|  |  |
| --- | --- |
| **Functions** | |
|  | |  |  | | --- | --- | | [**DeprecationWarnings**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#DeprecationWarnings)() Have any DeprecationWarnings been reported or raised? |  | |
|  | |  |  | | --- | --- | | [**UtmUps**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#UtmUps)(zone, hemipole, easting, northing, band='', datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, falsed=True, name='') Class-like function to create a UTM/UPS coordinate. |  | |
|  | |  |  | | --- | --- | | [**a\_b2e**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2e)(a, b) Return e, the *1st eccentricity* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2e2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2e2)(a, b) Return e2, the *1st eccentricity squared* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2e22**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2e22)(a, b) Return e22, the *2nd eccentricity squared* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2e32**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2e32)(a, b) Return e32, the *3rd eccentricity squared* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2f**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2f)(a, b) Return f, the *flattening* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2f2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2f2)(a, b) Return f2, the *2nd flattening* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2f\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2f_)(a, b) Return f\_, the *inverse flattening* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_b2n**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_b2n)(a, b) Return n, the *3rd flattening* for a given *equatorial* and *polar* radius. |  | |
|  | |  |  | | --- | --- | | [**a\_f2b**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_f2b)(a, f) Return b, the *polar* radius for a given *equatorial* radius and *flattening*. |  | |
|  | |  |  | | --- | --- | | [**a\_f\_2b**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#a_f_2b)(a, f\_) Return b, the *polar* radius for a given *equatorial* radius and *inverse flattening*. |  | |
|  | |  |  | | --- | --- | | **acos1**(x) Return math.acos(max(-1, min(1, **x**))). |  | |
|  | |  |  | | --- | --- | | [**acre2ha**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#acre2ha)(acres) Convert acres to hectare. |  | |
|  | |  |  | | --- | --- | | [**acre2m2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#acre2m2)(acres) Convert acres to *square* meter. |  | |
|  | |  |  | | --- | --- | | [**anStr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#anStr)(name, OKd=**'**.\_-**'**, sub='\_') DEPRECATED, use function [pygeodesy.anstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "anstr). |  | |
|  | |  |  | | --- | --- | | [**anstr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#anstr)(name, OKd=**'**.\_-**'**, sub='\_') Make a valid name of alphanumeric and OKd characters. |  | |
|  | |  |  | | --- | --- | | [**antipode**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#antipode)(lat, lon) Return the antipode, the point diametrically opposite to a given point in degrees. |  | |
|  | |  |  | | --- | --- | | [**antipode\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#antipode_)(phi, lam) Return the antipode, the point diametrically opposite to a given point in radians. |  | |
|  | |  |  | | --- | --- | | [**areaOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#areaOf)(points, adjust=True, radius=6371008.77141, wrap=True) Approximate the area of a polygon. |  | |
|  | |  |  | | --- | --- | | [**areaof**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#areaof)(points, adjust=True, radius=6371008.77141, wrap=True) DEPRECATED, use function [pygeodesy.areaOf](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html" \l "areaOf). |  | |
|  | |  |  | | --- | --- | | **asin1**(x) Return math.asin(max(-1, min(1, **x**))). |  | |
|  | |  |  | | --- | --- | | [**atan2b**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#atan2b)(y, x) Return atan2(**y**, **x**) in degrees *[0..+360]*. |  | |
|  | |  |  | | --- | --- | | [**atan2d**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#atan2d)(y, x, reverse=False) Return atan2(**y**, **x**) in degrees *[-180..+180]*, optionally reversed (by 180 degrees for azi2). |  | |
|  | |  |  | | --- | --- | | [**atand**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#atand)(y\_x) Return atan(**y\_x**) angle in degrees. |  | |
|  | |  |  | | --- | --- | | [**attrs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#attrs)(inst, \*names, \*\*Nones\_True\_pairs\_kwds) Get instance attributes as *name=value* strings, with floats formatted by function [fstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstr). |  | |
