.motion.Actuator attribute), 210
- `__lt__` (Goulib.motion.PVA attribute), 203
- `__lt__` (Goulib.motion.Segment attribute), 204
- `__lt__` (Goulib.motion.SegmentPoly attribute), 207
- `__lt__` (Goulib.motion.Segments attribute), 206
- `__lt__` (Goulib.motion.TimeDiagram attribute), 212
- `__lt__` (Goulib.optim.BinDict attribute), 216
- `__lt__` (Goulib.optim.BinList attribute), 218
- `__lt__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__lt__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__lt__` (Goulib.plot.Plot attribute), 225
- `__lt__` (Goulib.stats.Discrete attribute), 233
- `__lt__` (Goulib.stats.Stats attribute), 232
- `__lt__` (Goulib.table.Cell attribute), 240
- `__lt__` (Goulib.table.Row attribute), 242
- `__lt__` (Goulib.table.Table attribute), 244
- `__lt__` (Goulib.tests.TestCase attribute), 247
- `__lt__` (Goulib.workdays.WorkCalendar attribute), 254
- `__lt__()` (Goulib.expr.Expr method), 61
- `__lt__()` (Goulib.image.Image method), 147
- `__lt__()` (Goulib.interval.Interval method), 157

`__lt__` (Goulib.interval.Intervals method), 160
`__lt__` (Goulib.piecewise.Piecewise method), 223
`__lt__` (Goulib.polynomial.Polynomial method), 226
`__lt__` (Goulib.stats.Normal method), 238
`__lt__` (Goulib.stats.PDF method), 235
`__mod__` (Goulib.datetime2.timedelta2 attribute), 21
`__mod__` (Goulib.container.Sequence method), 13
`__mul__` (Goulib.datetime2.timedelta2 attribute), 21
`__mul__` (Goulib.drawing.BBox attribute), 33
`__mul__` (Goulib.drawing.Chain attribute), 47
`__mul__` (Goulib.drawing.Drawing attribute), 57
`__mul__` (Goulib.drawing.Group attribute), 41
`__mul__` (Goulib.drawing.Rect attribute), 51
`__mul__` (Goulib.interval.Box attribute), 164
`__mul__` (Goulib.interval.Interval attribute), 158
`__mul__` (Goulib.optim.BinList attribute), 218
`__mul__` (Goulib.table.Table attribute), 244
`__mul__` (Goulib.colors.Color method), 7
`__mul__` (Goulib.expr.Expr method), 61
`__mul__` (Goulib.geom.Matrix3 method), 93
`__mul__` (Goulib.geom.Point2 method), 72
`__mul__` (Goulib.geom.Vector2 method), 69
`__mul__` (Goulib.geom3d.Matrix4 method), 99
`__mul__` (Goulib.geom3d.Quaternion method), 102
`__mul__` (Goulib.geom3d.Vector3 method), 95
`__mul__` (Goulib.image.Image method), 150
`__mul__` (Goulib.interval.Intervals method), 160
`__mul__` (Goulib.piecewise.Piecewise method), 223
`__mul__` (Goulib.polynomial.Polynomial method), 226
`__mul__` (Goulib.stats.Discrete method), 233
`__mul__` (Goulib.stats.Normal method), 237
`__mul__` (Goulib.stats.PDF method), 235
`__mul__` (Goulib.stats.Stats method), 231
`__ne__` (Goulib.colors.Color attribute), 8
`__ne__` (Goulib.colors.Palette attribute), 9
`__ne__` (Goulib.container.Record attribute), 12
`__ne__` (Goulib.container.Sequence attribute), 14
`__ne__` (Goulib.datetime2.date2 attribute), 18
`__ne__` (Goulib.datetime2.datetime2 attribute), 15
`__ne__` (Goulib.datetime2.time2 attribute), 20
`__ne__` (Goulib.datetime2.timedelta2 attribute), 21
`__ne__` (Goulib.drawing.BBox attribute), 33
`__ne__` (Goulib.drawing.Chain attribute), 47
`__ne__` (Goulib.drawing.Drawing attribute), 57
`__ne__` (Goulib.drawing.Entity attribute), 36
`__ne__` (Goulib.drawing.Group attribute), 41
`__ne__` (Goulib.drawing.Instance attribute), 44
`__ne__` (Goulib.drawing.Rect attribute), 51
`__ne__` (Goulib.drawing.Spline attribute), 37
`__ne__` (Goulib.drawing.Text attribute), 54
`__ne__` (Goulib.expr.TextVisitor attribute), 63
`__ne__` (Goulib.geom.Arc2 attribute), 84
`__ne__` (Goulib.geom.Circle attribute), 81
`__ne__` (Goulib.geom.Ellipse attribute), 87
`__ne__` (Goulib.geom.Geometry attribute), 65
`__ne__` (Goulib.geom.Line2 attribute), 76
`__ne__` (Goulib.geom.Matrix3 attribute), 93
`__ne__` (Goulib.geom.Point2 attribute), 72
`__ne__` (Goulib.geom.Ray2 attribute), 77
`__ne__` (Goulib.geom.Segment2 attribute), 78
`__ne__` (Goulib.geom.Vector2 attribute), 70
`__ne__` (Goulib.graph.AGraph attribute), 110
`__ne__` (Goulib.graph.DiGraph attribute), 127
`__ne__` (Goulib.graph.GeoGraph attribute), 112
`__ne__` (Goulib.graph.index attribute), 109
`__ne__` (Goulib.graph.index.Index attribute), 107
`__ne__` (Goulib.graph.index.Property attribute), 106
`__ne__` (Goulib.image.Ditherer attribute), 153
`__ne__` (Goulib.image.ErrorDiffusion attribute), 154
`__ne__` (Goulib.image.FloydSteinberg attribute), 155
`__ne__` (Goulib.image.Image attribute), 151
`__ne__` (Goulib.image.Mode attribute), 146
`__ne__` (Goulib.interval.Box attribute), 164
`__ne__` (Goulib.interval.Interval attribute), 158
`__ne__` (Goulib.itertools2.SortingError attribute), 171
`__ne__` (Goulib.itertools2.iter2 attribute), 170
`__ne__` (Goulib.itertools2.keep attribute), 173
`__ne__` (Goulib.markup.ArgumentError attribute), 185
`__ne__` (Goulib.markup.ClosingError attribute), 182
`__ne__` (Goulib.markup.CustomizationError attribute), 189
`__ne__` (Goulib.markup.DeprecationError attribute), 187
`__ne__` (Goulib.markup.InvalidElementError attribute), 186
`__ne__` (Goulib.markup.MarkupError attribute), 181
`__ne__` (Goulib.markup.ModeError attribute), 188
`__ne__` (Goulib.markup.OpeningError attribute), 184
`__ne__` (Goulib.markup.dummy attribute), 179
`__ne__` (Goulib.markup.element attribute), 175
`__ne__` (Goulib.markup.page attribute), 178
`__ne__` (Goulib.markup.russell attribute), 180
`__ne__` (Goulib.motion.Actuator attribute), 211
`__ne__` (Goulib.motion.PVA attribute), 203
`__ne__` (Goulib.motion.Segment attribute), 204
`__ne__` (Goulib.motion.SegmentPoly attribute), 207
`__ne__` (Goulib.motion.Segments attribute), 206
`__ne__` (Goulib.motion.TimeDiagram attribute), 212
`__ne__` (Goulib.optim.BinDict attribute), 216
`__ne__` (Goulib.optim.BinList attribute), 218
`__ne__` (Goulib.optim.DifferentialEvolution attribute), 221
`__ne__` (Goulib.optim.ObjectiveFunction attribute), 214
`__ne__` (Goulib.plot.Plot attribute), 225
`__ne__` (Goulib.stats.Discrete attribute), 233
`__ne__` (Goulib.stats.Stats attribute), 232
`__ne__` (Goulib.table.Cell attribute), 240
`__ne__` (Goulib.table.Row attribute), 242
`__ne__` (Goulib.table.Table attribute), 244

- `__ne__` (Goulib.tests.TestCase attribute), 247
- `__ne__` (Goulib.workdays.WorkCalendar attribute), 254
- `__ne__()` (Goulib.expr.Expr method), 61
- `__ne__()` (Goulib.geom3d.Vector3 method), 94
- `__ne__()` (Goulib.interval.Intervals method), 161
- `__ne__()` (Goulib.piecewise.Piecewise method), 223
- `__ne__()` (Goulib.polynomial.Polynomial method), 227
- `__ne__()` (Goulib.stats.Normal method), 238
- `__ne__()` (Goulib.stats.PDF method), 235
- `__neg__` (Goulib.datetime2.timedelta2 attribute), 21
- `__neg__()` (Goulib.colors.Color method), 7
- `__neg__()` (Goulib.expr.Expr method), 61
- `__neg__()` (Goulib.geom.Point2 method), 72
- `__neg__()` (Goulib.geom.Vector2 method), 69
- `__neg__()` (Goulib.geom3d.Vector3 method), 95
- `__neg__()` (Goulib.image.Image method), 149
- `__neg__()` (Goulib.piecewise.Piecewise method), 223
- `__neg__()` (Goulib.polynomial.Polynomial method), 226
- `__neg__()` (Goulib.stats.Discrete method), 233
- `__neg__()` (Goulib.stats.Normal method), 237
- `__neg__()` (Goulib.stats.PDF method), 235
- `__neg__()` (Goulib.stats.Stats method), 231
- `__new__()` (Goulib.colors.Color method), 8
- `__new__()` (Goulib.colors.Palette method), 9
- `__new__()` (Goulib.container.Record method), 12
- `__new__()` (Goulib.container.Sequence method), 14
- `__new__()` (Goulib.datetime2.date2 method), 18
- `__new__()` (Goulib.datetime2.datetime2 method), 15
- `__new__()` (Goulib.datetime2.time2 method), 20
- `__new__()` (Goulib.datetime2.timedelta2 method), 21
- `__new__()` (Goulib.drawing.BBox method), 33
- `__new__()` (Goulib.drawing.Chain method), 47
- `__new__()` (Goulib.drawing.Drawing method), 57
- `__new__()` (Goulib.drawing.Entity method), 36
- `__new__()` (Goulib.drawing.Group method), 41
- `__new__()` (Goulib.drawing.Instance method), 44
- `__new__()` (Goulib.drawing.Rect method), 51
- `__new__()` (Goulib.drawing.Spline method), 37
- `__new__()` (Goulib.drawing.Text method), 54
- `__new__()` (Goulib.expr.Expr method), 62
- `__new__()` (Goulib.expr.TextVisitor method), 63
- `__new__()` (Goulib.geom.Arc2 method), 84
- `__new__()` (Goulib.geom.Circle method), 81
- `__new__()` (Goulib.geom.Ellipse method), 87
- `__new__()` (Goulib.geom.Geometry method), 65
- `__new__()` (Goulib.geom.Line2 method), 76
- `__new__()` (Goulib.geom.Matrix3 method), 93
- `__new__()` (Goulib.geom.Point2 method), 72
- `__new__()` (Goulib.geom.Ray2 method), 77
- `__new__()` (Goulib.geom.Segment2 method), 78
- `__new__()` (Goulib.geom.Vector2 method), 70
- `__new__()` (Goulib.graph.AGraph method), 110
- `__new__()` (Goulib.graph.DiGraph method), 127
- `__new__()` (Goulib.graph.GeoGraph method), 112
- `__new__()` (Goulib.graph.index method), 109
- `__new__()` (Goulib.graph.index.Index method), 107
- `__new__()` (Goulib.graph.index.Property method), 106
- `__new__()` (Goulib.image.Ditherer method), 153
- `__new__()` (Goulib.image.ErrorDiffusion method), 154
- `__new__()` (Goulib.image.FloydSteinberg method), 155
- `__new__()` (Goulib.image.Image method), 151
- `__new__()` (Goulib.image.Mode method), 146
- `__new__()` (Goulib.interval.Box method), 164
- `__new__()` (Goulib.interval.Interval method), 158
- `__new__()` (Goulib.interval.Intervals method), 161
- `__new__()` (Goulib.itertools2.SortingError method), 171
- `__new__()` (Goulib.itertools2.iter2 method), 170
- `__new__()` (Goulib.itertools2.keep method), 173
- `__new__()` (Goulib.markup.ArgumentError method), 185
- `__new__()` (Goulib.markup.ClosingError method), 182
- `__new__()` (Goulib.markup.CustomizationError method), 189
- `__new__()` (Goulib.markup.DeprecationError method), 187
- `__new__()` (Goulib.markup.InvalidElementError method), 186
- `__new__()` (Goulib.markup.MarkupError method), 181
- `__new__()` (Goulib.markup.ModeError method), 188
- `__new__()` (Goulib.markup.OpeningError method), 184
- `__new__()` (Goulib.markup.dummy method), 179
- `__new__()` (Goulib.markup.element method), 175
- `__new__()` (Goulib.markup.page method), 178
- `__new__()` (Goulib.markup.russell method), 180
- `__new__()` (Goulib.motion.Actuator method), 211
- `__new__()` (Goulib.motion.PVA method), 203
- `__new__()` (Goulib.motion.Segment method), 204
- `__new__()` (Goulib.motion.SegmentPoly method), 207
- `__new__()` (Goulib.motion.Segments method), 206
- `__new__()` (Goulib.motion.TimeDiagram method), 212
- `__new__()` (Goulib.optim.BinDict method), 216
- `__new__()` (Goulib.optim.BinList method), 218
- `__new__()` (Goulib.optim.DifferentialEvolution method), 221
- `__new__()` (Goulib.optim.ObjectiveFunction method), 215
- `__new__()` (Goulib.piecewise.Piecewise method), 223
- `__new__()` (Goulib.plot.Plot method), 225
- `__new__()` (Goulib.polynomial.Polynomial method), 227
- `__new__()` (Goulib.stats.Discrete method), 233
- `__new__()` (Goulib.stats.Normal method), 238
- `__new__()` (Goulib.stats.PDF method), 235
- `__new__()` (Goulib.stats.Stats method), 232
- `__new__()` (Goulib.table.Cell method), 240
- `__new__()` (Goulib.table.Row method), 242
- `__new__()` (Goulib.table.Table method), 244
- `__new__()` (Goulib.tests.TestCase method), 247
- `__new__()` (Goulib.workdays.WorkCalendar method), 254

`__next__()` (Goulib.itertools2.iter2 method), 170
`__next__()` (Goulib.itertools2.keep method), 172
`__nonzero__()` (Goulib.drawing.BBox method), 33
`__nonzero__()` (Goulib.geom3d.Vector3 method), 95
`__nonzero__()` (Goulib.graph.DiGraph method), 127
`__nonzero__()` (Goulib.graph.GeoGraph method), 112
`__nonzero__()` (Goulib.image.Image method), 147
`__nonzero__()` (Goulib.interval.Box method), 163
`__nonzero__()` (Goulib.interval.Interval method), 157
`__or__()` (Goulib.container.Sequence method), 13
`__or__()` (Goulib.expr.Expr method), 61
`__or__()` (Goulib.piecewise.Piecewise method), 223
`__or__()` (Goulib.polynomial.Polynomial method), 227
`__or__()` (Goulib.stats.Normal method), 238
`__or__()` (Goulib.stats.PDF method), 235
`__pos__()` (Goulib.datetime2.timedelta2 attribute), 22
`__pos__()` (Goulib.geom.Point2 method), 72
`__pos__()` (Goulib.geom.Vector2 method), 69
`__pos__()` (Goulib.geom3d.Vector3 method), 95
`__pow__()` (Goulib.expr.Expr method), 61
`__pow__()` (Goulib.piecewise.Piecewise method), 223
`__pow__()` (Goulib.polynomial.Polynomial method), 226
`__pow__()` (Goulib.stats.Discrete method), 233
`__pow__()` (Goulib.stats.Normal method), 238
`__pow__()` (Goulib.stats.PDF method), 235
`__pow__()` (Goulib.stats.Stats method), 231
`__radd__()` (Goulib.datetime2.date2 attribute), 18
`__radd__()` (Goulib.datetime2.datetime2 attribute), 15
`__radd__()` (Goulib.datetime2.timedelta2 attribute), 22
`__radd__()` (Goulib.colors.Color method), 7
`__radd__()` (Goulib.geom.Point2 method), 72
`__radd__()` (Goulib.geom.Vector2 method), 69
`__radd__()` (Goulib.geom3d.Vector3 method), 95
`__radd__()` (Goulib.image.Image method), 150
`__radd__()` (Goulib.polynomial.Polynomial method), 226
`__radd__()` (Goulib.stats.Normal method), 237
`__rdiv__()` (Goulib.geom.Point2 method), 72
`__rdiv__()` (Goulib.geom.Vector2 method), 69
`__rdiv__()` (Goulib.geom3d.Vector3 method), 95
`__rdivmod__()` (Goulib.datetime2.timedelta2 attribute), 22
`__reduce__()` (Goulib.colors.Color method), 8
`__reduce__()` (Goulib.colors.Palette method), 9
`__reduce__()` (Goulib.container.Record method), 12
`__reduce__()` (Goulib.container.Sequence method), 14
`__reduce__()` (Goulib.datetime2.date2 method), 18
`__reduce__()` (Goulib.datetime2.datetime2 method), 15
`__reduce__()` (Goulib.datetime2.time2 method), 20
`__reduce__()` (Goulib.datetime2.timedelta2 method), 22
`__reduce__()` (Goulib.drawing.BBox method), 33
`__reduce__()` (Goulib.drawing.Chain method), 47
`__reduce__()` (Goulib.drawing.Drawing method), 57
`__reduce__()` (Goulib.drawing.Entity method), 36
`__reduce__()` (Goulib.drawing.Group method), 41
`__reduce__()` (Goulib.drawing.Instance method), 44
`__reduce__()` (Goulib.drawing.Rect method), 51
`__reduce__()` (Goulib.drawing.Spline method), 37
`__reduce__()` (Goulib.drawing.Text method), 54
`__reduce__()` (Goulib.expr.Expr method), 62
`__reduce__()` (Goulib.expr.TextVisitor method), 63
`__reduce__()` (Goulib.geom.Arc2 method), 84
`__reduce__()` (Goulib.geom.Circle method), 81
`__reduce__()` (Goulib.geom.Ellipse method), 87
`__reduce__()` (Goulib.geom.Geometry method), 65
`__reduce__()` (Goulib.geom.Line2 method), 76
`__reduce__()` (Goulib.geom.Matrix3 method), 93
`__reduce__()` (Goulib.geom.Point2 method), 72
`__reduce__()` (Goulib.geom.Ray2 method), 77
`__reduce__()` (Goulib.geom.Segment2 method), 78
`__reduce__()` (Goulib.geom.Vector2 method), 70
`__reduce__()` (Goulib.graph.AGraph method), 110
`__reduce__()` (Goulib.graph.DiGraph method), 127
`__reduce__()` (Goulib.graph.GeoGraph method), 112
`__reduce__()` (Goulib.graph.index method), 109
`__reduce__()` (Goulib.graph.index.Index method), 107
`__reduce__()` (Goulib.graph.index.Property method), 106
`__reduce__()` (Goulib.image.Ditherer method), 153
`__reduce__()` (Goulib.image.ErrorDiffusion method), 154
`__reduce__()` (Goulib.image.FloydSteinberg method), 155
`__reduce__()` (Goulib.image.Image method), 151
`__reduce__()` (Goulib.image.Mode method), 146
`__reduce__()` (Goulib.interval.Box method), 164
`__reduce__()` (Goulib.interval.Interval method), 158
`__reduce__()` (Goulib.interval.Intervals method), 161
`__reduce__()` (Goulib.itertools2.SortingError method), 171
`__reduce__()` (Goulib.itertools2.iter2 method), 170
`__reduce__()` (Goulib.itertools2.keep method), 173
`__reduce__()` (Goulib.markup.ArgumentError method), 185
`__reduce__()` (Goulib.markup.ClosingError method), 182
`__reduce__()` (Goulib.markup.CustomizationError method), 189
`__reduce__()` (Goulib.markup.DeprecationError method), 187
`__reduce__()` (Goulib.markup.InvalidElementError method), 186
`__reduce__()` (Goulib.markup.MarkupError method), 181
`__reduce__()` (Goulib.markup.ModeError method), 188
`__reduce__()` (Goulib.markup.OpeningError method), 184
`__reduce__()` (Goulib.markup.dummy method), 179
`__reduce__()` (Goulib.markup.element method), 175
`__reduce__()` (Goulib.markup.page method), 178
`__reduce__()` (Goulib.markup.russell method), 180
`__reduce__()` (Goulib.motion.Actuator method), 211
`__reduce__()` (Goulib.motion.PVA method), 203
`__reduce__()` (Goulib.motion.Segment method), 204