|  | |  |  | | --- | --- | | [**b\_f2a**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#b_f2a)(b, f) Return a, the *equatorial* radius for a given *polar* radius and *flattening*. |  | |
|  | |  |  | | --- | --- | | [**b\_f\_2a**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#b_f_2a)(b, f\_) Return a, the *equatorial* radius for a given *polar* radius and *inverse flattening*. |  | |
|  | |  |  | | --- | --- | | [**bearing**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#bearing)(lat1, lon1, lat2, lon2, \*\*options) Compute the initial or final bearing (forward or reverse azimuth) between a (spherical) start and end point. |  | |
|  | |  |  | | --- | --- | | [**bearingDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#bearingDMS)(bearing, form=**'**d**'**, prec=None, sep='') Convert bearing to a string (without compass point suffix). |  | |
|  | |  |  | | --- | --- | | [**bearing\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#bearing_)(phi1, lam1, phi2, lam2, final=False, wrap=False) Compute the initial or final bearing (forward or reverse azimuth) between a (spherical) start and end point. |  | |
|  | |  |  | | --- | --- | | [**bounds**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#bounds)(points, wrap=True, LatLon=None) DEPRECATED, use function [pygeodesy.boundsOf](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html" \l "boundsOf). |  | |
|  | |  |  | | --- | --- | | [**boundsOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#boundsOf)(points, wrap=True, LatLon=None) Determine the lower-left SW and upper-right NE corners of a path or polygon. |  | |
|  | |  |  | | --- | --- | | [**callername**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#callername)(up=1, dflt='', source=False, underOK=False) Get the name of the invoking callable. |  | |
|  | |  |  | | --- | --- | | [**cassini**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cassini)(pointA, pointB, pointC, alpha, beta, useZ=False, Clas=None, \*\*Clas\_kwds) 3-Point resection using [Cassini](https://nl.wikipedia.org/wiki/Achterwaartse_insnijding)'s method. |  | |
|  | |  |  | | --- | --- | | [**cbrt**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cbrt)(x3) Compute the cube root *x3\*\*(1/3)*. |  | |
|  | |  |  | | --- | --- | | [**cbrt2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cbrt2)(x3) Compute the cube root *squared* *x3\*\*(2/3)*. |  | |
|  | |  |  | | --- | --- | | [**centroidOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#centroidOf)(points, wrap=True, LatLon=None) Determine the centroid of a polygon. |  | |
|  | |  |  | | --- | --- | | [**chain2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#chain2m)(chains) Convert *UK* chains to meter. |  | |
|  | |  |  | | --- | --- | | [**circin6**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#circin6)(point1, point2, point3, eps=8.881784197e-16, useZ=True) Return the radius and center of the *inscribed* aka *In- circle* of a (2- or 3-D) triangle. |  | |
|  | |  |  | | --- | --- | | [**circle4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#circle4)(earth, lat) Get the equatorial or a parallel *circle of latitude*. |  | |
|  | |  |  | | --- | --- | | [**circum3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#circum3)(point1, point2, point3, circum=True, eps=8.881784197e-16, useZ=True) Return the radius and center of the smallest circle *through* or *containing* three (2- or 3-D) points. |  | |
|  | |  |  | | --- | --- | | [**circum4\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#circum4_)(\*points, \*\*Vector\_and\_kwds) Best-fit a sphere through three or more (3-D) points. |  | |
|  | |  |  | | --- | --- | | [**classname**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#classname)(inst, prefixed=None) Return the instance' class name optionally prefixed with the module name. |  | |
|  | |  |  | | --- | --- | | [**classnaming**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#classnaming)(prefixed=None) Get/set the default class naming for [module.]class names. |  | |
|  | |  |  | | --- | --- | | [**clipCS3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipCS3)(points, lowerleft, upperright, closed=False, inull=False) DEPRECATED, use function [pygeodesy.clipCS4](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.clipy-module.html#clipCS4). |  | |
|  | |  |  | | --- | --- | | [**clipCS4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipCS4)(points, lowerleft, upperright, closed=False, inull=False) Clip a path against a rectangular clip box using the [Cohen-Sutherland](https://wikipedia.org/wiki/Cohen-Sutherland_algorithm) algorithm. |  | |
|  | |  |  | | --- | --- | | [**clipDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipDMS)(deg, limit) DEPRECATED, use function [pygeodesy.clipDegrees](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.dms-module.html" \l "clipDegrees). |  | |
|  | |  |  | | --- | --- | | [**clipDegrees**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipDegrees)(deg, limit) Clip a lat- or longitude to the given range. |  | |
|  | |  |  | | --- | --- | | [**clipLB6**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipLB6)(points, lowerleft, upperright, closed=False, inull=False) Clip a path against a rectangular clip box using the [Liang-Barsky](https://www.cse.unt.edu/~renka/4230/LineClipping.pdf) algorithm. |  | |
|  | |  |  | | --- | --- | | [**clipRadians**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipRadians)(rad, limit) Clip a lat- or longitude to the given range. |  | |
|  | |  |  | | --- | --- | | [**clipSH**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipSH)(points, corners, closed=False, inull=False) Clip a polygon against a clip region or box using the [Sutherland-Hodgman](https://wikipedia.org/wiki/Sutherland-Hodgman_algorithm) algorithm. |  | |
|  | |  |  | | --- | --- | | [**clipSH3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipSH3)(points, corners, closed=False, inull=False) Clip a polygon against a clip region or box using the [Sutherland-Hodgman](https://wikipedia.org/wiki/Sutherland-Hodgman_algorithm) algorithm. |  | |
|  | |  |  | | --- | --- | | [**clipStr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clipStr)(bstr, limit=50, white='') DEPRECATED, use function [pygeodesy.clips](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.basics-module.html" \l "clips). |  | |
|  | |  |  | | --- | --- | | [**clips**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#clips)(bstr, limit=50, white='') Clip a string to the given length limit. |  | |
|  | |  |  | | --- | --- | | [**collins**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#collins)(pointA, pointB, pointC, alpha, beta, useZ=False, Clas=None, \*\*Clas\_kwds) 3-Point resection using [Collins](https://dokumen.tips/documents/three-point-resection-problem-introduction-kaestner-burkhardt-method.html)' method. |  | |
|  | |  |  | | --- | --- | | [**compassAngle**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#compassAngle)(lat1, lon1, lat2, lon2, adjust=True, wrap=False) Return the angle from North for the direction vector *(lon2 - lon1, lat2 - lat1)* between two points. |  | |
|  | |  |  | | --- | --- | | [**compassDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#compassDMS)(bearing, form=**'**d**'**, prec=None, sep='') Convert bearing to a string suffixed with compass point. |  | |
|  | |  |  | | --- | --- | | [**compassPoint**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#compassPoint)(bearing, prec=3) Convert bearing to a compass point. |  | |
|  | |  |  | | --- | --- | | [**copysign0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#copysign0)(x, y) Like math.copysign(x, y) except zero, *unsigned*. |  | |
|  | |  |  | | --- | --- | | [**copytype**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#copytype)(x, y) Return the value of **x** as type of y. |  | |
|  | |  |  | | --- | --- | | [**cosineAndoyerLambert**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineAndoyerLambert)(lat1, lon1, lat2, lon2, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, wrap=False) Compute the distance between two (ellipsoidal) points using the [Andoyer-Lambert correction](https://navlib.net/wp-content/uploads/2013/10/admiralty-manual-of-navigation-vol-1-1964-english501c.pdf" \t "_top) of the [Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) fromula. |  | |
|  | |  |  | | --- | --- | | [**cosineAndoyerLambert\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineAndoyerLambert_)(phi2, phi1, lam21, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**) Compute the *angular* distance between two (ellipsoidal) points using the [Andoyer-Lambert correction](https://navlib.net/wp-content/uploads/2013/10/admiralty-manual-of-navigation-vol-1-1964-english501c.pdf" \t "_top) of the [Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) fromula. |  | |
|  | |  |  | | --- | --- | | [**cosineForsytheAndoyerLambert**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineForsytheAndoyerLambert)(lat1, lon1, lat2, lon2, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, wrap=False) Compute the distance between two (ellipsoidal) points using the [Forsythe-Andoyer-Lambert correction](https://www2.unb.ca/gge/Pubs/TR77.pdf) of the [Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) formula. |  | |