- `__reduce__()` (Goulib.motion.SegmentPoly method), 208
- `__reduce__()` (Goulib.motion.Segments method), 206
- `__reduce__()` (Goulib.motion.TimeDiagram method), 212
- `__reduce__()` (Goulib.optim.BinDict method), 216
- `__reduce__()` (Goulib.optim.BinList method), 218
- `__reduce__()` (Goulib.optim.DifferentialEvolution method), 221
- `__reduce__()` (Goulib.optim.ObjectiveFunction method), 215
- `__reduce__()` (Goulib.piecewise.Piecewise method), 223
- `__reduce__()` (Goulib.plot.Plot method), 225
- `__reduce__()` (Goulib.polynomial.Polynomial method), 227
- `__reduce__()` (Goulib.stats.Discrete method), 233
- `__reduce__()` (Goulib.stats.Normal method), 238
- `__reduce__()` (Goulib.stats.PDF method), 235
- `__reduce__()` (Goulib.stats.Stats method), 232
- `__reduce__()` (Goulib.table.Cell method), 240
- `__reduce__()` (Goulib.table.Row method), 242
- `__reduce__()` (Goulib.table.Table method), 244
- `__reduce__()` (Goulib.tests.TestCase method), 247
- `__reduce__()` (Goulib.workdays.WorkCalendar method), 254
- `__reduce_ex__()` (Goulib.colors.Color method), 8
- `__reduce_ex__()` (Goulib.colors.Palette method), 9
- `__reduce_ex__()` (Goulib.container.Record method), 12
- `__reduce_ex__()` (Goulib.container.Sequence method), 14
- `__reduce_ex__()` (Goulib.datetime2.date2 method), 18
- `__reduce_ex__()` (Goulib.datetime2.datetime2 method), 15
- `__reduce_ex__()` (Goulib.datetime2.time2 method), 20
- `__reduce_ex__()` (Goulib.datetime2.timedelta2 method), 22
- `__reduce_ex__()` (Goulib.drawing.BBox method), 33
- `__reduce_ex__()` (Goulib.drawing.Chain method), 47
- `__reduce_ex__()` (Goulib.drawing.Drawing method), 57
- `__reduce_ex__()` (Goulib.drawing.Entity method), 36
- `__reduce_ex__()` (Goulib.drawing.Group method), 41
- `__reduce_ex__()` (Goulib.drawing.Instance method), 44
- `__reduce_ex__()` (Goulib.drawing.Rect method), 51
- `__reduce_ex__()` (Goulib.drawing.Spline method), 37
- `__reduce_ex__()` (Goulib.drawing.Text method), 54
- `__reduce_ex__()` (Goulib.expr.Expr method), 62
- `__reduce_ex__()` (Goulib.expr.TextVisitor method), 63
- `__reduce_ex__()` (Goulib.geom.Arc2 method), 84
- `__reduce_ex__()` (Goulib.geom.Circle method), 81
- `__reduce_ex__()` (Goulib.geom.Ellipse method), 87
- `__reduce_ex__()` (Goulib.geom.Geometry method), 65
- `__reduce_ex__()` (Goulib.geom.Line2 method), 76
- `__reduce_ex__()` (Goulib.geom.Matrix3 method), 93
- `__reduce_ex__()` (Goulib.geom.Point2 method), 72
- `__reduce_ex__()` (Goulib.geom.Ray2 method), 77
- `__reduce_ex__()` (Goulib.geom.Segment2 method), 78
- `__reduce_ex__()` (Goulib.geom.Vector2 method), 70
- `__reduce_ex__()` (Goulib.graph.AGraph method), 110
- `__reduce_ex__()` (Goulib.graph.DiGraph method), 128
- `__reduce_ex__()` (Goulib.graph.GeoGraph method), 112
- `__reduce_ex__()` (Goulib.graph.index method), 109
- `__reduce_ex__()` (Goulib.graph.index.Index method), 108
- `__reduce_ex__()` (Goulib.graph.index.Property method), 106
- `__reduce_ex__()` (Goulib.image.Ditherer method), 153
- `__reduce_ex__()` (Goulib.image.ErrorDiffusion method), 154
- `__reduce_ex__()` (Goulib.image.FloydSteinberg method), 155
- `__reduce_ex__()` (Goulib.image.Image method), 151
- `__reduce_ex__()` (Goulib.image.Mode method), 146
- `__reduce_ex__()` (Goulib.interval.Box method), 164
- `__reduce_ex__()` (Goulib.interval.Interval method), 158
- `__reduce_ex__()` (Goulib.interval.Intervals method), 161
- `__reduce_ex__()` (Goulib.itertools2.SortingError method), 171
- `__reduce_ex__()` (Goulib.itertools2.iter2 method), 170
- `__reduce_ex__()` (Goulib.itertools2.keep method), 173
- `__reduce_ex__()` (Goulib.markup.ArgumentError method), 185
- `__reduce_ex__()` (Goulib.markup.ClosingError method), 182
- `__reduce_ex__()` (Goulib.markup.CustomizationError method), 189
- `__reduce_ex__()` (Goulib.markup.DeprecationError method), 187
- `__reduce_ex__()` (Goulib.markup.InvalidElementError method), 186
- `__reduce_ex__()` (Goulib.markup.MarkupError method), 181
- `__reduce_ex__()` (Goulib.markup.ModeError method), 188
- `__reduce_ex__()` (Goulib.markup.OpeningError method), 184
- `__reduce_ex__()` (Goulib.markup.dummy method), 179
- `__reduce_ex__()` (Goulib.markup.element method), 175
- `__reduce_ex__()` (Goulib.markup.page method), 178
- `__reduce_ex__()` (Goulib.markup.russell method), 180
- `__reduce_ex__()` (Goulib.motion.Actuator method), 211
- `__reduce_ex__()` (Goulib.motion.PVA method), 203
- `__reduce_ex__()` (Goulib.motion.Segment method), 204
- `__reduce_ex__()` (Goulib.motion.SegmentPoly method), 208
- `__reduce_ex__()` (Goulib.motion.Segments method), 206
- `__reduce_ex__()` (Goulib.motion.TimeDiagram method), 212
- `__reduce_ex__()` (Goulib.optim.BinDict method), 216
- `__reduce_ex__()` (Goulib.optim.BinList method), 219
- `__reduce_ex__()` (Goulib.optim.DifferentialEvolution method), 221

method), 221
__reduce_ex__() (Goulib.optim.ObjectiveFunction method), 215
__reduce_ex__() (Goulib.piecewise.Piecewise method), 223
__reduce_ex__() (Goulib.plot.Plot method), 225
__reduce_ex__() (Goulib.polynomial.Polynomial method), 227
__reduce_ex__() (Goulib.stats.Discrete method), 233
__reduce_ex__() (Goulib.stats.Normal method), 238
__reduce_ex__() (Goulib.stats.PDF method), 235
__reduce_ex__() (Goulib.stats.Stats method), 232
__reduce_ex__() (Goulib.table.Cell method), 240
__reduce_ex__() (Goulib.table.Row method), 242
__reduce_ex__() (Goulib.table.Table method), 245
__reduce_ex__() (Goulib.tests.TestCase method), 247
__reduce_ex__() (Goulib.workdays.WorkCalendar method), 255
__repr__ (Goulib.container.Record attribute), 12
__repr__ (Goulib.datetime2.date2 attribute), 18
__repr__ (Goulib.datetime2.datetime2 attribute), 15
__repr__ (Goulib.datetime2.time2 attribute), 20
__repr__ (Goulib.datetime2.timedelta2 attribute), 22
__repr__ (Goulib.drawing.BBox attribute), 33
__repr__ (Goulib.drawing.Drawing attribute), 57
__repr__ (Goulib.drawing.Group attribute), 41
__repr__ (Goulib.expr.TextVisitor attribute), 63
__repr__ (Goulib.geom.Geometry attribute), 65
__repr__ (Goulib.graph.AGraph attribute), 110
__repr__ (Goulib.graph.DiGraph attribute), 128
__repr__ (Goulib.graph.GeoGraph attribute), 112
__repr__ (Goulib.graph.index attribute), 109
__repr__ (Goulib.graph.index.Index attribute), 108
__repr__ (Goulib.graph.index.Property attribute), 106
__repr__ (Goulib.interval.Box attribute), 164
__repr__ (Goulib.itertools2.SortingError attribute), 172
__repr__ (Goulib.itertools2.iter2 attribute), 170
__repr__ (Goulib.itertools2.keep attribute), 173
__repr__ (Goulib.markup.ArgumentError attribute), 185
__repr__ (Goulib.markup.ClosingError attribute), 183
__repr__ (Goulib.markup.CustomizationError attribute), 190
__repr__ (Goulib.markup.DeprecationError attribute), 187
__repr__ (Goulib.markup.InvalidElementError attribute), 186
__repr__ (Goulib.markup.MarkupError attribute), 181
__repr__ (Goulib.markup.ModeError attribute), 188
__repr__ (Goulib.markup.OpeningError attribute), 184
__repr__ (Goulib.markup.dummy attribute), 179
__repr__ (Goulib.markup.element attribute), 175
__repr__ (Goulib.markup.page attribute), 178
__repr__ (Goulib.markup.russell attribute), 180
__repr__ (Goulib.motion.Actuator attribute), 211
__repr__ (Goulib.motion.PVA attribute), 203
__repr__ (Goulib.motion.Segment attribute), 205
__repr__ (Goulib.motion.SegmentPoly attribute), 208
__repr__ (Goulib.motion.Segments attribute), 206
__repr__ (Goulib.optim.DifferentialEvolution attribute), 221
__repr__ (Goulib.optim.ObjectiveFunction attribute), 215
__repr__ (Goulib.plot.Plot attribute), 225
__repr__ (Goulib.workdays.WorkCalendar attribute), 255
__repr__() (Goulib.colors.Color method), 7
__repr__() (Goulib.colors.Palette method), 8
__repr__() (Goulib.container.Sequence method), 13
__repr__() (Goulib.drawing.Chain method), 28, 46
__repr__() (Goulib.drawing.Entity method), 26, 34
__repr__() (Goulib.drawing.Instance method), 28, 43
__repr__() (Goulib.drawing.Rect method), 29, 50
__repr__() (Goulib.drawing.Spline method), 38
__repr__() (Goulib.drawing.Text method), 55
__repr__() (Goulib.expr.Expr method), 60
__repr__() (Goulib.geom.Arc2 method), 84
__repr__() (Goulib.geom.Circle method), 80
__repr__() (Goulib.geom.Ellipse method), 86
__repr__() (Goulib.geom.Line2 method), 75
__repr__() (Goulib.geom.Matrix3 method), 92
__repr__() (Goulib.geom.Point2 method), 72
__repr__() (Goulib.geom.Ray2 method), 77
__repr__() (Goulib.geom.Segment2 method), 77
__repr__() (Goulib.geom.Vector2 method), 68
__repr__() (Goulib.geom3d.Line3 method), 97
__repr__() (Goulib.geom3d.Matrix4 method), 99
__repr__() (Goulib.geom3d.Plane method), 99
__repr__() (Goulib.geom3d.Quaternion method), 102
__repr__() (Goulib.geom3d.Segment3 method), 97
__repr__() (Goulib.geom3d.Sphere method), 98
__repr__() (Goulib.geom3d.Vector3 method), 94
__repr__() (Goulib.image.Ditherer method), 152
__repr__() (Goulib.image.ErrorDiffusion method), 154
__repr__() (Goulib.image.FloydSteinberg method), 155
__repr__() (Goulib.image.Image method), 149
__repr__() (Goulib.image.Mode method), 145
__repr__() (Goulib.interval.Interval method), 157
__repr__() (Goulib.interval.Intervals method), 161
__repr__() (Goulib.motion.TimeDiagram method), 211
__repr__() (Goulib.optim.BinDict method), 216
__repr__() (Goulib.optim.BinList method), 219
__repr__() (Goulib.piecewise.Piecewise method), 223
__repr__() (Goulib.polynomial.Polynomial method), 227
__repr__() (Goulib.statemachine.TimeMarker method), 229
__repr__() (Goulib.stats.Discrete method), 233
__repr__() (Goulib.stats.Normal method), 238
__repr__() (Goulib.stats.PDF method), 235
__repr__() (Goulib.stats.Stats method), 231
__repr__() (Goulib.table.Cell method), 240

__repr__() (Goulib.table.Row method), 241
 __repr__() (Goulib.table.Table method), 242
 __repr__() (Goulib.tests.TestCase method), 248
 __reset__() (Goulib.statemachine.StateMachine method), 229
 __reversed__() (Goulib.colors.Palette method), 9
 __reversed__() (Goulib.container.Record method), 12
 __reversed__() (Goulib.drawing.BBox method), 33
 __reversed__() (Goulib.drawing.Chain method), 48
 __reversed__() (Goulib.drawing.Drawing method), 57
 __reversed__() (Goulib.drawing.Group method), 41
 __reversed__() (Goulib.drawing.Rect method), 51
 __reversed__() (Goulib.interval.Box method), 164
 __reversed__() (Goulib.interval.Interval method), 158
 __reversed__() (Goulib.interval.Intervals method), 161
 __reversed__() (Goulib.optim.BinList method), 219
 __reversed__() (Goulib.table.Table method), 245
 __rfloordiv__() (Goulib.datetime2.timedelta2 attribute), 22
 __rfloordiv__() (Goulib.geom.Point2 method), 72
 __rfloordiv__() (Goulib.geom.Vector2 method), 69
 __rfloordiv__() (Goulib.geom3d.Vector3 method), 95
 __rmod__() (Goulib.datetime2.timedelta2 attribute), 22
 __rmul__() (Goulib.datetime2.timedelta2 attribute), 22
 __rmul__() (Goulib.drawing.BBox attribute), 33
 __rmul__() (Goulib.drawing.Chain attribute), 48
 __rmul__() (Goulib.drawing.Drawing attribute), 57
 __rmul__() (Goulib.drawing.Group attribute), 41
 __rmul__() (Goulib.drawing.Rect attribute), 51
 __rmul__() (Goulib.interval.Box attribute), 164
 __rmul__() (Goulib.interval.Interval attribute), 158
 __rmul__() (Goulib.optim.BinList attribute), 219
 __rmul__() (Goulib.table.Table attribute), 245
 __rmul__() (Goulib.expr.Expr method), 61
 __rmul__() (Goulib.geom.Point2 method), 72
 __rmul__() (Goulib.geom.Vector2 method), 69
 __rmul__() (Goulib.geom3d.Vector3 method), 95
 __rmul__() (Goulib.piecewise.Piecewise method), 223
 __rmul__() (Goulib.polynomial.Polynomial method), 226
 __rmul__() (Goulib.stats.Normal method), 238
 __rmul__() (Goulib.stats.PDF method), 235
 __rshift__() (Goulib.expr.Expr method), 61
 __rshift__() (Goulib.piecewise.Piecewise method), 222
 __rshift__() (Goulib.polynomial.Polynomial method), 227
 __rshift__() (Goulib.stats.Normal method), 238
 __rshift__() (Goulib.stats.PDF method), 235
 __rsub__() (Goulib.datetime2.date2 attribute), 18
 __rsub__() (Goulib.datetime2.datetime2 attribute), 15
 __rsub__() (Goulib.datetime2.timedelta2 attribute), 22
 __rsub__() (Goulib.geom.Point2 method), 72
 __rsub__() (Goulib.geom.Vector2 method), 69
 __rsub__() (Goulib.geom3d.Vector3 method), 95
 __rsub__() (Goulib.polynomial.Polynomial method), 226
 __rsub__() (Goulib.stats.Normal method), 237
 __rtruediv__() (Goulib.datetime2.timedelta2 attribute), 22
 __rtruediv__() (Goulib.geom.Point2 method), 72
 __rtruediv__() (Goulib.geom.Vector2 method), 69
 __rtruediv__() (Goulib.geom3d.Vector3 method), 95
 __setattr__() (Goulib.colors.Color attribute), 8
 __setattr__() (Goulib.colors.Palette attribute), 10
 __setattr__() (Goulib.container.Sequence attribute), 14
 __setattr__() (Goulib.datetime2.date2 attribute), 18
 __setattr__() (Goulib.datetime2.datetime2 attribute), 15
 __setattr__() (Goulib.datetime2.time2 attribute), 20
 __setattr__() (Goulib.datetime2.timedelta2 attribute), 22
 __setattr__() (Goulib.drawing.BBox attribute), 33
 __setattr__() (Goulib.drawing.Chain attribute), 48
 __setattr__() (Goulib.drawing.Drawing attribute), 57
 __setattr__() (Goulib.drawing.Entity attribute), 36
 __setattr__() (Goulib.drawing.Group attribute), 41
 __setattr__() (Goulib.drawing.Instance attribute), 44
 __setattr__() (Goulib.drawing.Rect attribute), 51
 __setattr__() (Goulib.drawing.Spline attribute), 38
 __setattr__() (Goulib.drawing.Text attribute), 55
 __setattr__() (Goulib.expr.Expr attribute), 62
 __setattr__() (Goulib.expr.TextVisitor attribute), 63
 __setattr__() (Goulib.geom.Arc2 attribute), 84
 __setattr__() (Goulib.geom.Circle attribute), 81
 __setattr__() (Goulib.geom.Ellipse attribute), 87
 __setattr__() (Goulib.geom.Geometry attribute), 65
 __setattr__() (Goulib.geom.Line2 attribute), 76
 __setattr__() (Goulib.geom.Matrix3 attribute), 93
 __setattr__() (Goulib.geom.Point2 attribute), 72
 __setattr__() (Goulib.geom.Ray2 attribute), 77
 __setattr__() (Goulib.geom.Segment2 attribute), 78
 __setattr__() (Goulib.geom.Vector2 attribute), 70
 __setattr__() (Goulib.graph.AGraph attribute), 110
 __setattr__() (Goulib.graph.DiGraph attribute), 128
 __setattr__() (Goulib.graph.GeoGraph attribute), 112
 __setattr__() (Goulib.graph.index attribute), 109
 __setattr__() (Goulib.graph.index.Index attribute), 108
 __setattr__() (Goulib.graph.index.Property attribute), 106
 __setattr__() (Goulib.image.Ditherer attribute), 153
 __setattr__() (Goulib.image.ErrorDiffusion attribute), 154
 __setattr__() (Goulib.image.FloydSteinberg attribute), 155
 __setattr__() (Goulib.image.Image attribute), 151
 __setattr__() (Goulib.image.Mode attribute), 146
 __setattr__() (Goulib.interval.Box attribute), 164
 __setattr__() (Goulib.interval.Interval attribute), 158
 __setattr__() (Goulib.interval.Intervals attribute), 161
 __setattr__() (Goulib.itertools2.SortingError attribute), 172
 __setattr__() (Goulib.itertools2.iter2 attribute), 170
 __setattr__() (Goulib.itertools2.keep attribute), 173
 __setattr__() (Goulib.markup.ArgumentError attribute), 185
 __setattr__() (Goulib.markup.ClosingError attribute), 183
 __setattr__() (Goulib.markup.CustomizationError attribute), 190

`__setattr__` (Goulib.markup.DeprecationError attribute), 187

`__setattr__` (Goulib.markup.InvalidElementError attribute), 186

`__setattr__` (Goulib.markup.MarkupError attribute), 181

`__setattr__` (Goulib.markup.ModeError attribute), 188

`__setattr__` (Goulib.markup.OpeningError attribute), 184

`__setattr__` (Goulib.markup.dummy attribute), 179

`__setattr__` (Goulib.markup.element attribute), 175

`__setattr__` (Goulib.markup.page attribute), 178

`__setattr__` (Goulib.markup.russell attribute), 180

`__setattr__` (Goulib.motion.Actuator attribute), 211

`__setattr__` (Goulib.motion.PVA attribute), 203

`__setattr__` (Goulib.motion.Segment attribute), 205

`__setattr__` (Goulib.motion.SegmentPoly attribute), 208

`__setattr__` (Goulib.motion.Segments attribute), 206

`__setattr__` (Goulib.motion.TimeDiagram attribute), 212

`__setattr__` (Goulib.optim.BinDict attribute), 216

`__setattr__` (Goulib.optim.BinList attribute), 219

`__setattr__` (Goulib.optim.DifferentialEvolution attribute), 221

`__setattr__` (Goulib.optim.ObjectiveFunction attribute), 215

`__setattr__` (Goulib.piecewise.Piecewise attribute), 223

`__setattr__` (Goulib.plot.Plot attribute), 225

`__setattr__` (Goulib.polynomial.Polynomial attribute), 227

`__setattr__` (Goulib.stats.Discrete attribute), 233

`__setattr__` (Goulib.stats.Normal attribute), 238

`__setattr__` (Goulib.stats.PDF attribute), 235

`__setattr__` (Goulib.stats.Stats attribute), 232

`__setattr__` (Goulib.table.Cell attribute), 241

`__setattr__` (Goulib.table.Row attribute), 242

`__setattr__` (Goulib.table.Table attribute), 245

`__setattr__` (Goulib.tests.TestCase attribute), 248

`__setattr__` (Goulib.workdays.WorkCalendar attribute), 255

`__setattr__()` (Goulib.container.Record method), 11

`__setitem__` (Goulib.colors.Palette attribute), 10

`__setitem__` (Goulib.container.Record attribute), 12

`__setitem__` (Goulib.drawing.BBox attribute), 33

`__setitem__` (Goulib.drawing.Chain attribute), 48

`__setitem__` (Goulib.drawing.Drawing attribute), 57

`__setitem__` (Goulib.drawing.Group attribute), 41

`__setitem__` (Goulib.drawing.Rect attribute), 51

`__setitem__` (Goulib.graph.index.Index attribute), 108

`__setitem__` (Goulib.interval.Box attribute), 164

`__setitem__` (Goulib.interval.Interval attribute), 158

`__setitem__` (Goulib.table.Table attribute), 245

`__setitem__()` (Goulib.geom.Matrix3 method), 93

`__setitem__()` (Goulib.geom3d.Matrix4 method), 99

`__setitem__()` (Goulib.interval.Intervals method), 161

`__setitem__()` (Goulib.optim.BinDict method), 215

`__setitem__()` (Goulib.optim.BinList method), 217

`__setstate__()` (Goulib.itertools2.SortingError method), 172

`__setstate__()` (Goulib.markup.ArgumentError method), 185

`__setstate__()` (Goulib.markup.ClosingError method), 183

`__setstate__()` (Goulib.markup.CustomizationError method), 190

`__setstate__()` (Goulib.markup.DeprecationError method), 187

`__setstate__()` (Goulib.markup.InvalidElementError method), 186

`__setstate__()` (Goulib.markup.MarkupError method), 181

`__setstate__()` (Goulib.markup.ModeError method), 188

`__setstate__()` (Goulib.markup.OpeningError method), 184

`__sizeof__()` (Goulib.colors.Color method), 8

`__sizeof__()` (Goulib.colors.Palette method), 10

`__sizeof__()` (Goulib.container.Record method), 12

`__sizeof__()` (Goulib.container.Sequence method), 14

`__sizeof__()` (Goulib.datetime2.date2 method), 18

`__sizeof__()` (Goulib.datetime2.datetime2 method), 15

`__sizeof__()` (Goulib.datetime2.time2 method), 20

`__sizeof__()` (Goulib.datetime2.timedelta2 method), 22

`__sizeof__()` (Goulib.drawing.BBox method), 33

`__sizeof__()` (Goulib.drawing.Chain method), 48

`__sizeof__()` (Goulib.drawing.Drawing method), 57

`__sizeof__()` (Goulib.drawing.Entity method), 36

`__sizeof__()` (Goulib.drawing.Group method), 41

`__sizeof__()` (Goulib.drawing.Instance method), 44

`__sizeof__()` (Goulib.drawing.Rect method), 51

`__sizeof__()` (Goulib.drawing.Spline method), 38

`__sizeof__()` (Goulib.drawing.Text method), 55

`__sizeof__()` (Goulib.expr.Expr method), 62

`__sizeof__()` (Goulib.expr.TextVisitor method), 63

`__sizeof__()` (Goulib.geom.Arc2 method), 84

`__sizeof__()` (Goulib.geom.Circle method), 81

`__sizeof__()` (Goulib.geom.Ellipse method), 87

`__sizeof__()` (Goulib.geom.Geometry method), 65

`__sizeof__()` (Goulib.geom.Line2 method), 76

`__sizeof__()` (Goulib.geom.Matrix3 method), 93

`__sizeof__()` (Goulib.geom.Point2 method), 72

`__sizeof__()` (Goulib.geom.Ray2 method), 77

`__sizeof__()` (Goulib.geom.Segment2 method), 78

`__sizeof__()` (Goulib.geom.Vector2 method), 70

`__sizeof__()` (Goulib.graph.AGraph method), 110

`__sizeof__()` (Goulib.graph.DiGraph method), 128

`__sizeof__()` (Goulib.graph.GeoGraph method), 112

`__sizeof__()` (Goulib.graph.index method), 109

`__sizeof__()` (Goulib.graph.index.Index method), 108

`__sizeof__()` (Goulib.graph.index.Property method), 106

`__sizeof__()` (Goulib.image.Ditherer method), 153

`__sizeof__()` (Goulib.image.ErrorDiffusion method), 154

- `__sizeof__()` (Goulib.image.FloydSteinberg method), 155
- `__sizeof__()` (Goulib.image.Image method), 151
- `__sizeof__()` (Goulib.image.Mode method), 146
- `__sizeof__()` (Goulib.interval.Box method), 164
- `__sizeof__()` (Goulib.interval.Interval method), 158
- `__sizeof__()` (Goulib.interval.Intervals method), 161
- `__sizeof__()` (Goulib.itertools2.SortingError method), 172
- `__sizeof__()` (Goulib.itertools2.iter2 method), 170
- `__sizeof__()` (Goulib.itertools2.keep method), 173
- `__sizeof__()` (Goulib.markup.ArgumentError method), 185
- `__sizeof__()` (Goulib.markup.ClosingError method), 183
- `__sizeof__()` (Goulib.markup.CustomizationError method), 190
- `__sizeof__()` (Goulib.markup.DeprecationError method), 187
- `__sizeof__()` (Goulib.markup.InvalidElementError method), 186
- `__sizeof__()` (Goulib.markup.MarkupError method), 181
- `__sizeof__()` (Goulib.markup.ModeError method), 188
- `__sizeof__()` (Goulib.markup.OpeningError method), 184
- `__sizeof__()` (Goulib.markup.dummy method), 179
- `__sizeof__()` (Goulib.markup.element method), 175
- `__sizeof__()` (Goulib.markup.page method), 178
- `__sizeof__()` (Goulib.markup.russell method), 180
- `__sizeof__()` (Goulib.motion.Actuator method), 211
- `__sizeof__()` (Goulib.motion.PVA method), 203
- `__sizeof__()` (Goulib.motion.Segment method), 205
- `__sizeof__()` (Goulib.motion.SegmentPoly method), 208
- `__sizeof__()` (Goulib.motion.Segments method), 206
- `__sizeof__()` (Goulib.motion.TimeDiagram method), 212
- `__sizeof__()` (Goulib.optim.BinDict method), 217
- `__sizeof__()` (Goulib.optim.BinList method), 219
- `__sizeof__()` (Goulib.optim.DifferentialEvolution method), 222
- `__sizeof__()` (Goulib.optim.ObjectiveFunction method), 215
- `__sizeof__()` (Goulib.piecewise.Piecewise method), 223
- `__sizeof__()` (Goulib.plot.Plot method), 225
- `__sizeof__()` (Goulib.polynomial.Polynomial method), 227
- `__sizeof__()` (Goulib.stats.Discrete method), 234
- `__sizeof__()` (Goulib.stats.Normal method), 238
- `__sizeof__()` (Goulib.stats.PDF method), 235
- `__sizeof__()` (Goulib.stats.Stats method), 232
- `__sizeof__()` (Goulib.table.Cell method), 241
- `__sizeof__()` (Goulib.table.Row method), 242
- `__sizeof__()` (Goulib.table.Table method), 245
- `__sizeof__()` (Goulib.tests.TestCase method), 248
- `__sizeof__()` (Goulib.workdays.WorkCalendar method), 255
- `__slotnames__` (Goulib.geom.Matrix3 attribute), 93
- `__slots__` (Goulib.interval.Intervals attribute), 161
- `__slots__` (Goulib.itertools2.keep attribute), 173
- `__str__` (Goulib.colors.Color attribute), 8
- `__str__` (Goulib.colors.Palette attribute), 10
- `__str__` (Goulib.container.Sequence attribute), 14
- `__str__` (Goulib.datetime2.date2 attribute), 18
- `__str__` (Goulib.datetime2.datetime2 attribute), 15
- `__str__` (Goulib.datetime2.time2 attribute), 20
- `__str__` (Goulib.datetime2.timedelta2 attribute), 22
- `__str__` (Goulib.drawing.BBox attribute), 34
- `__str__` (Goulib.drawing.Chain attribute), 48
- `__str__` (Goulib.drawing.Drawing attribute), 57
- `__str__` (Goulib.drawing.Entity attribute), 36
- `__str__` (Goulib.drawing.Group attribute), 41
- `__str__` (Goulib.drawing.Instance attribute), 44
- `__str__` (Goulib.drawing.Rect attribute), 51
- `__str__` (Goulib.drawing.Spline attribute), 38
- `__str__` (Goulib.drawing.Text attribute), 55
- `__str__` (Goulib.expr.TextVisitor attribute), 63
- `__str__` (Goulib.geom.Arc2 attribute), 84
- `__str__` (Goulib.geom.Circle attribute), 82
- `__str__` (Goulib.geom.Ellipse attribute), 87
- `__str__` (Goulib.geom.Geometry attribute), 65
- `__str__` (Goulib.geom.Line2 attribute), 76
- `__str__` (Goulib.geom.Matrix3 attribute), 94
- `__str__` (Goulib.geom.Point2 attribute), 72
- `__str__` (Goulib.geom.Ray2 attribute), 77
- `__str__` (Goulib.geom.Segment2 attribute), 78
- `__str__` (Goulib.geom.Vector2 attribute), 70
- `__str__` (Goulib.graph.AGraph attribute), 110
- `__str__` (Goulib.graph.index attribute), 109
- `__str__` (Goulib.graph.index.Index attribute), 108
- `__str__` (Goulib.graph.index.Property attribute), 106
- `__str__` (Goulib.image.Ditherer attribute), 153
- `__str__` (Goulib.image.ErrorDiffusion attribute), 154
- `__str__` (Goulib.image.FloydSteinberg attribute), 155
- `__str__` (Goulib.image.Image attribute), 151
- `__str__` (Goulib.image.Mode attribute), 146
- `__str__` (Goulib.interval.Box attribute), 164
- `__str__` (Goulib.itertools2.SortingError attribute), 172
- `__str__` (Goulib.itertools2.iter2 attribute), 171
- `__str__` (Goulib.itertools2.keep attribute), 173
- `__str__` (Goulib.markup.dummy attribute), 179
- `__str__` (Goulib.markup.element attribute), 175
- `__str__` (Goulib.markup.russell attribute), 180
- `__str__` (Goulib.motion.Actuator attribute), 211
- `__str__` (Goulib.motion.PVA attribute), 203
- `__str__` (Goulib.motion.Segment attribute), 205
- `__str__` (Goulib.motion.SegmentPoly attribute), 208
- `__str__` (Goulib.motion.TimeDiagram attribute), 212
- `__str__` (Goulib.optim.BinDict attribute), 217
- `__str__` (Goulib.optim.BinList attribute), 219
- `__str__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__str__` (Goulib.optim.ObjectiveFunction attribute), 215

`__str__` (Goulib.plot.Plot attribute), 225
`__str__` (Goulib.stats.Discrete attribute), 234
`__str__` (Goulib.stats.Stats attribute), 232
`__str__` (Goulib.table.Cell attribute), 241
`__str__` (Goulib.table.Row attribute), 242
`__str__` (Goulib.workdays.WorkCalendar attribute), 255
`__str__()` (Goulib.container.Record method), 11
`__str__()` (Goulib.expr.Expr method), 61
`__str__()` (Goulib.graph.DiGraph method), 128
`__str__()` (Goulib.graph.GeoGraph method), 112
`__str__()` (Goulib.interval.Interval method), 157
`__str__()` (Goulib.interval.Intervals method), 159
`__str__()` (Goulib.markup.ArgumentError method), 185
`__str__()` (Goulib.markup.ClosingError method), 183
`__str__()` (Goulib.markup.CustomizationError method), 190
`__str__()` (Goulib.markup.DeprecationError method), 187
`__str__()` (Goulib.markup.InvalidElementError method), 186
`__str__()` (Goulib.markup.MarkupError method), 181
`__str__()` (Goulib.markup.ModeError method), 189
`__str__()` (Goulib.markup.OpeningError method), 184
`__str__()` (Goulib.markup.page method), 176
`__str__()` (Goulib.motion.Segments method), 205
`__str__()` (Goulib.piecewise.Piecewise method), 222
`__str__()` (Goulib.polynomial.Polynomial method), 227
`__str__()` (Goulib.stats.Normal method), 237
`__str__()` (Goulib.stats.PDF method), 235
`__str__()` (Goulib.table.Table method), 242
`__str__()` (Goulib.tests.TestCase method), 248
`__sub__` (Goulib.datetime2.date2 attribute), 18
`__sub__` (Goulib.datetime2.timedelta2 attribute), 22
`__sub__()` (Goulib.colors.Color method), 7
`__sub__()` (Goulib.container.Sequence method), 13
`__sub__()` (Goulib.datetime2.datetime2 method), 14
`__sub__()` (Goulib.expr.Expr method), 61
`__sub__()` (Goulib.geom.Matrix3 method), 93
`__sub__()` (Goulib.geom.Point2 method), 72
`__sub__()` (Goulib.geom.Vector2 method), 69
`__sub__()` (Goulib.geom3d.Vector3 method), 95
`__sub__()` (Goulib.image.Image method), 150
`__sub__()` (Goulib.piecewise.Piecewise method), 223
`__sub__()` (Goulib.polynomial.Polynomial method), 226
`__sub__()` (Goulib.stats.Discrete method), 234
`__sub__()` (Goulib.stats.Normal method), 237
`__sub__()` (Goulib.stats.PDF method), 235
`__sub__()` (Goulib.stats.Stats method), 231
`__suppress_context__` (Goulib.itertools2.SortingError attribute), 172
`__suppress_context__` (Goulib.markup.ArgumentError attribute), 185
`__suppress_context__` (Goulib.markup.ClosingError attribute), 183
`__suppress_context__` (Goulib.markup.CustomizationError attribute), 190
`__suppress_context__` (Goulib.markup.DeprecationError attribute), 187
`__suppress_context__` (Goulib.markup.InvalidElementError attribute), 186
`__suppress_context__` (Goulib.markup.MarkupError attribute), 182
`__suppress_context__` (Goulib.markup.ModeError attribute), 189
`__suppress_context__` (Goulib.markup.OpeningError attribute), 184
`__traceback__` (Goulib.itertools2.SortingError attribute), 172
`__traceback__` (Goulib.markup.ArgumentError attribute), 185
`__traceback__` (Goulib.markup.ClosingError attribute), 183
`__traceback__` (Goulib.markup.CustomizationError attribute), 190
`__traceback__` (Goulib.markup.DeprecationError attribute), 187
`__traceback__` (Goulib.markup.InvalidElementError attribute), 186
`__traceback__` (Goulib.markup.MarkupError attribute), 182
`__traceback__` (Goulib.markup.ModeError attribute), 189
`__traceback__` (Goulib.markup.OpeningError attribute), 184
`__truediv__` (Goulib.datetime2.timedelta2 attribute), 22
`__truediv__()` (Goulib.expr.Expr method), 61
`__truediv__()` (Goulib.geom.Point2 method), 72
`__truediv__()` (Goulib.geom.Vector2 method), 69
`__truediv__()` (Goulib.geom3d.Vector3 method), 95
`__truediv__()` (Goulib.image.Image method), 150
`__truediv__()` (Goulib.piecewise.Piecewise method), 223
`__truediv__()` (Goulib.polynomial.Polynomial method), 227
`__truediv__()` (Goulib.stats.Normal method), 237
`__truediv__()` (Goulib.stats.PDF method), 235
`__xor__` (Goulib.expr.Expr method), 61
`__xor__` (Goulib.piecewise.Piecewise method), 223
`__xor__` (Goulib.polynomial.Polynomial method), 227
`__xor__` (Goulib.stats.Normal method), 238
`__xor__` (Goulib.stats.PDF method), 235

A

`abundance()` (in module Goulib.math2), 200
`accsum()` (in module Goulib.math2), 192
`accumulate()` (Goulib.container.Sequence method), 14
`accumulate()` (in module Goulib.itertools2), 167
`aci_to_color()` (in module Goulib.colors), 10
`Actuator` (class in Goulib.motion), 208
`adapt_rgb()` (in module Goulib.image), 146

- add() (Goulib.image.Image method), 150
- add() (Goulib.interval.Intervals method), 159
- add() (Goulib.markup.page method), 177
- add() (Goulib.motion.Segments method), 205
- add() (in module Goulib.polynomial), 228
- add_cycle() (Goulib.graph.DiGraph method), 128
- add_cycle() (Goulib.graph.GeoGraph method), 112
- add_edge() (Goulib.graph.DiGraph method), 128
- add_edge() (Goulib.graph.GeoGraph method), 112
- add_edge2() (Goulib.graph.DiGraph method), 128
- add_edge2() (Goulib.graph.GeoGraph method), 112
- add_edges_from() (Goulib.graph.DiGraph method), 128
- add_edges_from() (Goulib.graph.GeoGraph method), 112
- add_function() (in module Goulib.expr), 60
- add_module() (in module Goulib.expr), 60
- add_months() (in module Goulib.datetime2), 24
- add_node() (Goulib.graph.DiGraph method), 129
- add_node() (Goulib.graph.GeoGraph method), 113
- add_nodes_from() (Goulib.graph.DiGraph method), 129
- add_nodes_from() (Goulib.graph.GeoGraph method), 113
- add_path() (Goulib.graph.DiGraph method), 129
- add_path() (Goulib.graph.GeoGraph method), 113
- add_star() (Goulib.graph.DiGraph method), 129
- add_star() (Goulib.graph.GeoGraph method), 113
- add_weighted_edges_from() (Goulib.graph.DiGraph method), 129
- add_weighted_edges_from() (Goulib.graph.GeoGraph method), 113
- addCleanup() (Goulib.tests.TestCase method), 248
- addcol() (Goulib.table.Table method), 245
- addcontent() (Goulib.markup.page method), 177
- addfooter() (Goulib.markup.page method), 177
- addheader() (Goulib.markup.page method), 177
- addholidays() (Goulib.workdays.WorkCalendar method), 253
- addTypeEqualityFunc() (Goulib.tests.TestCase method), 248
- adj (Goulib.graph.DiGraph attribute), 129
- adj (Goulib.graph.GeoGraph attribute), 113
- adjacency() (Goulib.graph.DiGraph method), 129
- adjacency() (Goulib.graph.GeoGraph method), 114
- adjlist_inner_dict_factory (Goulib.graph.DiGraph attribute), 129
- adjlist_inner_dict_factory (Goulib.graph.GeoGraph attribute), 114
- adjlist_outer_dict_factory (Goulib.graph.DiGraph attribute), 129
- adjlist_outer_dict_factory (Goulib.graph.GeoGraph attribute), 114
- AGraph (class in Goulib.graph), 103, 109
- all() (in module Goulib.itertools2), 168
- all_pairs() (in module Goulib.itertools2), 168
- alpha_composite() (in module Goulib.image), 151
- alpha_composite_with_color() (in module Goulib.image), 151
- angle() (Goulib.geom.Arc2 method), 83
- angle() (Goulib.geom.Matrix3 method), 94
- angle() (Goulib.geom.Point2 method), 72
- angle() (Goulib.geom.Vector2 method), 69
- angle() (Goulib.geom3d.Vector3 method), 95
- angle() (in module Goulib.math2), 200
- anneal() (in module Goulib.optim), 219
- any() (in module Goulib.itertools2), 168
- append() (Goulib.drawing.BBox method), 34
- append() (Goulib.drawing.Chain method), 29, 46
- append() (Goulib.drawing.Drawing method), 57
- append() (Goulib.drawing.Group method), 27, 39
- append() (Goulib.drawing.Rect method), 51
- append() (Goulib.interval.Box method), 164
- append() (Goulib.interval.Interval method), 158
- append() (Goulib.interval.Intervals method), 161
- append() (Goulib.itertools2.iter2 method), 170
- append() (Goulib.optim.BinList method), 217
- append() (Goulib.piecewise.Piecewise method), 222
- append() (Goulib.stats.Discrete method), 234
- append() (Goulib.stats.Normal method), 238
- append() (Goulib.stats.PDF method), 235
- append() (Goulib.stats.Stats method), 231
- append() (Goulib.table.Table method), 245
- applx() (Goulib.expr.Expr method), 61
- applx() (Goulib.piecewise.Piecewise method), 222
- applx() (Goulib.polynomial.Polynomial method), 227
- applx() (Goulib.stats.Normal method), 238
- applx() (Goulib.stats.PDF method), 235
- apply() (Goulib.container.Sequence method), 13
- apply() (Goulib.expr.Expr method), 61
- apply() (Goulib.piecewise.Piecewise method), 222
- apply() (Goulib.polynomial.Polynomial method), 227
- apply() (Goulib.stats.Normal method), 238
- apply() (Goulib.stats.PDF method), 235
- applyf() (Goulib.table.Table method), 245
- arange() (in module Goulib.itertools2), 166
- Arc2 (class in Goulib.geom), 83
- arc_from_3_points() (in module Goulib.geom), 83
- area (Goulib.drawing.BBox attribute), 25, 32
- argPair() (in module Goulib.geom), 65
- args (Goulib.itertools2.SortingError attribute), 172
- args (Goulib.markup.ArgumentError attribute), 185
- args (Goulib.markup.ClosingError attribute), 183
- args (Goulib.markup.CustomizationError attribute), 190
- args (Goulib.markup.DeprecationError attribute), 187
- args (Goulib.markup.InvalidElementError attribute), 186
- args (Goulib.markup.MarkupError attribute), 182
- args (Goulib.markup.ModeError attribute), 189
- args (Goulib.markup.OpeningError attribute), 184
- ArgumentError, 184