|  | |  |  | | --- | --- | | [**cosineForsytheAndoyerLambert\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineForsytheAndoyerLambert_)(phi2, phi1, lam21, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**) Compute the *angular* distance between two (ellipsoidal) points using the [Forsythe-Andoyer-Lambert correction](https://www2.unb.ca/gge/Pubs/TR77.pdf) of the [Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) formula. |  | |
|  | |  |  | | --- | --- | | [**cosineLaw**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineLaw)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the distance between two points using the [spherical Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) formula. |  | |
|  | |  |  | | --- | --- | | [**cosineLaw\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cosineLaw_)(phi2, phi1, lam21) Compute the *angular* distance between two points using the [spherical Law of Cosines](https://www.movable-type.co.uk/scripts/latlong.html#cosine-law) formula. |  | |
|  | |  |  | | --- | --- | | [**cot**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cot)(rad, \*\*error\_kwds) Return the cotangent of an angle in radians. |  | |
|  | |  |  | | --- | --- | | [**cot\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cot_)(\*rads, \*\*error\_kwds) Return the cotangent of angle(s) in radiansresection. |  | |
|  | |  |  | | --- | --- | | [**cotd**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cotd)(deg, \*\*error\_kwds) Return the cotangent of an angle in degrees. |  | |
|  | |  |  | | --- | --- | | [**cotd\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#cotd_)(\*degs, \*\*error\_kwds) Return the cotangent of angle(s) in degrees. |  | |
|  | |  |  | | --- | --- | | [**crosserrors**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#crosserrors)(raiser=None) Report or ignore vectorial cross product errors. |  | |
|  | |  |  | | --- | --- | | [**date2epoch**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#date2epoch)(year, month, day) Return the reference frame epoch for a calendar day. |  | |
|  | |  |  | | --- | --- | | [**decodeEPSG2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#decodeEPSG2)(arg) DEPRECATED, use function [epsg.decode2](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.epsg-module.html#decode2). |  | |
|  | |  |  | | --- | --- | | [**degDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degDMS)(deg, prec=6, s\_D=**'**°**'**, s\_M=**'**\xe2\x80\xb2**'**, s\_S=**'**″**'**, neg='-', pos='') Convert degrees to a string in degrees, minutes *or* seconds. |  | |
|  | |  |  | | --- | --- | | **degrees**(x) Convert angle x from radians to degrees. |  | |
|  | |  |  | | --- | --- | | [**degrees180**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degrees180)(rad) Convert radians to degrees and wrap *[-180..+180]*. |  | |
|  | |  |  | | --- | --- | | [**degrees2grades**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degrees2grades)(deg) Convert degrees to *grades* (aka *gons* or *gradians*). |  | |
|  | |  |  | | --- | --- | | [**degrees2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degrees2m)(deg, radius=6371008.77141, lat=0) Convert an angle to a distance along the equator or along the parallel at an other (geodetic) latitude. |  | |
|  | |  |  | | --- | --- | | [**degrees360**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degrees360)(rad) Convert radians to degrees and wrap *[0..+360)*. |  | |
|  | |  |  | | --- | --- | | [**degrees90**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#degrees90)(rad) Convert radians to degrees and wrap *[-270..+90]*. |  | |
|  | |  |  | | --- | --- | | [**deprecated\_Property\_RO**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#deprecated_Property_RO)(method) Decorator for a DEPRECATED [Property\_RO](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.props.Property_RO-class.html). |  | |
|  | |  |  | | --- | --- | | [**deprecated\_class**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#deprecated_class)(cls\_or\_class) Use inside \_\_new\_\_ or \_\_init\_\_ of a DEPRECATED class. |  | |
|  | |  |  | | --- | --- | | [**deprecated\_function**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#deprecated_function)(call) Decorator for a DEPRECATED function. |  | |
|  | |  |  | | --- | --- | | [**deprecated\_method**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#deprecated_method)(call) Decorator for a DEPRECATED method. |  | |
|  | |  |  | | --- | --- | | [**deprecated\_property\_RO**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#deprecated_property_RO)(method) Decorator for a DEPRECATED [property\_RO](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.props.property_RO-class.html). |  | |