`asdict()` (Goulib.table.Table method), 245
`assert_()` (Goulib.tests.TestCase method), 251
`assertAlmostEqual()` (Goulib.tests.TestCase method), 248
`assertAlmostEquals()` (Goulib.tests.TestCase method), 248
`assertCountEqual()` (Goulib.tests.TestCase method), 247
`assertDictContainsSubset()` (Goulib.tests.TestCase method), 248
`assertDictEqual()` (Goulib.tests.TestCase method), 248
`assertEqual()` (Goulib.tests.TestCase method), 246
`assertEquals()` (Goulib.tests.TestCase method), 248
`assertFalse()` (Goulib.tests.TestCase method), 248
`assertGreater()` (Goulib.tests.TestCase method), 248
`assertGreaterEqual()` (Goulib.tests.TestCase method), 248
`assertIn()` (Goulib.tests.TestCase method), 248
`assertIs()` (Goulib.tests.TestCase method), 248
`assertIsInstance()` (Goulib.tests.TestCase method), 249
`assertIsNone()` (Goulib.tests.TestCase method), 249
`assertIsNot()` (Goulib.tests.TestCase method), 249
`assertIsNotNone()` (Goulib.tests.TestCase method), 249
`assertLess()` (Goulib.tests.TestCase method), 249
`assertLessEqual()` (Goulib.tests.TestCase method), 249
`assertListEqual()` (Goulib.tests.TestCase method), 249
`assertLogs()` (Goulib.tests.TestCase method), 249
`assertMultiLineEqual()` (Goulib.tests.TestCase method), 249
`assertNotAlmostEqual()` (Goulib.tests.TestCase method), 249
`assertNotAlmostEquals()` (Goulib.tests.TestCase method), 249
`assertNotEqual()` (Goulib.tests.TestCase method), 249
`assertNotEquals()` (Goulib.tests.TestCase method), 249
`assertNotIn()` (Goulib.tests.TestCase method), 250
`assertNotIsInstance()` (Goulib.tests.TestCase method), 250
`assertNotRegex()` (Goulib.tests.TestCase method), 250
`assertNotRegexMatches()` (Goulib.tests.TestCase method), 250
`assertRaises()` (Goulib.tests.TestCase method), 250
`assertRaisesRegex()` (Goulib.tests.TestCase method), 250
`assertRaisesRegexp()` (Goulib.tests.TestCase method), 250
`assertRegex()` (Goulib.tests.TestCase method), 250
`assertRegexMatches()` (Goulib.tests.TestCase method), 250
`assertSequenceEqual()` (Goulib.tests.TestCase method), 246
`assertSetEqual()` (Goulib.tests.TestCase method), 250
`assertTrue()` (Goulib.tests.TestCase method), 250
`assertTupleEqual()` (Goulib.tests.TestCase method), 250
`assertWarns()` (Goulib.tests.TestCase method), 251
`assertWarnsRegex()` (Goulib.tests.TestCase method), 251
`astimezone()` (Goulib.datetime2.datetime2 method), 15
`attr()` (in module Goulib.table), 239

`average` (Goulib.stats.Discrete attribute), 234
`average` (Goulib.stats.Normal attribute), 238
`average` (Goulib.stats.PDF attribute), 236
`average` (Goulib.stats.Stats attribute), 231
`average_hash()` (Goulib.image.Image method), 149
`avg` (Goulib.stats.Discrete attribute), 234
`avg` (Goulib.stats.Normal attribute), 238
`avg` (Goulib.stats.PDF attribute), 236
`avg` (Goulib.stats.Stats attribute), 231
`avg()` (in module Goulib.stats), 230

B

`baby_step_giant_step()` (in module Goulib.math2), 202
`base_types` (Goulib.tests.TestCase attribute), 246
`BBox` (class in Goulib.drawing), 25, 31
`bbox()` (Goulib.drawing.Chain method), 48
`bbox()` (Goulib.drawing.Drawing method), 57
`bbox()` (Goulib.drawing.Entity method), 26, 34
`bbox()` (Goulib.drawing.Group method), 41
`bbox()` (Goulib.drawing.Instance method), 44
`bbox()` (Goulib.drawing.Rect method), 51
`bbox()` (Goulib.drawing.Spline method), 27, 37
`bbox()` (Goulib.drawing.Text method), 30, 54
`bbox()` (Goulib.geom.Arc2 method), 85
`bbox()` (Goulib.geom.Circle method), 82
`bbox()` (Goulib.geom.Ellipse method), 87
`bbox()` (Goulib.geom.Point2 method), 72
`bbox()` (Goulib.geom.Segment2 method), 78
`bernouilli()` (in module Goulib.math2), 198
`bernouilli_gen()` (in module Goulib.math2), 198
`best()` (in module Goulib.itertools2), 169
`bigomega()` (in module Goulib.math2), 196
`BinDict` (class in Goulib.optim), 215
`BinList` (class in Goulib.optim), 217
`binomial()` (in module Goulib.math2), 200
`binomial_exponent()` (in module Goulib.math2), 200
`bisect()` (Goulib.geom.Segment2 method), 77
`bisect()` (Goulib.interval.Intervals method), 161
`bisect_key()` (Goulib.interval.Intervals method), 161
`bisect_key_left()` (Goulib.interval.Intervals method), 161
`bisect_key_right()` (Goulib.interval.Intervals method), 161
`bisect_left()` (Goulib.interval.Intervals method), 161
`bisect_right()` (Goulib.interval.Intervals method), 161
`bool2gray()` (in module Goulib.image), 155
`bool2rgb()` (in module Goulib.image), 155
`bouncy()` (in module Goulib.math2), 198
`Box` (class in Goulib.interval), 163
`box()` (Goulib.graph.DiGraph method), 129
`box()` (Goulib.graph.GeoGraph method), 114
`box_size()` (Goulib.graph.DiGraph method), 129
`box_size()` (Goulib.graph.GeoGraph method), 114

C

- `carries()` (in module `Goulib.math2`), 197
- `cartesian_product()` (in module `Goulib.itertools2`), 168
- `cast()` (`Goulib.workdays.WorkCalendar` method), 253
- `catalan()` (in module `Goulib.math2`), 194
- `catalan_gen()` (in module `Goulib.math2`), 194
- `ceildiv()` (in module `Goulib.math2`), 191
- `Cell` (class in `Goulib.table`), 239
- `center` (`Goulib.drawing.Chain` attribute), 48
- `center` (`Goulib.drawing.Drawing` attribute), 57
- `center` (`Goulib.drawing.Entity` attribute), 26, 34
- `center` (`Goulib.drawing.Group` attribute), 41
- `center` (`Goulib.drawing.Instance` attribute), 44
- `center` (`Goulib.drawing.Rect` attribute), 51
- `center` (`Goulib.drawing.Spline` attribute), 38
- `center` (`Goulib.drawing.Text` attribute), 55
- `center` (`Goulib.geom.Arc2` attribute), 85
- `center` (`Goulib.geom.Circle` attribute), 82
- `center` (`Goulib.geom.Ellipse` attribute), 87
- `center` (`Goulib.geom.Point2` attribute), 72
- `center` (`Goulib.geom.Segment2` attribute), 78
- `center` (`Goulib.interval.Box` attribute), 163
- `center` (`Goulib.interval.Interval` attribute), 157
- `center()` (`Goulib.drawing.BBox` method), 25, 32
- `cgiprint()` (in module `Goulib.markup`), 174
- `Chain` (class in `Goulib.drawing`), 28, 45
- `chainify()` (`Goulib.drawing.Chain` method), 48
- `chainify()` (`Goulib.drawing.Drawing` method), 57
- `chainify()` (`Goulib.drawing.Group` method), 27, 39
- `chainify()` (`Goulib.drawing.Rect` method), 51
- `chains()` (in module `Goulib.drawing`), 29, 49
- `chakravala()` (in module `Goulib.math2`), 200
- `checkOnTimeAndWait()` (`Goulib.statemachine.StateMachine` method), 230
- `chinese_remainder()` (in module `Goulib.math2`), 201
- `Circle` (class in `Goulib.geom`), 80
- `circle_from_3_points()` (in module `Goulib.geom`), 83
- `clear()` (`Goulib.colors.Palette` method), 10
- `clear()` (`Goulib.container.Record` method), 12
- `clear()` (`Goulib.drawing.BBox` method), 34
- `clear()` (`Goulib.drawing.Chain` method), 48
- `clear()` (`Goulib.drawing.Drawing` method), 57
- `clear()` (`Goulib.drawing.Group` method), 41
- `clear()` (`Goulib.drawing.Rect` method), 51
- `clear()` (`Goulib.graph.DiGraph` method), 130
- `clear()` (`Goulib.graph.GeoGraph` method), 114
- `clear()` (`Goulib.graph.index.Index` method), 108
- `clear()` (`Goulib.interval.Box` method), 164
- `clear()` (`Goulib.interval.Interval` method), 158
- `clear()` (`Goulib.interval.Intervals` method), 162
- `clear()` (`Goulib.optim.BinDict` method), 217
- `clear()` (`Goulib.optim.BinList` method), 219
- `clear()` (`Goulib.table.Table` method), 245
- `close()` (`Goulib.markup.element` method), 174
- `closest_edges()` (`Goulib.graph.DiGraph` method), 130
- `closest_edges()` (`Goulib.graph.GeoGraph` method), 114
- `closest_nodes()` (`Goulib.graph.DiGraph` method), 130
- `closest_nodes()` (`Goulib.graph.GeoGraph` method), 114
- `ClosingError`, 182
- `cmp()` (in module `Goulib.math2`), 190
- `cmyk` (`Goulib.colors.Color` attribute), 7
- `cmyk2rgb()` (in module `Goulib.colors`), 6
- `cmyk2rgb()` (in module `Goulib.image`), 155
- `col()` (`Goulib.table.Table` method), 243
- `Color` (class in `Goulib.colors`), 6
- `color` (`Goulib.drawing.Chain` attribute), 48
- `color` (`Goulib.drawing.Drawing` attribute), 57
- `color` (`Goulib.drawing.Entity` attribute), 26, 34
- `color` (`Goulib.drawing.Group` attribute), 27, 39
- `color` (`Goulib.drawing.Instance` attribute), 44
- `color` (`Goulib.drawing.Rect` attribute), 51
- `color` (`Goulib.drawing.Spline` attribute), 38
- `color` (`Goulib.drawing.Text` attribute), 55
- `color` (`Goulib.geom.Arc2` attribute), 85
- `color` (`Goulib.geom.Circle` attribute), 82
- `color` (`Goulib.geom.Ellipse` attribute), 87
- `color` (`Goulib.geom.Point2` attribute), 72
- `color` (`Goulib.geom.Segment2` attribute), 78
- `color_range()` (in module `Goulib.colors`), 10
- `color_to_aci()` (in module `Goulib.colors`), 10
- `colorize()` (`Goulib.image.Image` method), 150
- `ColorTable()` (in module `Goulib.colors`), 10
- `cols()` (`Goulib.table.Table` method), 243
- `combinations_with_replacement()` (in module `Goulib.itertools2`), 168
- `combine()` (`Goulib.datetime2.datetime2` method), 16
- `compact()` (in module `Goulib.itertools2`), 166
- `complexity()` (`Goulib.expr.Expr` method), 61
- `complexity()` (`Goulib.pieewise.Pieewise` method), 223
- `complexity()` (`Goulib.polynomial.Polynomial` method), 227
- `complexity()` (`Goulib.stats.Normal` method), 238
- `complexity()` (`Goulib.stats.PDF` method), 236
- `compose()` (`Goulib.colors.Color` method), 7
- `compose()` (`Goulib.image.Image` method), 150
- `compose()` (in module `Goulib.itertools2`), 167
- `compress()` (in module `Goulib.itertools2`), 166
- `confidence_interval()` (in module `Goulib.stats`), 230
- `conjugated()` (`Goulib.geom3d.Quaternion` method), 102
- `connect()` (`Goulib.drawing.Chain` method), 48
- `connect()` (`Goulib.drawing.Drawing` method), 58
- `connect()` (`Goulib.drawing.Group` method), 41
- `connect()` (`Goulib.drawing.Instance` method), 44
- `connect()` (`Goulib.drawing.Rect` method), 51
- `connect()` (`Goulib.drawing.Spline` method), 38
- `connect()` (`Goulib.geom.Arc2` method), 85
- `connect()` (`Goulib.geom.Circle` method), 81
- `connect()` (`Goulib.geom.Ellipse` method), 87

`connect()` (Goulib.geom.Geometry method), 64
`connect()` (Goulib.geom.Line2 method), 75
`connect()` (Goulib.geom.Point2 method), 71
`connect()` (Goulib.geom.Ray2 method), 77
`connect()` (Goulib.geom.Segment2 method), 78
`connect()` (Goulib.geom3d.Line3 method), 97
`connect()` (Goulib.geom3d.Plane method), 99
`connect()` (Goulib.geom3d.Point3 method), 96
`connect()` (Goulib.geom3d.Sphere method), 98
`contiguity()` (Goulib.graph.DiGraph method), 130
`contiguity()` (Goulib.graph.GeoGraph method), 114
`contiguous()` (Goulib.drawing.Chain method), 29, 46
`contiguous()` (Goulib.drawing.Rect method), 51
`convert()` (Goulib.colors.Color method), 7
`convert()` (Goulib.image.Image method), 149
`convert()` (in module Goulib.colors), 6
`convert()` (in module Goulib.image), 156
`converter()` (in module Goulib.colors), 6
`coprime()` (in module Goulib.math2), 190
`coprimes_gen()` (in module Goulib.math2), 190
`copy()` (Goulib.colors.Palette method), 10
`copy()` (Goulib.container.Record method), 12
`copy()` (Goulib.drawing.BBox method), 34
`copy()` (Goulib.drawing.Chain method), 48
`copy()` (Goulib.drawing.Drawing method), 58
`copy()` (Goulib.drawing.Group method), 41
`copy()` (Goulib.drawing.Rect method), 51
`copy()` (Goulib.graph.DiGraph method), 130
`copy()` (Goulib.graph.GeoGraph method), 114
`copy()` (Goulib.graph.index.Index method), 108
`copy()` (Goulib.interval.Box method), 164
`copy()` (Goulib.interval.Interval method), 159
`copy()` (Goulib.interval.Intervals method), 162
`copy()` (Goulib.optim.BinDict method), 217
`copy()` (Goulib.optim.BinList method), 219
`copy()` (Goulib.table.Table method), 245
`corner()` (Goulib.drawing.BBox method), 34
`corner()` (Goulib.interval.Box method), 163
`corr()` (Goulib.stats.Normal method), 237
`correlation()` (Goulib.image.Image method), 150
`correlation()` (Goulib.stats.Normal method), 237
`count()` (Goulib.drawing.BBox method), 34
`count()` (Goulib.drawing.Chain method), 48
`count()` (Goulib.drawing.Drawing method), 58
`count()` (Goulib.drawing.Group method), 41
`count()` (Goulib.drawing.Rect method), 52
`count()` (Goulib.graph.index.Index method), 103, 107
`count()` (Goulib.interval.Box method), 165
`count()` (Goulib.interval.Interval method), 159
`count()` (Goulib.interval.Intervals method), 162
`count()` (Goulib.optim.BinList method), 219
`count()` (Goulib.table.Table method), 245
`count_unique()` (in module Goulib.itertools2), 168
`countTestCases()` (Goulib.tests.TestCase method), 251

`cousin_primes()` (in module Goulib.math2), 196
`cov()` (Goulib.stats.Normal method), 237
`covariance()` (Goulib.stats.Discrete method), 234
`covariance()` (Goulib.stats.Normal method), 237
`covariance()` (Goulib.stats.PDF method), 236
`covariance()` (Goulib.stats.Stats method), 231
`covariance()` (in module Goulib.stats), 231
`crop()` (Goulib.image.Image method), 148
`cross()` (Goulib.geom.Point2 method), 72
`cross()` (Goulib.geom.Vector2 method), 69
`cross()` (Goulib.geom3d.Vector3 method), 95
`css()` (Goulib.markup.page method), 177
`ctime()` (Goulib.datetime2.date2 method), 18
`ctime()` (Goulib.datetime2.datetime2 method), 16
`cumsum()` (in module Goulib.math2), 192
`CustomizationError`, 189

D

`date()` (Goulib.datetime2.datetime2 method), 16
`date2` (class in Goulib.datetime2), 17
`date_add()` (in module Goulib.datetime2), 24
`datef()` (in module Goulib.datetime2), 23
`datetime2` (class in Goulib.datetime2), 14
`datetime_intersect()` (in module Goulib.datetime2), 24
`datetimef()` (in module Goulib.datetime2), 23
`day` (Goulib.datetime2.date2 attribute), 18
`day` (Goulib.datetime2.datetime2 attribute), 16
`days` (Goulib.datetime2.timedelta2 attribute), 22
`days()` (in module Goulib.datetime2), 24
`daysgen()` (in module Goulib.datetime2), 24
`de_bruijn()` (in module Goulib.math2), 201
`debug()` (Goulib.tests.TestCase method), 251
`debug()` (in module Goulib.decorators), 24
`defaultTestResult()` (Goulib.tests.TestCase method), 251
`degree` (Goulib.graph.DiGraph attribute), 130
`degree` (Goulib.graph.GeoGraph attribute), 114
`delauney_triangulation()` (in module Goulib.graph), 105, 145
`delete()` (Goulib.graph.index.Index method), 103, 107
`deltaE()` (Goulib.colors.Color method), 7
`deltaE()` (Goulib.image.Image method), 150
`deltaE()` (in module Goulib.colors), 10
`DeprecationError`, 186
`derivative()` (Goulib.polynomial.Polynomial method), 226
`derivative()` (in module Goulib.polynomial), 228
`detect_cycle()` (in module Goulib.itertools2), 174
`determinant()` (Goulib.geom.Matrix3 method), 94
`determinant()` (Goulib.geom3d.Matrix4 method), 100
`diag()` (in module Goulib.math2), 192
`dictsplit()` (in module Goulib.itertools2), 169
`diff()` (Goulib.workdays.WorkCalendar method), 253
`diff()` (in module Goulib.itertools2), 172
`DifferentialEvolution` (class in Goulib.optim), 220
`digits()` (in module Goulib.math2), 196

digits_gen() (in module Goulib.math2), 196
 DiGraph (class in Goulib.graph), 104, 126
 digsum() (in module Goulib.math2), 196
 discard() (Goulib.interval.Intervals method), 162
 Discrete (class in Goulib.stats), 232
 disk() (in module Goulib.image), 152
 display() (Goulib.motion.Actuator method), 210
 display() (Goulib.statemachine.StateMachine method), 230
 displayGraph() (Goulib.statemachine.StateMachine method), 230
 displayLast() (Goulib.motion.Actuator method), 210
 dist() (Goulib.graph.DiGraph method), 130
 dist() (Goulib.graph.GeoGraph method), 115
 dist() (Goulib.image.Image method), 149
 dist() (in module Goulib.math2), 193
 distance() (Goulib.drawing.Chain method), 48
 distance() (Goulib.drawing.Drawing method), 58
 distance() (Goulib.drawing.Group method), 41
 distance() (Goulib.drawing.Instance method), 44
 distance() (Goulib.drawing.Rect method), 52
 distance() (Goulib.drawing.Spline method), 38
 distance() (Goulib.geom.Arc2 method), 85
 distance() (Goulib.geom.Circle method), 82
 distance() (Goulib.geom.Ellipse method), 87
 distance() (Goulib.geom.Geometry method), 64
 distance() (Goulib.geom.Line2 method), 76
 distance() (Goulib.geom.Point2 method), 70
 distance() (Goulib.geom.Ray2 method), 77
 distance() (Goulib.geom.Segment2 method), 78
 distance_on_sphere() (Goulib.geom3d.Sphere method), 98
 dither() (Goulib.image.Image method), 150
 dither() (in module Goulib.image), 155
 Ditherer (class in Goulib.image), 152
 divisors() (in module Goulib.math2), 195
 doCleanups() (Goulib.tests.TestCase method), 251
 dot() (Goulib.geom.Point2 method), 73
 dot() (Goulib.geom.Vector2 method), 69
 dot() (Goulib.geom3d.Vector3 method), 95
 dot() (in module Goulib.math2), 192
 dot_mm() (in module Goulib.math2), 192
 dot_mv() (in module Goulib.math2), 192
 dot_vv() (in module Goulib.math2), 192
 draw() (Goulib.drawing.Chain method), 48
 draw() (Goulib.drawing.Drawing method), 58
 draw() (Goulib.drawing.Entity method), 27, 35
 draw() (Goulib.drawing.Group method), 41
 draw() (Goulib.drawing.Instance method), 44
 draw() (Goulib.drawing.Rect method), 52
 draw() (Goulib.drawing.Spline method), 38
 draw() (Goulib.drawing.Text method), 55
 draw() (Goulib.geom.Arc2 method), 85
 draw() (Goulib.geom.Circle method), 82

draw() (Goulib.geom.Ellipse method), 87
 draw() (Goulib.geom.Point2 method), 73
 draw() (Goulib.geom.Segment2 method), 78
 draw() (Goulib.graph.DiGraph method), 130
 draw() (Goulib.graph.GeoGraph method), 115
 draw_networkx() (in module Goulib.graph), 104, 144
 Drawing (class in Goulib.drawing), 30, 56
 drop() (in module Goulib.itertools2), 165
 dst() (Goulib.datetime2.datetime2 method), 16
 dst() (Goulib.datetime2.time2 method), 20
 dt() (Goulib.motion.Segment method), 203
 dt() (Goulib.motion.SegmentPoly method), 208
 dt() (Goulib.motion.Segments method), 206
 dummy (class in Goulib.markup), 178

E

edge_attr_dict_factory (Goulib.graph.DiGraph attribute), 130
 edge_attr_dict_factory (Goulib.graph.GeoGraph attribute), 115
 edge_key_dict_factory (Goulib.graph.DiGraph attribute), 130
 edge_key_dict_factory (Goulib.graph.GeoGraph attribute), 115
 edge_subgraph() (Goulib.graph.DiGraph method), 130
 edge_subgraph() (Goulib.graph.GeoGraph method), 115
 edges (Goulib.graph.DiGraph attribute), 131
 edges (Goulib.graph.GeoGraph attribute), 115
 element (class in Goulib.markup), 174
 Ellipse (class in Goulib.geom), 86
 empty() (Goulib.drawing.BBox method), 34
 empty() (Goulib.interval.Box method), 163
 empty() (Goulib.interval.Interval method), 157
 end (Goulib.drawing.BBox attribute), 34
 end (Goulib.drawing.Chain attribute), 28, 46
 end (Goulib.drawing.Drawing attribute), 58
 end (Goulib.drawing.Entity attribute), 26, 34
 end (Goulib.drawing.Group attribute), 41
 end (Goulib.drawing.Instance attribute), 44
 end (Goulib.drawing.Rect attribute), 52
 end (Goulib.drawing.Spline attribute), 27, 37
 end (Goulib.drawing.Text attribute), 55
 end (Goulib.geom.Arc2 attribute), 85
 end (Goulib.geom.Circle attribute), 82
 end (Goulib.geom.Ellipse attribute), 87
 end (Goulib.geom.Point2 attribute), 73
 end (Goulib.geom.Segment2 attribute), 79
 end (Goulib.interval.Box attribute), 163
 end (Goulib.interval.Interval attribute), 157
 end (Goulib.workdays.WorkCalendar attribute), 253
 end() (Goulib.motion.Segment method), 204
 end() (Goulib.motion.SegmentPoly method), 208
 end() (Goulib.motion.Segments method), 205
 endAcc() (Goulib.motion.Segment method), 204

endAcc() (Goulib.motion.SegmentPoly method), 208
endAcc() (Goulib.motion.Segments method), 206
endJerk() (Goulib.motion.Segment method), 204
endJerk() (Goulib.motion.SegmentPoly method), 208
endJerk() (Goulib.motion.Segments method), 206
endPos() (Goulib.motion.Segment method), 204
endPos() (Goulib.motion.SegmentPoly method), 208
endPos() (Goulib.motion.Segments method), 206
endSpeed() (Goulib.motion.Segment method), 204
endSpeed() (Goulib.motion.SegmentPoly method), 208
endSpeed() (Goulib.motion.Segments method), 206
endTime() (Goulib.motion.Actuator method), 210
endTime() (Goulib.motion.Segment method), 204
endTime() (Goulib.motion.SegmentPoly method), 208
endTime() (Goulib.motion.Segments method), 206
ensure_sorted() (in module Goulib.itertools2), 172
Entity (class in Goulib.drawing), 26, 34
enumerates() (in module Goulib.itertools2), 166
equal() (in module Goulib.datetime2), 24
ErrorDiffusion (class in Goulib.image), 153
escape() (in module Goulib.markup), 178
euclid_gen() (in module Goulib.math2), 195
euclidean_minimum_spanning_tree() (in module Goulib.graph), 105, 145
euler_phi() (in module Goulib.math2), 196
eval() (in module Goulib.expr), 60
EventLog (class in Goulib.statemachine), 229
every() (in module Goulib.itertools2), 165
evolve() (Goulib.optim.DifferentialEvolution method), 222
expand() (Goulib.image.Image method), 150
Expr (class in Goulib.expr), 60
extend() (Goulib.drawing.BBox method), 34
extend() (Goulib.drawing.Chain method), 48
extend() (Goulib.drawing.Drawing method), 58
extend() (Goulib.drawing.Group method), 27, 39
extend() (Goulib.drawing.Rect method), 52
extend() (Goulib.interval.Box method), 165
extend() (Goulib.interval.Interval method), 159
extend() (Goulib.interval.Intervals method), 162
extend() (Goulib.optim.BinList method), 217
extend() (Goulib.piecewise.Piecewise method), 222
extend() (Goulib.stats.Discrete method), 234
extend() (Goulib.stats.Normal method), 238
extend() (Goulib.stats.PDF method), 236
extend() (Goulib.stats.Stats method), 231
extend() (Goulib.table.Table method), 245
eye() (in module Goulib.math2), 192

F

factorial_gen() (in module Goulib.math2), 200
factorize() (in module Goulib.math2), 195
factors() (in module Goulib.math2), 195
fail() (Goulib.tests.TestCase method), 251

failIf() (Goulib.tests.TestCase method), 251
failIfAlmostEqual() (Goulib.tests.TestCase method), 251
failIfEqual() (Goulib.tests.TestCase method), 251
failUnless() (Goulib.tests.TestCase method), 251
failUnlessAlmostEqual() (Goulib.tests.TestCase method), 252
failUnlessEqual() (Goulib.tests.TestCase method), 252
failUnlessRaises() (Goulib.tests.TestCase method), 252
failureException (Goulib.tests.TestCase attribute), 252
faulhaber() (in module Goulib.math2), 198
fibonacci() (in module Goulib.math2), 194
fibonacci_gen() (in module Goulib.math2), 194
fig2img() (in module Goulib.image), 152
figure() (Goulib.drawing.Chain method), 48
figure() (Goulib.drawing.Drawing method), 58
figure() (Goulib.drawing.Entity static method), 27, 35
figure() (Goulib.drawing.Group method), 41
figure() (Goulib.drawing.Instance method), 44
figure() (Goulib.drawing.Rect method), 52
figure() (Goulib.drawing.Spline method), 38
figure() (Goulib.drawing.Text method), 55
figure() (Goulib.geom.Arc2 method), 85
figure() (Goulib.geom.Circle method), 82
figure() (Goulib.geom.Ellipse method), 87
figure() (Goulib.geom.Point2 method), 73
figure() (Goulib.geom.Segment2 method), 79
figure() (in module Goulib.graph), 104, 144
filter() (Goulib.container.Sequence method), 13
filter() (Goulib.image.Image method), 150
filter2() (in module Goulib.itertools2), 169
find() (in module Goulib.itertools2), 169
find_col() (Goulib.table.Table method), 243
first() (in module Goulib.itertools2), 165
first_fit_decreasing() (in module Goulib.optim), 219
first_match() (in module Goulib.itertools2), 173
fits() (Goulib.optim.BinDict method), 217
fits() (Goulib.optim.BinList method), 219
flatten() (in module Goulib.itertools2), 166
flip() (Goulib.image.Image method), 148
floyd() (in module Goulib.itertools2), 173
FloydSteinberg (class in Goulib.image), 154
fmt2regex() (in module Goulib.datetime2), 23
format() (in module Goulib.math2), 191
fresh_copy() (Goulib.graph.DiGraph method), 132
fresh_copy() (Goulib.graph.GeoGraph method), 116
FRI (Goulib.workdays.WorkCalendar attribute), 254
from_dxf() (Goulib.drawing.Chain static method), 29, 46
from_dxf() (Goulib.drawing.Drawing method), 58
from_dxf() (Goulib.drawing.Entity static method), 26, 35
from_dxf() (Goulib.drawing.Group method), 28, 39
from_dxf() (Goulib.drawing.Instance static method), 28, 43
from_dxf() (Goulib.drawing.Rect method), 52
from_dxf() (Goulib.drawing.Spline method), 38

[from_dxf\(\)](#) (Goulib.drawing.Text method), 55
[from_dxf\(\)](#) (Goulib.geom.Arc2 method), 85
[from_dxf\(\)](#) (Goulib.geom.Circle method), 82
[from_dxf\(\)](#) (Goulib.geom.Ellipse method), 87
[from_dxf\(\)](#) (Goulib.geom.Point2 method), 73
[from_dxf\(\)](#) (Goulib.geom.Segment2 method), 79
[from_pdf\(\)](#) (Goulib.drawing.Chain static method), 29, 46
[from_pdf\(\)](#) (Goulib.drawing.Drawing method), 58
[from_pdf\(\)](#) (Goulib.drawing.Entity static method), 26, 35
[from_pdf\(\)](#) (Goulib.drawing.Group method), 42
[from_pdf\(\)](#) (Goulib.drawing.Instance method), 45
[from_pdf\(\)](#) (Goulib.drawing.Rect method), 52
[from_pdf\(\)](#) (Goulib.drawing.Spline method), 38
[from_pdf\(\)](#) (Goulib.drawing.Text method), 55
[from_pdf\(\)](#) (Goulib.geom.Arc2 method), 85
[from_pdf\(\)](#) (Goulib.geom.Circle method), 82
[from_pdf\(\)](#) (Goulib.geom.Ellipse method), 88
[from_pdf\(\)](#) (Goulib.geom.Point2 method), 73
[from_pdf\(\)](#) (Goulib.geom.Segment2 method), 79
[from_svg\(\)](#) (Goulib.drawing.Chain static method), 29, 46
[from_svg\(\)](#) (Goulib.drawing.Drawing method), 58
[from_svg\(\)](#) (Goulib.drawing.Entity static method), 26, 35
[from_svg\(\)](#) (Goulib.drawing.Group method), 42
[from_svg\(\)](#) (Goulib.drawing.Instance method), 45
[from_svg\(\)](#) (Goulib.drawing.Rect method), 52
[from_svg\(\)](#) (Goulib.drawing.Spline method), 38
[from_svg\(\)](#) (Goulib.drawing.Text method), 55
[from_svg\(\)](#) (Goulib.geom.Arc2 method), 85
[from_svg\(\)](#) (Goulib.geom.Circle method), 82
[from_svg\(\)](#) (Goulib.geom.Ellipse method), 88
[from_svg\(\)](#) (Goulib.geom.Point2 method), 73
[from_svg\(\)](#) (Goulib.geom.Segment2 method), 79
[fromkeys\(\)](#) (Goulib.colors.Palette method), 10
[fromkeys\(\)](#) (Goulib.container.Record method), 12
[fromkeys\(\)](#) (Goulib.graph.index.Index method), 108
[fromkeys\(\)](#) (Goulib.optim.BinDict method), 217
[fromordinal\(\)](#) (Goulib.datetime2.date2 method), 18
[fromordinal\(\)](#) (Goulib.datetime2.datetime2 method), 16
[fromtimestamp\(\)](#) (Goulib.datetime2.date2 method), 18
[fromtimestamp\(\)](#) (Goulib.datetime2.datetime2 method), 16
[fspecial\(\)](#) (in module Goulib.image), 152
[FullTime](#) (in module Goulib.workdays), 255

G

[gcd\(\)](#) (in module Goulib.math2), 190
[generic_visit\(\)](#) (Goulib.expr.TextVisitor method), 63
[GeoGraph](#) (class in Goulib.graph), 104, 110
[Geometry](#) (class in Goulib.geom), 64
[get\(\)](#) (Goulib.colors.Palette method), 10
[get\(\)](#) (Goulib.container.Record method), 12
[get\(\)](#) (Goulib.graph.index.Index method), 108
[get\(\)](#) (Goulib.optim.BinDict method), 217
[get\(\)](#) (Goulib.table.Table method), 243

[get_angle_axis\(\)](#) (Goulib.geom3d.Quaternion method), 102
[get_cardinal_name\(\)](#) (in module Goulib.math2), 200
[get_edge_data\(\)](#) (Goulib.graph.DiGraph method), 132
[get_edge_data\(\)](#) (Goulib.graph.GeoGraph method), 116
[get_euler\(\)](#) (Goulib.geom3d.Quaternion method), 103
[get_function_source\(\)](#) (in module Goulib.expr), 60
[get_matrix\(\)](#) (Goulib.geom3d.Quaternion method), 103
[get_thread_pool\(\)](#) (in module Goulib.decorators), 24
[getcolors\(\)](#) (Goulib.image.Image method), 147
[getdata\(\)](#) (Goulib.image.Image method), 147
[gethours\(\)](#) (Goulib.workdays.WorkCalendar method), 253
[getpalette\(\)](#) (Goulib.image.Image method), 147
[getpixel\(\)](#) (Goulib.image.Image method), 147
[Goulib.colors](#) (module), 6
[Goulib.container](#) (module), 11
[Goulib.datetime2](#) (module), 14
[Goulib.decorators](#) (module), 24
[Goulib.drawing](#) (module), 24, 31
[Goulib.expr](#) (module), 60
[Goulib.geom](#) (module), 64
[Goulib.geom3d](#) (module), 94
[Goulib.graph](#) (module), 103, 105
[Goulib.image](#) (module), 145
[Goulib.interval](#) (module), 156
[Goulib.itertools2](#) (module), 165
[Goulib.markup](#) (module), 174
[Goulib.math2](#) (module), 190
[Goulib.motion](#) (module), 202
[Goulib.optim](#) (module), 214
[Goulib.piecewise](#) (module), 222
[Goulib.plot](#) (module), 224
[Goulib.polynomial](#) (module), 225
[Goulib.statemachine](#) (module), 228
[Goulib.stats](#) (module), 230
[Goulib.table](#) (module), 239
[Goulib.tests](#) (module), 246
[Goulib.workdays](#) (module), 253
[gray2bool\(\)](#) (in module Goulib.image), 155
[gray2rgb\(\)](#) (in module Goulib.image), 155
[grayscale\(\)](#) (Goulib.image.Image method), 150
[Group](#) (class in Goulib.drawing), 27, 39
[groupby\(\)](#) (Goulib.table.Table method), 245
[groupby_gen\(\)](#) (Goulib.table.Table method), 245
[groups\(\)](#) (in module Goulib.itertools2), 167
[guessmode\(\)](#) (in module Goulib.image), 146

H

[hamming\(\)](#) (in module Goulib.math2), 193
[has_edge\(\)](#) (Goulib.graph.DiGraph method), 133
[has_edge\(\)](#) (Goulib.graph.GeoGraph method), 117
[has_node\(\)](#) (Goulib.graph.DiGraph method), 133
[has_node\(\)](#) (Goulib.graph.GeoGraph method), 118

`has_predecessor()` (Goulib.graph.DiGraph method), 133
`has_successor()` (Goulib.graph.DiGraph method), 134
`height` (Goulib.drawing.BBox attribute), 25, 32
`heptagonal()` (in module Goulib.math2), 199
`herror()` (Goulib.statemachine.StateMachine method), 230
`hex` (Goulib.colors.Color attribute), 7
`hex2rgb()` (in module Goulib.colors), 6
`hexagonal()` (in module Goulib.math2), 199
`hierarchy()` (Goulib.table.Table method), 245
`hillclimb()` (in module Goulib.optim), 219
`hillclimb_and_restart()` (in module Goulib.optim), 219
`hinfo()` (Goulib.statemachine.StateMachine method), 230
`hour` (Goulib.datetime2.datetime2 attribute), 16
`hour` (Goulib.datetime2.time2 attribute), 20
`hours()` (in module Goulib.datetime2), 23
`hsuccess()` (Goulib.statemachine.StateMachine method), 230
`hsv` (Goulib.colors.Color attribute), 7
`html()` (Goulib.drawing.Chain method), 48
`html()` (Goulib.drawing.Drawing method), 58
`html()` (Goulib.drawing.Entity method), 36
`html()` (Goulib.drawing.Group method), 42
`html()` (Goulib.drawing.Instance method), 45
`html()` (Goulib.drawing.Rect method), 52
`html()` (Goulib.drawing.Spline method), 38
`html()` (Goulib.drawing.Text method), 55
`html()` (Goulib.expr.Expr method), 62
`html()` (Goulib.geom.Arc2 method), 85
`html()` (Goulib.geom.Circle method), 82
`html()` (Goulib.geom.Ellipse method), 88
`html()` (Goulib.geom.Point2 method), 73
`html()` (Goulib.geom.Segment2 method), 79
`html()` (Goulib.graph.DiGraph method), 134
`html()` (Goulib.graph.GeoGraph method), 118
`html()` (Goulib.image.Image method), 151
`html()` (Goulib.motion.PVA method), 203
`html()` (Goulib.motion.Segment method), 205
`html()` (Goulib.motion.SegmentPoly method), 208
`html()` (Goulib.motion.Segments method), 205
`html()` (Goulib.motion.TimeDiagram method), 212
`html()` (Goulib.piecewise.Piecewise method), 223
`html()` (Goulib.plot.Plot method), 224
`html()` (Goulib.polynomial.Polynomial method), 227
`html()` (Goulib.stats.Normal method), 238
`html()` (Goulib.stats.PDF method), 236
`html()` (Goulib.table.Cell method), 240
`html()` (Goulib.table.Row method), 241
`html()` (Goulib.table.Table method), 242
`hull()` (Goulib.interval.Interval method), 157
`hwarning()` (Goulib.statemachine.StateMachine method), 230

I
`iapply()` (Goulib.piecewise.Piecewise method), 222
`icol()` (Goulib.table.Table method), 243
`icross()` (in module Goulib.itertools2), 168
`id()` (Goulib.tests.TestCase method), 252
`identity()` (Goulib.geom.Matrix3 method), 93
`identity()` (Goulib.geom3d.Matrix4 method), 99
`identity()` (Goulib.geom3d.Quaternion method), 102
`identity()` (in module Goulib.itertools2), 168
`identity()` (in module Goulib.math2), 192
`ifind()` (in module Goulib.itertools2), 169
`ilen()` (in module Goulib.itertools2), 165
`ilog()` (in module Goulib.math2), 200
`Image` (class in Goulib.image), 146
`in_degree` (Goulib.graph.DiGraph attribute), 134
`in_edges` (Goulib.graph.DiGraph attribute), 134
`in_interval()` (in module Goulib.interval), 156
`ind2any()` (in module Goulib.image), 156
`ind2rgb()` (in module Goulib.image), 156
`index` (class in Goulib.graph), 103, 105
`index()` (Goulib.colors.Palette method), 8
`index()` (Goulib.container.Sequence method), 13
`index()` (Goulib.drawing.BBox method), 34
`index()` (Goulib.drawing.Chain method), 48
`index()` (Goulib.drawing.Drawing method), 58
`index()` (Goulib.drawing.Group method), 42
`index()` (Goulib.drawing.Rect method), 52
`index()` (Goulib.interval.Box method), 165
`index()` (Goulib.interval.Interval method), 159
`index()` (Goulib.interval.Intervals method), 162
`index()` (Goulib.optim.BinList method), 219
`index()` (Goulib.piecewise.Piecewise method), 222
`index()` (Goulib.table.Table method), 243
`index()` (in module Goulib.itertools2), 165
`index.Index` (class in Goulib.graph), 103, 106
`index.Property` (class in Goulib.graph), 103, 105
`index_max()` (in module Goulib.itertools2), 169
`index_min()` (in module Goulib.itertools2), 169
`init()` (Goulib.markup.page method), 177
`init__()` (Goulib.datetime2.date2 method), 17
`insert()` (Goulib.drawing.BBox method), 34
`insert()` (Goulib.drawing.Chain method), 48
`insert()` (Goulib.drawing.Drawing method), 58
`insert()` (Goulib.drawing.Group method), 42
`insert()` (Goulib.drawing.Rect method), 52
`insert()` (Goulib.graph.index.Index method), 103, 107
`insert()` (Goulib.interval.Box method), 165
`insert()` (Goulib.interval.Interval method), 159
`insert()` (Goulib.interval.Intervals method), 159
`insert()` (Goulib.itertools2.iter2 method), 170
`insert()` (Goulib.motion.Segments method), 205
`insert()` (Goulib.optim.BinList method), 217
`insert()` (Goulib.table.Table method), 245
`Instance` (class in Goulib.drawing), 28, 43

int_or_float() (in module Goulib.math2), 191
 integer_exponent() (in module Goulib.math2), 197
 integral() (Goulib.polynomial.Polynomial method), 226
 integral() (in module Goulib.polynomial), 228
 interleave() (in module Goulib.itertools2), 168
 intersect() (Goulib.drawing.Chain method), 48
 intersect() (Goulib.drawing.Drawing method), 58
 intersect() (Goulib.drawing.Group method), 42
 intersect() (Goulib.drawing.Instance method), 45
 intersect() (Goulib.drawing.Rect method), 52
 intersect() (Goulib.drawing.Spline method), 38
 intersect() (Goulib.drawing.Text method), 30, 54
 intersect() (Goulib.geom.Arc2 method), 84
 intersect() (Goulib.geom.Circle method), 81
 intersect() (Goulib.geom.Ellipse method), 88
 intersect() (Goulib.geom.Geometry method), 64
 intersect() (Goulib.geom.Line2 method), 75
 intersect() (Goulib.geom.Point2 method), 71
 intersect() (Goulib.geom.Ray2 method), 77
 intersect() (Goulib.geom.Segment2 method), 79
 intersect() (Goulib.geom3d.Line3 method), 97
 intersect() (Goulib.geom3d.Plane method), 99
 intersect() (Goulib.geom3d.Point3 method), 96
 intersect() (Goulib.geom3d.Sphere method), 98
 intersect() (in module Goulib.interval), 156
 intersect() (in module Goulib.itertools2), 172
 intersection() (Goulib.interval.Interval method), 157
 intersection() (in module Goulib.interval), 156
 intersectlen() (in module Goulib.interval), 156
 Interval (class in Goulib.interval), 156
 Intervals (class in Goulib.interval), 159
 InvalidElementError, 185
 inverse() (Goulib.geom.Matrix3 method), 94
 inverse() (Goulib.geom3d.Matrix4 method), 100
 invert() (Goulib.image.Image method), 149
 irange() (Goulib.interval.Intervals method), 162
 irange() (in module Goulib.itertools2), 165
 irange_key() (Goulib.interval.Intervals method), 162
 ireduce() (in module Goulib.itertools2), 167
 iremove() (in module Goulib.itertools2), 169
 is_directed() (Goulib.graph.DiGraph method), 134
 is_directed() (Goulib.graph.GeoGraph method), 118
 is_fibonacci() (in module Goulib.math2), 194
 is_happy() (in module Goulib.math2), 199
 is_heptagonal() (in module Goulib.math2), 199
 is_hexagonal() (in module Goulib.math2), 199
 is_integer() (in module Goulib.math2), 191
 is_lichrel() (in module Goulib.math2), 199
 is_multigraph() (Goulib.graph.DiGraph method), 134
 is_multigraph() (Goulib.graph.GeoGraph method), 118
 is_multiple() (in module Goulib.math2), 195
 is_number() (in module Goulib.math2), 190
 is_octagonal() (in module Goulib.math2), 199
 is_palindromic() (in module Goulib.math2), 197
 is_pandigital() (in module Goulib.math2), 197
 is_pentagonal() (in module Goulib.math2), 199
 is_perfect() (in module Goulib.math2), 200
 is_permutation() (in module Goulib.math2), 197
 is_prime() (in module Goulib.math2), 195
 is_primitive_root() (in module Goulib.math2), 190
 is_pythagorean_triple() (in module Goulib.math2), 194
 is_square() (in module Goulib.math2), 191
 is_triangle() (in module Goulib.math2), 199
 is_triangular() (in module Goulib.math2), 199
 iscallable() (in module Goulib.itertools2), 166
 isclose() (Goulib.colors.Color method), 7
 isclose() (in module Goulib.math2), 191
 isclosed() (Goulib.drawing.Chain method), 49
 isclosed() (Goulib.drawing.Drawing method), 59
 isclosed() (Goulib.drawing.Entity method), 26, 35
 isclosed() (Goulib.drawing.Group method), 42
 isclosed() (Goulib.drawing.Instance method), 45
 isclosed() (Goulib.drawing.Rect method), 52
 isclosed() (Goulib.drawing.Spline method), 38
 isclosed() (Goulib.drawing.Text method), 55
 isclosed() (Goulib.geom.Arc2 method), 85
 isclosed() (Goulib.geom.Circle method), 82
 isclosed() (Goulib.geom.Ellipse method), 88
 isclosed() (Goulib.geom.Point2 method), 73
 isclosed() (Goulib.geom.Segment2 method), 79
 isconstant (Goulib.expr.Expr attribute), 60
 isconstant (Goulib.piecewise.Piecewise attribute), 224
 isconstant (Goulib.polynomial.Polynomial attribute), 227
 isconstant (Goulib.stats.Normal attribute), 238
 isconstant (Goulib.stats.PDF attribute), 236
 ishorizontal() (Goulib.drawing.Chain method), 49
 ishorizontal() (Goulib.drawing.Drawing method), 59
 ishorizontal() (Goulib.drawing.Entity method), 26, 35
 ishorizontal() (Goulib.drawing.Group method), 42
 ishorizontal() (Goulib.drawing.Instance method), 45
 ishorizontal() (Goulib.drawing.Rect method), 52
 ishorizontal() (Goulib.drawing.Spline method), 38
 ishorizontal() (Goulib.drawing.Text method), 55
 ishorizontal() (Goulib.geom.Arc2 method), 85
 ishorizontal() (Goulib.geom.Circle method), 82
 ishorizontal() (Goulib.geom.Ellipse method), 88
 ishorizontal() (Goulib.geom.Point2 method), 73
 ishorizontal() (Goulib.geom.Segment2 method), 79
 iterable() (in module Goulib.itertools2), 165
 islice() (Goulib.interval.Intervals method), 162
 isline() (Goulib.drawing.Chain method), 49
 isline() (Goulib.drawing.Drawing method), 59
 isline() (Goulib.drawing.Entity method), 26, 35
 isline() (Goulib.drawing.Group method), 42
 isline() (Goulib.drawing.Instance method), 45
 isline() (Goulib.drawing.Rect method), 52
 isline() (Goulib.drawing.Spline method), 38
 isline() (Goulib.drawing.Text method), 55

isline() (Goulib.geom.Arc2 method), 85
isline() (Goulib.geom.Circle method), 82
isline() (Goulib.geom.Ellipse method), 88
isline() (Goulib.geom.Point2 method), 73
isline() (Goulib.geom.Segment2 method), 79
isNum (Goulib.expr.Expr attribute), 60
isNum (Goulib.piecewise.Piecewise attribute), 224
isNum (Goulib.polynomial.Polynomial attribute), 227
isNum (Goulib.stats.Normal attribute), 238
isNum (Goulib.stats.PDF attribute), 236
isocalendar() (Goulib.datetime2.date2 method), 18
isocalendar() (Goulib.datetime2.datetime2 method), 16
isoformat() (Goulib.datetime2.date2 method), 18
isoformat() (Goulib.datetime2.datetime2 method), 16
isoformat() (Goulib.datetime2.time2 method), 20
isoformat() (Goulib.datetime2.timedelta2 method), 21
isowekday() (Goulib.datetime2.date2 method), 19
isowekday() (Goulib.datetime2.datetime2 method), 16
isplit() (in module Goulib.itertools2), 169
isqrt() (in module Goulib.math2), 191
isvertical() (Goulib.drawing.Chain method), 49
isvertical() (Goulib.drawing.Drawing method), 59
isvertical() (Goulib.drawing.Entity method), 26, 35
isvertical() (Goulib.drawing.Group method), 42
isvertical() (Goulib.drawing.Instance method), 45
isvertical() (Goulib.drawing.Rect method), 52
isvertical() (Goulib.drawing.Spline method), 38
isvertical() (Goulib.drawing.Text method), 55
isvertical() (Goulib.geom.Arc2 method), 85
isvertical() (Goulib.geom.Circle method), 82
isvertical() (Goulib.geom.Ellipse method), 88
isvertical() (Goulib.geom.Point2 method), 73
isvertical() (Goulib.geom.Segment2 method), 79
isworkday() (Goulib.workdays.WorkCalendar method), 253
isworktime() (Goulib.workdays.WorkCalendar method), 253
itemgetter() (in module Goulib.itertools2), 166
items() (Goulib.colors.Palette method), 10
items() (Goulib.container.Record method), 12
items() (Goulib.graph.index.Index method), 108
items() (Goulib.optim.BinDict method), 217
iter2 (class in Goulib.itertools2), 169
iterate() (in module Goulib.itertools2), 167
itimerout() (in module Goulib.decorators), 24

J

json() (Goulib.table.Table method), 243

K

keep (class in Goulib.itertools2), 172
keys() (Goulib.colors.Palette method), 10
keys() (Goulib.container.Record method), 12
keys() (Goulib.graph.index.Index method), 108

keys() (Goulib.optim.BinDict method), 217
kirkpatrick_cooling() (in module Goulib.optim), 219
kurtosis() (in module Goulib.stats), 231

L

lab (Goulib.colors.Color attribute), 7
lab2ind() (in module Goulib.image), 155
last() (in module Goulib.itertools2), 165
lastExitTime() (Goulib.statemachine.StateMachine method), 230
latex() (Goulib.expr.Expr method), 61
latex() (Goulib.piecewise.Piecewise method), 224
latex() (Goulib.polynomial.Polynomial method), 227
latex() (Goulib.stats.Normal method), 237
latex() (Goulib.stats.PDF method), 236
layer (Goulib.drawing.Chain attribute), 49
layer (Goulib.drawing.Drawing attribute), 59
layer (Goulib.drawing.Group attribute), 27, 39
layer (Goulib.drawing.Rect attribute), 52
lcm() (in module Goulib.math2), 190
length (Goulib.drawing.Chain attribute), 49
length (Goulib.drawing.Drawing attribute), 59
length (Goulib.drawing.Group attribute), 42
length (Goulib.drawing.Instance attribute), 45
length (Goulib.drawing.Rect attribute), 53
length (Goulib.drawing.Spline attribute), 27, 37
length (Goulib.drawing.Text attribute), 30, 54
length (Goulib.geom.Arc2 attribute), 85
length (Goulib.geom.Circle attribute), 80
length (Goulib.geom.Ellipse attribute), 88
length (Goulib.geom.Point2 attribute), 73
length (Goulib.geom.Segment2 attribute), 77
length (Goulib.geom.Vector2 attribute), 69
length (Goulib.geom3d.Segment3 attribute), 97
length() (Goulib.graph.DiGraph method), 135
length() (Goulib.graph.GeoGraph method), 118
levenshtein() (in module Goulib.math2), 194
Line2 (class in Goulib.geom), 74
Line3 (class in Goulib.geom3d), 96
linear() (Goulib.stats.Normal method), 237
linear_regression() (in module Goulib.stats), 239
linspace() (in module Goulib.itertools2), 166
load() (Goulib.drawing.Drawing method), 30, 59
load() (Goulib.image.Image method), 147
load() (Goulib.table.Table method), 243
log() (Goulib.statemachine.EventLog method), 229
log_binomial() (in module Goulib.math2), 200
log_factorial() (in module Goulib.math2), 200
longMessage (Goulib.tests.TestCase attribute), 252
lucas_lehmer() (in module Goulib.math2), 196
luv (Goulib.colors.Color attribute), 7
lychrel_count() (in module Goulib.math2), 199
lychrel_seq() (in module Goulib.math2), 199

M

- mag() (Goulib.geom.Matrix3 method), 94
- mag() (Goulib.geom.Point2 method), 73
- mag() (Goulib.geom.Vector2 method), 69
- mag() (Goulib.geom3d.Quaternion method), 102
- mag() (Goulib.geom3d.Vector3 method), 95
- mag2() (Goulib.geom.Matrix3 method), 94
- mag2() (Goulib.geom.Point2 method), 73
- mag2() (Goulib.geom.Segment2 method), 77
- mag2() (Goulib.geom.Vector2 method), 69
- mag2() (Goulib.geom3d.Quaternion method), 102
- mag2() (Goulib.geom3d.Segment3 method), 97
- mag2() (Goulib.geom3d.Vector3 method), 95
- make_random_population()
 - (Goulib.optim.DifferentialEvolution method), 221
- MarkupError, 180
- Matrix3 (class in Goulib.geom), 89
- Matrix4 (class in Goulib.geom3d), 99
- matrix_power() (in module Goulib.math2), 202
- max (Goulib.datetime2.date2 attribute), 19
- max (Goulib.datetime2.datetime2 attribute), 16
- max (Goulib.datetime2.time2 attribute), 20
- max (Goulib.datetime2.timedelta2 attribute), 22
- max (Goulib.drawing.BBox attribute), 34
- max (Goulib.interval.Box attribute), 165
- maxAbsAcc() (Goulib.motion.Actuator method), 210
- maxAbsSpeed() (Goulib.motion.Actuator method), 210
- maxDiff (Goulib.tests.TestCase attribute), 252
- maxForce() (Goulib.motion.Actuator method), 210
- maximum() (in module Goulib.math2), 193
- maxRpm() (Goulib.motion.Actuator method), 210
- maxTork() (Goulib.motion.Actuator method), 210
- mean (Goulib.stats.Discrete attribute), 234
- mean (Goulib.stats.Normal attribute), 238
- mean (Goulib.stats.PDF attribute), 236
- mean (Goulib.stats.Stats attribute), 231
- mean() (in module Goulib.stats), 230
- mean_var() (in module Goulib.stats), 230
- median() (in module Goulib.stats), 231
- memoize() (in module Goulib.decorators), 24
- metainfo() (Goulib.markup.page method), 177
- microsecond (Goulib.datetime2.datetime2 attribute), 16
- microsecond (Goulib.datetime2.time2 attribute), 20
- microseconds (Goulib.datetime2.timedelta2 attribute), 22
- midpoint() (Goulib.geom.Segment2 method), 77
- min (Goulib.datetime2.date2 attribute), 19
- min (Goulib.datetime2.datetime2 attribute), 16
- min (Goulib.datetime2.time2 attribute), 20
- min (Goulib.datetime2.timedelta2 attribute), 22
- min (Goulib.drawing.BBox attribute), 34
- min (Goulib.interval.Box attribute), 165
- minimum() (in module Goulib.math2), 193
- minus() (Goulib.workdays.WorkCalendar method), 254
- minute (Goulib.datetime2.datetime2 attribute), 16
- minute (Goulib.datetime2.time2 attribute), 20
- minutes() (in module Goulib.datetime2), 23
- mod_binomial() (in module Goulib.math2), 202
- mod_div() (in module Goulib.math2), 201
- mod_fact() (in module Goulib.math2), 201
- mod_inv() (in module Goulib.math2), 201
- mod_matmul() (in module Goulib.math2), 202
- mod_matpow() (in module Goulib.math2), 202
- mod_pow() (in module Goulib.math2), 201
- Mode (class in Goulib.image), 145
- mode() (in module Goulib.stats), 231
- ModeError, 188
- moebius() (in module Goulib.math2), 196
- MON (Goulib.workdays.WorkCalendar attribute), 254
- month (Goulib.datetime2.date2 attribute), 19
- month (Goulib.datetime2.datetime2 attribute), 16
- move() (Goulib.motion.Actuator method), 209
- move_to_end() (Goulib.colors.Palette method), 10
- move_to_end() (Goulib.container.Record method), 12
- mu (Goulib.stats.Discrete attribute), 234
- mu (Goulib.stats.Normal attribute), 238
- mu (Goulib.stats.PDF attribute), 236
- mu (Goulib.stats.Stats attribute), 231
- mul() (in module Goulib.math2), 192
- mult_const() (in module Goulib.polynomial), 228
- mult_one() (in module Goulib.polynomial), 228
- multi (Goulib.graph.DiGraph attribute), 135
- multi (Goulib.graph.GeoGraph attribute), 118
- multiply() (in module Goulib.math2), 191
- multiply() (in module Goulib.polynomial), 228

N

- name (Goulib.colors.Color attribute), 7
- name (Goulib.graph.DiGraph attribute), 135
- name (Goulib.graph.GeoGraph attribute), 118
- native (Goulib.colors.Color attribute), 7
- nbunch_iter() (Goulib.graph.DiGraph method), 135
- nbunch_iter() (Goulib.graph.GeoGraph method), 118
- nchannels (Goulib.image.Image attribute), 147
- nchannels() (in module Goulib.image), 146
- ncols() (Goulib.table.Table method), 243
- ncombinations() (in module Goulib.math2), 200
- ndim() (in module Goulib.itertools2), 167
- nearest() (Goulib.graph.index.Index method), 103, 107
- nearest_color() (in module Goulib.colors), 10
- neighbors() (Goulib.graph.DiGraph method), 135
- neighbors() (Goulib.graph.GeoGraph method), 119
- nelder_mead() (in module Goulib.optim), 215
- networkdays() (Goulib.workdays.WorkCalendar method), 254
- networkdays() (in module Goulib.workdays), 255
- new() (Goulib.geom3d.Matrix4 class method), 100
- new() (Goulib.image.Image static method), 147

`new_edge_key()` (Goulib.graph.DiGraph method), 135
`new_edge_key()` (Goulib.graph.GeoGraph method), 119
`new_identity()` (Goulib.geom.Matrix3 class method), 94
`new_identity()` (Goulib.geom3d.Matrix4 class method), 100
`new_identity()` (Goulib.geom3d.Quaternion class method), 103
`new_interpolate()` (Goulib.geom3d.Quaternion class method), 103
`new_look_at()` (Goulib.geom3d.Matrix4 class method), 100
`new_perspective()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotate()` (Goulib.geom.Matrix3 class method), 94
`new_rotate_axis()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotate_axis()` (Goulib.geom3d.Quaternion class method), 103
`new_rotate_euler()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotate_euler()` (Goulib.geom3d.Quaternion class method), 103
`new_rotate_matrix()` (Goulib.geom3d.Quaternion class method), 103
`new_rotate_triple_axis()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotatex()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotatey()` (Goulib.geom3d.Matrix4 class method), 100
`new_rotatez()` (Goulib.geom3d.Matrix4 class method), 100
`new_scale()` (Goulib.geom.Matrix3 class method), 94
`new_scale()` (Goulib.geom3d.Matrix4 class method), 100
`new_translate()` (Goulib.geom.Matrix3 class method), 94
`new_translate()` (Goulib.geom3d.Matrix4 class method), 100
`next()` (Goulib.itertools2.iter2 method), 170
`next()` (Goulib.itertools2.keep method), 172
`next_permutation()` (in module Goulib.itertools2), 169
`nextworkday()` (Goulib.workdays.WorkCalendar method), 253
`no()` (in module Goulib.itertools2), 168
`node` (Goulib.graph.DiGraph attribute), 135
`node` (Goulib.graph.GeoGraph attribute), 119
`node_dict_factory` (Goulib.graph.DiGraph attribute), 137
`node_dict_factory` (Goulib.graph.GeoGraph attribute), 121
`nodebug()` (in module Goulib.decorators), 24
`nodes` (Goulib.graph.DiGraph attribute), 137
`nodes` (Goulib.graph.GeoGraph attribute), 121
`nodes_with_selfloops()` (Goulib.graph.DiGraph method), 138
`nodes_with_selfloops()` (Goulib.graph.GeoGraph method), 122
`noPrint()` (in module Goulib.statemachine), 229
`norm()` (in module Goulib.math2), 193
`norm_1()` (in module Goulib.math2), 193
`norm_2()` (in module Goulib.math2), 193
`norm_inf()` (in module Goulib.math2), 193
`Normal` (class in Goulib.stats), 236
`normal_pdf()` (in module Goulib.stats), 236
`normalize()` (Goulib.geom.Point2 method), 73
`normalize()` (Goulib.geom.Vector2 method), 69
`normalize()` (Goulib.geom3d.Quaternion method), 102
`normalize()` (Goulib.geom3d.Vector3 method), 95
`normalize()` (Goulib.image.Image method), 150
`normalize()` (in module Goulib.image), 152
`normalized()` (Goulib.geom.Point2 method), 73
`normalized()` (Goulib.geom.Vector2 method), 69
`normalized()` (Goulib.geom3d.Quaternion method), 102
`normalized()` (Goulib.geom3d.Vector3 method), 95
`now()` (Goulib.datetime2.datetime2 method), 16
`npixels` (Goulib.image.Image attribute), 147
`nth()` (in module Goulib.itertools2), 168
`num_from_digits()` (in module Goulib.math2), 197
`number_of_digits()` (in module Goulib.math2), 200
`number_of_divisors()` (in module Goulib.math2), 195
`number_of_edges()` (Goulib.graph.DiGraph method), 138
`number_of_edges()` (Goulib.graph.GeoGraph method), 122
`number_of_nodes()` (Goulib.graph.DiGraph method), 139
`number_of_nodes()` (Goulib.graph.GeoGraph method), 122
`number_of_selfloops()` (Goulib.graph.DiGraph method), 139
`number_of_selfloops()` (Goulib.graph.GeoGraph method), 122
O
`ObjectiveFunction` (class in Goulib.optim), 214
`occurrences()` (in module Goulib.itertools2), 168
`octagonal()` (in module Goulib.math2), 199
`offset()` (Goulib.geom.Matrix3 method), 94
`omega()` (in module Goulib.math2), 195
`open()` (Goulib.image.Image static method), 147
`open()` (Goulib.markup.element method), 174
`OpeningError`, 183
`optimize()` (Goulib.image.Image method), 147
`optimize()` (Goulib.optim.DifferentialEvolution method), 221
`order()` (Goulib.graph.DiGraph method), 139
`order()` (Goulib.graph.GeoGraph method), 122
`orientation()` (Goulib.geom.Matrix3 method), 94
`out_degree` (Goulib.graph.DiGraph attribute), 139
`out_edges` (Goulib.graph.DiGraph attribute), 139
`overlap()` (Goulib.interval.Interval method), 157

P

- P() (Goulib.motion.Actuator method), 210
- P() (in module Goulib.optim), 219
- p1 (Goulib.drawing.Rect attribute), 29, 50
- p1 (Goulib.geom.Segment2 attribute), 77
- p1 (Goulib.geom3d.Line3 attribute), 97
- p2 (Goulib.drawing.Rect attribute), 29, 50
- p2 (Goulib.geom.Segment2 attribute), 77
- p2 (Goulib.geom3d.Line3 attribute), 97
- page (class in Goulib.markup), 175
- pairwise() (Goulib.container.Sequence method), 14
- pairwise() (in module Goulib.itertools2), 167
- Palette (class in Goulib.colors), 8
- palette() (in module Goulib.image), 155
- parse_string() (in module Goulib.polynomial), 228
- parseDoc() (Goulib.statemachine.StateMachine method), 230
- partition() (in module Goulib.math2), 200
- pascal_gen() (in module Goulib.math2), 194
- paste() (Goulib.image.Image method), 148
- patches() (Goulib.colors.Palette method), 8
- patches() (Goulib.drawing.Chain method), 49
- patches() (Goulib.drawing.Drawing method), 59
- patches() (Goulib.drawing.Entity method), 26, 35
- patches() (Goulib.drawing.Group method), 42
- patches() (Goulib.drawing.Instance method), 45
- patches() (Goulib.drawing.Rect method), 53
- patches() (Goulib.drawing.Spline method), 38
- patches() (Goulib.drawing.Text method), 30, 54
- patches() (Goulib.geom.Arc2 method), 85
- patches() (Goulib.geom.Circle method), 82
- patches() (Goulib.geom.Ellipse method), 88
- patches() (Goulib.geom.Point2 method), 73
- patches() (Goulib.geom.Segment2 method), 79
- PDF (class in Goulib.stats), 234
- pearson() (Goulib.stats.Normal method), 237
- pentagonal() (in module Goulib.math2), 199
- pep8() (in module Goulib.tests), 252
- perceptual_hash() (Goulib.image.Image method), 149
- peval() (in module Goulib.polynomial), 228
- pi_digits_gen() (in module Goulib.math2), 202
- Piecewise (class in Goulib.piecewise), 222
- pil (Goulib.colors.Palette attribute), 8
- pil (Goulib.image.Image attribute), 147
- pisano_cycle() (in module Goulib.math2), 194
- pisano_period() (in module Goulib.math2), 194
- Plane (class in Goulib.geom3d), 98
- plist() (in module Goulib.polynomial), 228
- Plot (class in Goulib.plot), 224
- plot() (Goulib.drawing.Chain method), 49
- plot() (Goulib.drawing.Drawing method), 59
- plot() (Goulib.drawing.Entity method), 36
- plot() (Goulib.drawing.Group method), 42
- plot() (Goulib.drawing.Instance method), 45
- plot() (Goulib.drawing.Rect method), 53
- plot() (Goulib.drawing.Spline method), 39
- plot() (Goulib.drawing.Text method), 55
- plot() (Goulib.expr.Expr method), 62
- plot() (Goulib.geom.Arc2 method), 85
- plot() (Goulib.geom.Circle method), 82
- plot() (Goulib.geom.Ellipse method), 88
- plot() (Goulib.geom.Point2 method), 73
- plot() (Goulib.geom.Segment2 method), 79
- plot() (Goulib.graph.DiGraph method), 140
- plot() (Goulib.graph.GeoGraph method), 123
- plot() (Goulib.image.Image method), 151
- plot() (Goulib.motion.PVA method), 203
- plot() (Goulib.motion.Segment method), 205
- plot() (Goulib.motion.SegmentPoly method), 208
- plot() (Goulib.motion.Segments method), 206
- plot() (Goulib.motion.TimeDiagram method), 212
- plot() (Goulib.piecewise.Piecewise method), 224
- plot() (Goulib.plot.Plot method), 224
- plot() (Goulib.polynomial.Polynomial method), 227
- plot() (Goulib.stats.Normal method), 238
- plot() (Goulib.stats.PDF method), 236
- plot() (in module Goulib.plot), 225
- plouffe() (in module Goulib.expr), 60
- plus() (Goulib.workdays.WorkCalendar method), 253
- png() (Goulib.drawing.Chain method), 49
- png() (Goulib.drawing.Drawing method), 59
- png() (Goulib.drawing.Entity method), 36
- png() (Goulib.drawing.Group method), 42
- png() (Goulib.drawing.Instance method), 45
- png() (Goulib.drawing.Rect method), 53
- png() (Goulib.drawing.Spline method), 39
- png() (Goulib.drawing.Text method), 55
- png() (Goulib.expr.Expr method), 62
- png() (Goulib.geom.Arc2 method), 86
- png() (Goulib.geom.Circle method), 83
- png() (Goulib.geom.Ellipse method), 88
- png() (Goulib.geom.Point2 method), 73
- png() (Goulib.geom.Segment2 method), 79
- png() (Goulib.graph.DiGraph method), 140
- png() (Goulib.graph.GeoGraph method), 123
- png() (Goulib.image.Image method), 151
- png() (Goulib.motion.PVA method), 203
- png() (Goulib.motion.Segment method), 205
- png() (Goulib.motion.SegmentPoly method), 208
- png() (Goulib.motion.Segments method), 206
- png() (Goulib.motion.TimeDiagram method), 212
- png() (Goulib.piecewise.Piecewise method), 224
- png() (Goulib.plot.Plot method), 224
- png() (Goulib.polynomial.Polynomial method), 227
- png() (Goulib.stats.Normal method), 238
- png() (Goulib.stats.PDF method), 236
- png() (in module Goulib.plot), 225
- point() (Goulib.drawing.Chain method), 49

point() (Goulib.drawing.Drawing method), 59
point() (Goulib.drawing.Group method), 42
point() (Goulib.drawing.Instance method), 45
point() (Goulib.drawing.Rect method), 53
point() (Goulib.drawing.Spline method), 39
point() (Goulib.geom.Arc2 method), 83
point() (Goulib.geom.Circle method), 80
point() (Goulib.geom.Ellipse method), 88
point() (Goulib.geom.Geometry method), 64
point() (Goulib.geom.Line2 method), 75
point() (Goulib.geom.Point2 method), 74
point() (Goulib.geom.Ray2 method), 77
point() (Goulib.geom.Segment2 method), 79
point() (Goulib.geom3d.Line3 method), 97
point() (Goulib.geom3d.Sphere method), 98
Point2 (class in Goulib.geom), 70
Point3 (class in Goulib.geom3d), 95
points() (Goulib.piecewise.Piecewise method), 224
points_on_sphere() (in module Goulib.graph), 105, 145
Polar() (in module Goulib.geom), 74
polygonal() (in module Goulib.math2), 199
Polynomial (class in Goulib.polynomial), 225
pop() (Goulib.colors.Palette method), 10
pop() (Goulib.container.Record method), 12
pop() (Goulib.drawing.BBox method), 34
pop() (Goulib.drawing.Chain method), 49
pop() (Goulib.drawing.Drawing method), 59
pop() (Goulib.drawing.Group method), 42
pop() (Goulib.drawing.Rect method), 53
pop() (Goulib.graph.index.Index method), 108
pop() (Goulib.interval.Box method), 165
pop() (Goulib.interval.Interval method), 159
pop() (Goulib.interval.Intervals method), 162
pop() (Goulib.optim.BinDict method), 217
pop() (Goulib.optim.BinList method), 217
pop() (Goulib.table.Table method), 245
popitem() (Goulib.colors.Palette method), 10
popitem() (Goulib.container.Record method), 12
popitem() (Goulib.graph.index.Index method), 108
popitem() (Goulib.optim.BinDict method), 217
pos() (Goulib.graph.DiGraph method), 140
pos() (Goulib.graph.GeoGraph method), 123
power() (in module Goulib.polynomial), 228
power_tower() (in module Goulib.math2), 197
powertrain() (in module Goulib.math2), 197
pprint() (in module Goulib.tests), 246
pprint_gen() (in module Goulib.tests), 246
prec() (Goulib.expr.TextVisitor method), 62
pred (Goulib.graph.DiGraph attribute), 140
predecessors() (Goulib.graph.DiGraph method), 140
prevworkday() (Goulib.workdays.WorkCalendar method), 253
prime_divisors() (in module Goulib.math2), 195
prime_factors() (in module Goulib.math2), 195

prime_ktuple() (in module Goulib.math2), 196
primes() (in module Goulib.math2), 195
primes_gen() (in module Goulib.math2), 195
primitive_root_gen() (in module Goulib.math2), 191
primitive_roots() (in module Goulib.math2), 191
primitive_triples() (in module Goulib.math2), 194
project() (Goulib.geom.Point2 method), 74
project() (Goulib.geom.Vector2 method), 69
project() (Goulib.geom3d.Vector3 method), 95
proper_divisors() (in module Goulib.math2), 195
proper_subset() (Goulib.interval.Interval method), 157
proportional() (in module Goulib.math2), 201
pure_pil_alpha_to_color_v1() (in module Goulib.image), 151
pure_pil_alpha_to_color_v2() (in module Goulib.image), 152
putpixel() (Goulib.image.Image method), 147
PVA (class in Goulib.motion), 202
pyramidal() (in module Goulib.math2), 198

Q

quad() (in module Goulib.math2), 191
quantify() (in module Goulib.itertools2), 168
quantize() (Goulib.image.Image method), 149
quantize() (in module Goulib.image), 152
Quaternion (class in Goulib.geom3d), 100

R

ramp() (in module Goulib.motion), 212
rand_seq() (in module Goulib.itertools2), 168
random_prime() (in module Goulib.math2), 195
randomize() (in module Goulib.image), 152
range() (Goulib.workdays.WorkCalendar method), 253
rational_cycle() (in module Goulib.math2), 198
rational_form() (in module Goulib.math2), 198
rational_str() (in module Goulib.math2), 198
Ray2 (class in Goulib.geom), 76
Ray3 (class in Goulib.geom3d), 97
read() (Goulib.table.Cell static method), 240
read_csv() (Goulib.table.Table method), 243
read_dxf() (Goulib.drawing.Drawing method), 30, 59
read_element() (Goulib.table.Table method), 243
read_html() (Goulib.table.Table method), 243
read_json() (Goulib.table.Table method), 243
read_json() (in module Goulib.graph), 105, 145
read_pdf() (Goulib.drawing.Drawing method), 30, 59
read_pdf() (in module Goulib.image), 152
read_svg() (Goulib.drawing.Drawing method), 30, 59
read_xls() (Goulib.table.Table method), 243
Record (class in Goulib.container), 11
Rect (class in Goulib.drawing), 29, 49
rectangular_repartition() (in module Goulib.math2), 201
recurrence() (in module Goulib.itertools2), 166
recurrence() (in module Goulib.math2), 194

- reflect() (Goulib.geom.Point2 method), 74
- reflect() (Goulib.geom.Vector2 method), 69
- reflect() (Goulib.geom3d.Vector3 method), 95
- remove() (Goulib.drawing.BBox method), 34
- remove() (Goulib.drawing.Chain method), 49
- remove() (Goulib.drawing.Drawing method), 59
- remove() (Goulib.drawing.Group method), 42
- remove() (Goulib.drawing.Rect method), 53
- remove() (Goulib.interval.Box method), 165
- remove() (Goulib.interval.Interval method), 159
- remove() (Goulib.interval.Intervals method), 162
- remove() (Goulib.optim.BinList method), 217
- remove() (Goulib.stats.Discrete method), 234
- remove() (Goulib.stats.Normal method), 239
- remove() (Goulib.stats.PDF method), 236
- remove() (Goulib.stats.Stats method), 231
- remove() (Goulib.table.Table method), 245
- remove_edge() (Goulib.graph.DiGraph method), 140
- remove_edge() (Goulib.graph.GeoGraph method), 123
- remove_edges_from() (Goulib.graph.DiGraph method), 141
- remove_edges_from() (Goulib.graph.GeoGraph method), 123
- remove_lines_where() (Goulib.table.Table method), 246
- remove_node() (Goulib.graph.DiGraph method), 141
- remove_node() (Goulib.graph.GeoGraph method), 123
- remove_nodes_from() (Goulib.graph.DiGraph method), 141
- remove_nodes_from() (Goulib.graph.GeoGraph method), 124
- removef() (in module Goulib.itertools2), 169
- render() (Goulib.drawing.Chain method), 49
- render() (Goulib.drawing.Drawing method), 59
- render() (Goulib.drawing.Entity method), 27, 35
- render() (Goulib.drawing.Group method), 42
- render() (Goulib.drawing.Instance method), 45
- render() (Goulib.drawing.Rect method), 53
- render() (Goulib.drawing.Spline method), 39
- render() (Goulib.drawing.Text method), 55
- render() (Goulib.expr.Expr method), 62
- render() (Goulib.geom.Arc2 method), 86
- render() (Goulib.geom.Circle method), 83
- render() (Goulib.geom.Ellipse method), 88
- render() (Goulib.geom.Point2 method), 74
- render() (Goulib.geom.Segment2 method), 79
- render() (Goulib.graph.DiGraph method), 141
- render() (Goulib.graph.GeoGraph method), 124
- render() (Goulib.image.Image method), 147
- render() (Goulib.markup.element method), 174
- render() (Goulib.motion.PVA method), 203
- render() (Goulib.motion.Segment method), 205
- render() (Goulib.motion.SegmentPoly method), 208
- render() (Goulib.motion.Segments method), 207
- render() (Goulib.motion.TimeDiagram method), 212
- render() (Goulib.piecewise.Piecewise method), 224
- render() (Goulib.plot.Plot method), 224
- render() (Goulib.polynomial.Polynomial method), 228
- render() (Goulib.stats.Normal method), 239
- render() (Goulib.stats.PDF method), 236
- render() (in module Goulib.plot), 225
- replace() (Goulib.datetime2.date2 method), 19
- replace() (Goulib.datetime2.datetime2 method), 16
- replace() (Goulib.datetime2.time2 method), 20
- replace() (Goulib.image.Image method), 147
- repunit() (in module Goulib.math2), 198
- repunit_gen() (in module Goulib.math2), 198
- reset() (Goulib.statemachine.StateMachine method), 229
- reshape() (in module Goulib.itertools2), 167
- resize() (Goulib.image.Image method), 148
- resolution (Goulib.datetime2.date2 attribute), 19
- resolution (Goulib.datetime2.datetime2 attribute), 16
- resolution (Goulib.datetime2.time2 attribute), 20
- resolution (Goulib.datetime2.timedelta2 attribute), 22
- reverse() (Goulib.drawing.BBox method), 34
- reverse() (Goulib.drawing.Chain method), 49
- reverse() (Goulib.drawing.Drawing method), 59
- reverse() (Goulib.drawing.Group method), 42
- reverse() (Goulib.drawing.Rect method), 53
- reverse() (Goulib.graph.DiGraph method), 142
- reverse() (Goulib.interval.Box method), 165
- reverse() (Goulib.interval.Interval method), 159
- reverse() (Goulib.interval.Intervals method), 163
- reverse() (Goulib.optim.BinList method), 219
- reverse() (Goulib.table.Table method), 245
- reverse() (in module Goulib.math2), 197
- reversed_sections() (in module Goulib.optim), 219
- rgb (Goulib.colors.Color attribute), 7
- rgb2cmk() (in module Goulib.colors), 6
- rgb2cmk() (in module Goulib.image), 155
- rgb2hex() (in module Goulib.colors), 6
- rgb2rgba() (in module Goulib.image), 155
- rint() (in module Goulib.math2), 191
- rotate() (Goulib.geom.Matrix3 method), 94
- rotate() (Goulib.image.Image method), 148
- rotate_around() (Goulib.geom3d.Vector3 method), 95
- rotate_axis() (Goulib.geom3d.Matrix4 method), 99
- rotate_axis() (Goulib.geom3d.Quaternion method), 102
- rotate_euler() (Goulib.geom3d.Matrix4 method), 99
- rotate_euler() (Goulib.geom3d.Quaternion method), 102
- rotate_matrix() (Goulib.geom3d.Quaternion method), 102
- rotate_triple_axis() (Goulib.geom3d.Matrix4 method), 100
- rotatex() (Goulib.geom3d.Matrix4 method), 99
- rotatey() (Goulib.geom3d.Matrix4 method), 99
- rotatez() (Goulib.geom3d.Matrix4 method), 99
- Row (class in Goulib.table), 241
- rowasdict() (Goulib.table.Table method), 245

run() (Goulib.statemachine.StateMachine method), 230
run() (Goulib.tests.TestCase method), 252
runmodule() (in module Goulib.tests), 252
runtests() (in module Goulib.tests), 252
russell (class in Goulib.markup), 179

S

SAT (Goulib.workdays.WorkCalendar attribute), 254
sat() (in module Goulib.math2), 193
save() (Goulib.container.Sequence method), 13
save() (Goulib.drawing.Chain method), 49
save() (Goulib.drawing.Drawing method), 30, 60
save() (Goulib.drawing.Entity method), 36
save() (Goulib.drawing.Group method), 43
save() (Goulib.drawing.Instance method), 45
save() (Goulib.drawing.Rect method), 53
save() (Goulib.drawing.Spline method), 39
save() (Goulib.drawing.Text method), 56
save() (Goulib.expr.Expr method), 62
save() (Goulib.geom.Arc2 method), 86
save() (Goulib.geom.Circle method), 83
save() (Goulib.geom.Ellipse method), 88
save() (Goulib.geom.Point2 method), 74
save() (Goulib.geom.Segment2 method), 79
save() (Goulib.graph.DiGraph method), 142
save() (Goulib.graph.GeoGraph method), 124
save() (Goulib.image.Image method), 147
save() (Goulib.motion.PVA method), 203
save() (Goulib.motion.Segment method), 205
save() (Goulib.motion.SegmentPoly method), 208
save() (Goulib.motion.Segments method), 207
save() (Goulib.motion.TimeDiagram method), 212
save() (Goulib.piecewise.Piecewise method), 224
save() (Goulib.plot.Plot method), 224
save() (Goulib.polynomial.Polynomial method), 228
save() (Goulib.stats.Normal method), 239
save() (Goulib.stats.PDF method), 236
save() (Goulib.table.Table method), 243
save() (in module Goulib.plot), 225
saveAsCsv() (Goulib.motion.TimeDiagram method), 211
scale() (Goulib.geom.Matrix3 method), 93
scale() (Goulib.geom3d.Matrix4 method), 99
scale() (Goulib.image.Image method), 150
score_population() (Goulib.optim.DifferentialEvolution method), 221
scripts() (Goulib.markup.page method), 177
second (Goulib.datetime2.datetime2 attribute), 16
second (Goulib.datetime2.time2 attribute), 20
seconds (Goulib.datetime2.timedelta2 attribute), 22
Segment (class in Goulib.motion), 203
Segment2 (class in Goulib.geom), 77
Segment2ndDegree() (in module Goulib.motion), 213
Segment3 (class in Goulib.geom3d), 97
Segment4thDegree() (in module Goulib.motion), 213

SegmentPoly (class in Goulib.motion), 207
Segments (class in Goulib.motion), 205
SegmentsTrapezoidalSpeed() (in module Goulib.motion), 213
selfloop_edges() (Goulib.graph.DiGraph method), 142
selfloop_edges() (Goulib.graph.GeoGraph method), 124
separation() (Goulib.interval.Interval method), 157
Sequence (class in Goulib.container), 12
set() (Goulib.statemachine.TimeMarker method), 229
set() (Goulib.table.Table method), 243
set_dimension() (Goulib.graph.index.Property method), 103, 105
setattr() (Goulib.drawing.Chain method), 49
setattr() (Goulib.drawing.Drawing method), 59
setattr() (Goulib.drawing.Entity method), 26, 34
setattr() (Goulib.drawing.Group method), 43
setattr() (Goulib.drawing.Instance method), 45
setattr() (Goulib.drawing.Rect method), 53
setattr() (Goulib.drawing.Spline method), 39
setattr() (Goulib.drawing.Text method), 56
setattr() (Goulib.geom.Arc2 method), 86
setattr() (Goulib.geom.Circle method), 83
setattr() (Goulib.geom.Ellipse method), 88
setattr() (Goulib.geom.Point2 method), 74
setattr() (Goulib.geom.Segment2 method), 79
setcol() (Goulib.table.Table method), 244
setdefault() (Goulib.colors.Palette method), 10
setdefault() (Goulib.container.Record method), 12
setdefault() (Goulib.graph.index.Index method), 108
setdefault() (Goulib.optim.BinDict method), 217
setlog() (in module Goulib.tests), 252
setOutput() (Goulib.statemachine.Simulation method), 229
setpalette() (Goulib.image.Image method), 147
sets_dist() (in module Goulib.math2), 194
sets_levenshtein() (in module Goulib.math2), 194
setUp() (Goulib.tests.TestCase method), 252
setUpClass() (Goulib.tests.TestCase method), 252
setworktime() (Goulib.workdays.WorkCalendar method), 253
sexy_prime_quadruplets() (in module Goulib.math2), 196
sexy_prime_triplets() (in module Goulib.math2), 196
sexy_primes() (in module Goulib.math2), 196
shape (Goulib.image.Image attribute), 147
shape() (in module Goulib.itertools2), 167
shift() (Goulib.image.Image method), 150
shortDescription() (Goulib.tests.TestCase method), 252
shortest_path() (Goulib.graph.DiGraph method), 142
shortest_path() (Goulib.graph.GeoGraph method), 124
shuffle() (in module Goulib.itertools2), 168
sieve() (in module Goulib.math2), 195
sigma (Goulib.stats.Discrete attribute), 234
sigma (Goulib.stats.Normal attribute), 237
sigma (Goulib.stats.PDF attribute), 236

- sigma (Goulib.stats.Stats attribute), 231
- sign() (in module Goulib.math2), 190
- Simulation (class in Goulib.statemachine), 229
- sin_over_x() (in module Goulib.math2), 201
- singleton() (Goulib.interval.Interval method), 157
- size (Goulib.image.Image attribute), 147
- size (Goulib.interval.Box attribute), 163
- size (Goulib.interval.Interval attribute), 157
- size() (Goulib.drawing.BBox method), 25, 32
- size() (Goulib.graph.DiGraph method), 142
- size() (Goulib.graph.GeoGraph method), 124
- size() (Goulib.optim.BinDict method), 217
- size() (Goulib.optim.BinList method), 219
- skipTest() (Goulib.tests.TestCase method), 252
- slerp() (in module Goulib.math2), 201
- sort() (Goulib.container.Sequence method), 14
- sort() (Goulib.drawing.BBox method), 34
- sort() (Goulib.drawing.Chain method), 49
- sort() (Goulib.drawing.Drawing method), 59
- sort() (Goulib.drawing.Group method), 43
- sort() (Goulib.drawing.Rect method), 53
- sort() (Goulib.interval.Box method), 165
- sort() (Goulib.interval.Interval method), 159
- sort() (Goulib.optim.BinList method), 219
- sort() (Goulib.table.Table method), 245
- sort_indexes() (in module Goulib.itertools2), 169
- sorted() (Goulib.colors.Palette method), 9
- sorted_iterable() (in module Goulib.itertools2), 172
- SortingError, 171
- Sphere (class in Goulib.geom3d), 97
- Spherical() (in module Goulib.geom3d), 97
- Spline (class in Goulib.drawing), 27, 36
- split() (Goulib.image.Image method), 147
- split() (in module Goulib.itertools2), 169
- sqrt() (in module Goulib.math2), 191
- square() (in module Goulib.math2), 199
- start (Goulib.drawing.BBox attribute), 34
- start (Goulib.drawing.Chain attribute), 28, 46
- start (Goulib.drawing.Drawing attribute), 59
- start (Goulib.drawing.Entity attribute), 26, 34
- start (Goulib.drawing.Group attribute), 43
- start (Goulib.drawing.Instance attribute), 45
- start (Goulib.drawing.Rect attribute), 53
- start (Goulib.drawing.Spline attribute), 27, 37
- start (Goulib.drawing.Text attribute), 56
- start (Goulib.geom.Arc2 attribute), 86
- start (Goulib.geom.Circle attribute), 83
- start (Goulib.geom.Ellipse attribute), 88
- start (Goulib.geom.Point2 attribute), 74
- start (Goulib.geom.Segment2 attribute), 80
- start (Goulib.interval.Box attribute), 163
- start (Goulib.interval.Interval attribute), 156
- start (Goulib.workdays.WorkCalendar attribute), 253
- start() (Goulib.motion.Segment method), 203
- start() (Goulib.motion.SegmentPoly method), 208
- start() (Goulib.motion.Segments method), 205
- startAcc() (Goulib.motion.Segment method), 203
- startAcc() (Goulib.motion.SegmentPoly method), 208
- startAcc() (Goulib.motion.Segments method), 207
- startJerk() (Goulib.motion.Segment method), 204
- startJerk() (Goulib.motion.SegmentPoly method), 208
- startJerk() (Goulib.motion.Segments method), 207
- startPos() (Goulib.motion.Segment method), 203
- startPos() (Goulib.motion.SegmentPoly method), 208
- startPos() (Goulib.motion.Segments method), 207
- startSpeed() (Goulib.motion.Segment method), 203
- startSpeed() (Goulib.motion.SegmentPoly method), 208
- startSpeed() (Goulib.motion.Segments method), 207
- startTime() (Goulib.motion.Segment method), 204
- startTime() (Goulib.motion.SegmentPoly method), 208
- startTime() (Goulib.motion.Segments method), 207
- state() (Goulib.statemachine.StateDiagram method), 229
- StateChangeLog (class in Goulib.statemachine), 229
- StateDiagram (class in Goulib.statemachine), 228
- StateMachine (class in Goulib.statemachine), 229
- Stats (class in Goulib.stats), 231
- stats() (Goulib.graph.DiGraph method), 142
- stats() (Goulib.graph.GeoGraph method), 124
- stats() (in module Goulib.stats), 231
- stddev (Goulib.stats.Discrete attribute), 234
- stddev (Goulib.stats.Normal attribute), 239
- stddev (Goulib.stats.PDF attribute), 236
- stddev (Goulib.stats.Stats attribute), 231
- stddev() (in module Goulib.stats), 230
- str() (Goulib.colors.Color method), 7
- str_base() (in module Goulib.math2), 197
- strftime() (Goulib.datetime2.date2 method), 19
- strftime() (Goulib.datetime2.datetime2 method), 16
- strftime() (Goulib.datetime2.time2 method), 20
- strtimedelta() (in module Goulib.datetime2), 23
- strptime() (Goulib.datetime2.datetime2 method), 16
- style_dict2str() (in module Goulib.markup), 174
- style_str2dict() (in module Goulib.markup), 174
- sub() (Goulib.image.Image method), 150
- sub() (in module Goulib.polynomial), 228
- subdict() (in module Goulib.itertools2), 171
- subgraph() (Goulib.graph.DiGraph method), 142
- subgraph() (Goulib.graph.GeoGraph method), 124
- subset() (Goulib.interval.Interval method), 157
- subTest() (Goulib.tests.TestCase method), 252
- succ (Goulib.graph.DiGraph attribute), 143
- successors() (Goulib.graph.DiGraph method), 143
- sum (Goulib.stats.Discrete attribute), 234
- sum (Goulib.stats.Normal attribute), 239
- sum (Goulib.stats.PDF attribute), 236
- sum (Goulib.stats.Stats attribute), 231
- sum1 (Goulib.stats.Discrete attribute), 234
- sum1 (Goulib.stats.Normal attribute), 239

sum1 (Goulib.stats.PDF attribute), 236
sum1 (Goulib.stats.Stats attribute), 231
sum2 (Goulib.stats.Discrete attribute), 234
sum2 (Goulib.stats.Normal attribute), 239
sum2 (Goulib.stats.PDF attribute), 236
sum2 (Goulib.stats.Stats attribute), 231
sum_of_cubes() (in module Goulib.math2), 198
sum_of_squares() (in module Goulib.math2), 198
SUN (Goulib.workdays.WorkCalendar attribute), 254
svg() (Goulib.drawing.Chain method), 49
svg() (Goulib.drawing.Drawing method), 59
svg() (Goulib.drawing.Entity method), 36
svg() (Goulib.drawing.Group method), 43
svg() (Goulib.drawing.Instance method), 45
svg() (Goulib.drawing.Rect method), 53
svg() (Goulib.drawing.Spline method), 39
svg() (Goulib.drawing.Text method), 56
svg() (Goulib.expr.Expr method), 62
svg() (Goulib.geom.Arc2 method), 86
svg() (Goulib.geom.Circle method), 83
svg() (Goulib.geom.Ellipse method), 88
svg() (Goulib.geom.Point2 method), 74
svg() (Goulib.geom.Segment2 method), 80
svg() (Goulib.graph.DiGraph method), 143
svg() (Goulib.graph.GeoGraph method), 125
svg() (Goulib.image.Image method), 151
svg() (Goulib.motion.PVA method), 203
svg() (Goulib.motion.Segment method), 205
svg() (Goulib.motion.SegmentPoly method), 208
svg() (Goulib.motion.Segments method), 207
svg() (Goulib.motion.TimeDiagram method), 212
svg() (Goulib.pieewise.Piecewise method), 224
svg() (Goulib.plot.Plot method), 224
svg() (Goulib.polynomial.Polynomial method), 228
svg() (Goulib.stats.Normal method), 239
svg() (Goulib.stats.PDF method), 236
svg() (in module Goulib.plot), 225
swap() (Goulib.drawing.Chain method), 49
swap() (Goulib.drawing.Drawing method), 59
swap() (Goulib.drawing.Group method), 27, 39
swap() (Goulib.drawing.Rect method), 53
swap() (Goulib.drawing.Spline method), 27, 37
swap() (Goulib.geom.Arc2 method), 84
swap() (Goulib.geom.Circle method), 81
swap() (Goulib.geom.Ellipse method), 88
swap() (Goulib.geom.Segment2 method), 77
swap() (Goulib.geom3d.Segment3 method), 97
swap() (in module Goulib.itertools2), 166
swapped_cities() (in module Goulib.optim), 219

T

Table (class in Goulib.table), 242
tag() (in module Goulib.markup), 174
tails() (in module Goulib.itertools2), 167

take() (in module Goulib.itertools2), 165
takeevery() (in module Goulib.itertools2), 165
takens() (in module Goulib.itertools2), 168
tangent() (Goulib.drawing.Chain method), 49
tangent() (Goulib.drawing.Drawing method), 59
tangent() (Goulib.drawing.Group method), 43
tangent() (Goulib.drawing.Instance method), 45
tangent() (Goulib.drawing.Rect method), 53
tangent() (Goulib.drawing.Spline method), 39
tangent() (Goulib.geom.Arc2 method), 84
tangent() (Goulib.geom.Circle method), 81
tangent() (Goulib.geom.Ellipse method), 88
tangent() (Goulib.geom.Geometry method), 64
tangent() (Goulib.geom.Line2 method), 75
tangent() (Goulib.geom.Point2 method), 74
tangent() (Goulib.geom.Ray2 method), 77
tangent() (Goulib.geom.Segment2 method), 80
tdround() (in module Goulib.datetime2), 23
tearDown() (Goulib.tests.TestCase method), 252
tearDownClass() (Goulib.tests.TestCase method), 252
tee() (in module Goulib.itertools2), 166
TestCase (class in Goulib.tests), 246
tetrahedral() (in module Goulib.math2), 198
Text (class in Goulib.drawing), 29, 53
TextVisitor (class in Goulib.expr), 62
threshold() (Goulib.image.Image method), 149
THU (Goulib.workdays.WorkCalendar attribute), 254
time() (Goulib.datetime2.datetime2 method), 16
time2 (class in Goulib.datetime2), 19
time_add() (in module Goulib.datetime2), 24
time_intersect() (in module Goulib.datetime2), 24
time_sub() (in module Goulib.datetime2), 24
timedelta2 (class in Goulib.datetime2), 20
timedelta_div() (in module Goulib.datetime2), 24
timedelta_mul() (in module Goulib.datetime2), 24
timedelta_sum() (in module Goulib.datetime2), 24
timedeltaaf() (in module Goulib.datetime2), 23
TimeDiagram (class in Goulib.motion), 211
timef() (in module Goulib.datetime2), 23
TimeMarker (class in Goulib.statemachine), 229
timeout() (in module Goulib.decorators), 24
timestamp() (Goulib.datetime2.datetime2 method), 17
timetuple() (Goulib.datetime2.date2 method), 19
timetuple() (Goulib.datetime2.datetime2 method), 17
timetz() (Goulib.datetime2.datetime2 method), 17
timeWhenPosBiggerThan() (Goulib.motion.Segment method), 204
timeWhenPosBiggerThan() (Goulib.motion.SegmentPoly method), 208
timeWhenPosBiggerThan() (Goulib.motion.Segments method), 207
to_date() (Goulib.table.Table method), 246
to_datetime() (Goulib.table.Table method), 246
to_directed() (Goulib.graph.DiGraph method), 143

to_directed() (Goulib.graph.GeoGraph method), 125
 to_drawing() (in module Goulib.graph), 104, 144
 to_dxf() (Goulib.drawing.Chain method), 29, 46
 to_dxf() (Goulib.drawing.Drawing method), 59
 to_dxf() (Goulib.drawing.Entity method), 26, 35
 to_dxf() (Goulib.drawing.Group method), 43
 to_dxf() (Goulib.drawing.Instance method), 45
 to_dxf() (Goulib.drawing.Rect method), 53
 to_dxf() (Goulib.drawing.Spline method), 39
 to_dxf() (Goulib.drawing.Text method), 30, 54
 to_dxf() (Goulib.geom.Arc2 method), 86
 to_dxf() (Goulib.geom.Circle method), 83
 to_dxf() (Goulib.geom.Ellipse method), 89
 to_dxf() (Goulib.geom.Point2 method), 74
 to_dxf() (Goulib.geom.Segment2 method), 80
 to_json() (in module Goulib.graph), 105, 145
 to_networkx_graph() (in module Goulib.graph), 103, 110
 to_time() (Goulib.table.Table method), 246
 to_timedelta() (Goulib.table.Table method), 246
 to_undirected() (Goulib.graph.DiGraph method), 143
 to_undirected() (Goulib.graph.GeoGraph method), 125
 today() (Goulib.datetime2.date2 method), 19
 today() (Goulib.datetime2.datetime2 method), 17
 tol (Goulib.graph.DiGraph attribute), 144
 tol (Goulib.graph.GeoGraph attribute), 126
 TooLateLog (class in Goulib.statemachine), 229
 toordinal() (Goulib.datetime2.date2 method), 19
 toordinal() (Goulib.datetime2.datetime2 method), 17
 tostring() (in module Goulib.polynomial), 228
 total() (Goulib.table.Table method), 246
 total_seconds() (Goulib.datetime2.timedelta2 method), 22
 totient() (in module Goulib.math2), 196
 tour_length() (in module Goulib.optim), 219
 trailing_zeros() (in module Goulib.math2), 197
 trans() (Goulib.drawing.BBox method), 26, 32
 Trans() (in module Goulib.drawing), 25, 31
 transform() (Goulib.geom3d.Matrix4 method), 99
 translate() (Goulib.geom.Matrix3 method), 94
 translate() (Goulib.geom3d.Matrix4 method), 99
 transpose() (Goulib.geom.Matrix3 method), 94
 transpose() (Goulib.geom3d.Matrix4 method), 100
 transpose() (Goulib.table.Table method), 243
 transpose() (in module Goulib.math2), 193
 transposed() (Goulib.geom.Matrix3 method), 94
 transposed() (Goulib.geom3d.Matrix4 method), 100
 trapeze() (in module Goulib.motion), 212
 triangle() (in module Goulib.math2), 199
 triangular() (in module Goulib.math2), 199
 triangular_repartition() (in module Goulib.math2), 201
 triples() (in module Goulib.math2), 194
 tsp() (in module Goulib.optim), 220
 TUE (Goulib.workdays.WorkCalendar attribute), 254
 twin_primes() (in module Goulib.math2), 196

tzinfo (Goulib.datetime2.datetime2 attribute), 17
 tzinfo (Goulib.datetime2.time2 attribute), 20
 tzname() (Goulib.datetime2.datetime2 method), 17
 tzname() (Goulib.datetime2.time2 method), 20

U

unescape() (in module Goulib.markup), 178
 unique() (Goulib.container.Sequence method), 14
 unique() (in module Goulib.itertools2), 167
 update() (Goulib.colors.Palette method), 8
 update() (Goulib.container.Record method), 12
 update() (Goulib.graph.index.Index method), 108
 update() (Goulib.interval.Intervals method), 159
 update() (Goulib.motion.Segments method), 205
 update() (Goulib.optim.BinDict method), 217
 utcfromtimestamp() (Goulib.datetime2.datetime2 method), 17
 utcnow() (Goulib.datetime2.datetime2 method), 17
 utcoffset() (Goulib.datetime2.datetime2 method), 17
 utcoffset() (Goulib.datetime2.time2 method), 20
 utctimetuple() (Goulib.datetime2.datetime2 method), 17

V

values() (Goulib.colors.Palette method), 10
 values() (Goulib.container.Record method), 12
 values() (Goulib.graph.index.Index method), 108
 values() (Goulib.optim.BinDict method), 217
 var (Goulib.stats.Discrete attribute), 234
 var (Goulib.stats.Normal attribute), 239
 var (Goulib.stats.PDF attribute), 236
 var (Goulib.stats.Stats attribute), 231
 var() (in module Goulib.stats), 230
 varDict() (Goulib.motion.Actuator method), 210
 variance (Goulib.stats.Discrete attribute), 234
 variance (Goulib.stats.Normal attribute), 239
 variance (Goulib.stats.PDF attribute), 236
 variance (Goulib.stats.Stats attribute), 231
 variance() (in module Goulib.stats), 230
 varNames() (Goulib.motion.Actuator method), 210
 varRowUnits() (Goulib.motion.Actuator method), 210
 vecadd() (in module Goulib.math2), 193
 veccompare() (in module Goulib.math2), 193
 vecdiv() (in module Goulib.math2), 193
 vecmul() (in module Goulib.math2), 193
 vecneg() (in module Goulib.math2), 193
 vecsub() (in module Goulib.math2), 193
 Vector2 (class in Goulib.geom), 65
 Vector3 (class in Goulib.geom3d), 94
 vecunit() (in module Goulib.math2), 193
 visit() (Goulib.expr.TextVisitor method), 63
 visit_BinOp() (Goulib.expr.TextVisitor method), 63
 visit_Call() (Goulib.expr.TextVisitor method), 63
 visit_Compare() (Goulib.expr.TextVisitor method), 63
 visit_Name() (Goulib.expr.TextVisitor method), 63

visit_NameConstant() (Goulib.expr.TextVisitor method), 63
visit_Num() (Goulib.expr.TextVisitor method), 63
visit_UnaryOp() (Goulib.expr.TextVisitor method), 63

W

wait() (Goulib.statemachine.StateMachine method), 230
WaitLog (class in Goulib.statemachine), 229
WED (Goulib.workdays.WorkCalendar attribute), 254
weekday() (Goulib.datetime2.date2 method), 19
weekday() (Goulib.datetime2.datetime2 method), 17
width (Goulib.drawing.BBox attribute), 25, 32
with_traceback() (Goulib.itertools2.SortingError method), 172
with_traceback() (Goulib.markup.ArgumentError method), 185
with_traceback() (Goulib.markup.ClosingError method), 183
with_traceback() (Goulib.markup.CustomizationError method), 190
with_traceback() (Goulib.markup.DeprecationError method), 187
with_traceback() (Goulib.markup.InvalidElementError method), 186
with_traceback() (Goulib.markup.MarkupError method), 182
with_traceback() (Goulib.markup.ModeError method), 189
with_traceback() (Goulib.markup.OpeningError method), 184
WorkCalendar (class in Goulib.workdays), 253
workdatetime() (Goulib.workdays.WorkCalendar method), 253
workday() (Goulib.workdays.WorkCalendar method), 253
workday() (in module Goulib.workdays), 255
workdays() (Goulib.workdays.WorkCalendar method), 253
worktime() (Goulib.workdays.WorkCalendar method), 253
write_csv() (Goulib.table.Table method), 243
write_dot() (in module Goulib.graph), 105, 145
write_dxf() (in module Goulib.graph), 105, 145
write_json() (in module Goulib.graph), 105, 145
write_xlsx() (Goulib.table.Table method), 243

X

xgcd() (in module Goulib.math2), 190
xmax (Goulib.drawing.BBox attribute), 25, 32
xmed (Goulib.drawing.BBox attribute), 25, 32
xmin (Goulib.drawing.BBox attribute), 25, 32
xy (Goulib.drawing.Spline attribute), 27, 37
xy (Goulib.geom.Point2 attribute), 74
xy (Goulib.geom.Vector2 attribute), 68

xyY (Goulib.colors.Color attribute), 7
xyy2xyz() (in module Goulib.colors), 6
xyz (Goulib.colors.Color attribute), 7
xyz (Goulib.geom3d.Vector3 attribute), 94
xyz2xyy() (in module Goulib.colors), 6

Y

year (Goulib.datetime2.date2 attribute), 19
year (Goulib.datetime2.datetime2 attribute), 17
ymax (Goulib.drawing.BBox attribute), 25, 32
ymed (Goulib.drawing.BBox attribute), 25, 32
ymin (Goulib.drawing.BBox attribute), 25, 32

Z

zeros() (in module Goulib.math2), 192