|  | |  |  | | --- | --- | | [**egmGeoidHeights**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#egmGeoidHeights)(GeoidHeights\_dat) Generate geoid [egm\*.pgm](https://geographiclib.sourceforge.io/html/geoid.html" \l "geoidinst" \t "_top) height tests from [GeoidHeights.dat](https://sourceforge.net/projects/geographiclib/files/testdata/) [Test data for Geoids](https://geographiclib.sourceforge.io/html/geoid.html#testgeoid). |  | |
|  | |  |  | | --- | --- | | [**elevation2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#elevation2)(lat, lon, timeout=2.0) Get the geoid elevation at an NAD83 to NAVD88 location. |  | |
|  | |  |  | | --- | --- | | [**enStr2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#enStr2)(easting, northing, prec, \*extras) DEPRECATED, use function [pygeodesy.enstr2](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html#enstr2). |  | |
|  | |  |  | | --- | --- | | [**encodeEPSG**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#encodeEPSG)(zone, hemipole='', band='') DEPRECATED, use function [epsg.encode](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.epsg-module.html" \l "encode). |  | |
|  | |  |  | | --- | --- | | [**enstr2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#enstr2)(easting, northing, prec, \*extras) Return easting, northing string representations. |  | |
|  | |  |  | | --- | --- | | [**epoch2date**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#epoch2date)(epoch) Return the date for a reference frame epoch. |  | |
|  | |  |  | | --- | --- | | [**equidistant**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#equidistant)(lat0, lon0, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, exact=False, geodsolve=False, name='') Return an [EquidistantExact](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.EquidistantExact-class.html), [EquidistantGeodSolve](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.EquidistantGeodSolve-class.html) or (if *Karney*'s [geographiclib](https://pypi.org/project/geographiclib" \t "_top) package is installed) an [EquidistantKarney](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.EquidistantKarney-class.html), otherwise an [Equidistant](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.Equidistant-class.html) instance. |  | |
|  | |  |  | | --- | --- | | [**equirectangular**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#equirectangular)(lat1, lon1, lat2, lon2, radius=6371008.77141, \*\*options) Compute the distance between two points using the [Equirectangular Approximation / Projection](https://www.movable-type.co.uk/scripts/latlong.html#equirectangular). |  | |
|  | |  |  | | --- | --- | | [**equirectangular3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#equirectangular3)(lat1, lon1, lat2, lon2, \*\*options) DEPRECATED, use function equirectangular\_. |  | |
|  | |  |  | | --- | --- | | [**equirectangular\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#equirectangular_)(lat1, lon1, lat2, lon2, adjust=True, limit=45, wrap=False) Compute the distance between two points using the [Equirectangular Approximation / Projection](https://www.movable-type.co.uk/scripts/latlong.html#equirectangular). |  | |
|  | |  |  | | --- | --- | | [**euclid**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#euclid)(x, y) *Appoximate* the norm *sqrt(x\*\*2 + y\*\*2)* by *max(abs(x), abs(y)) + min(abs(x), abs(y)) \* 0.4142...*. |  | |
|  | |  |  | | --- | --- | | [**euclid\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#euclid_)(\*xs) *Appoximate* the norm *sqrt(sum(x\*\*2 for x in xs))* by cascaded [euclid](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.fmath-module.html" \l "euclid). |  | |
|  | |  |  | | --- | --- | | [**euclidean**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#euclidean)(lat1, lon1, lat2, lon2, radius=6371008.77141, adjust=True, wrap=False) Approximate the Euclidean distance between two (spherical) points. |  | |
|  | |  |  | | --- | --- | | [**euclidean\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#euclidean_)(phi2, phi1, lam21, adjust=True) Approximate the *angular* Euclidean distance between two (spherical) points. |  | |
|  | |  |  | | --- | --- | | [**exception\_chaining**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#exception_chaining)(error=None) Get the previous exception's or exception chaining setting. |  | |
|  | |  |  | | --- | --- | | [**excessAbc**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessAbc)(A, b, c) Compute the *spherical excess* E of a (spherical) triangle from two sides and the included angle. |  | |
|  | |  |  | | --- | --- | | [**excessGirard**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessGirard)(A, B, C) Compute the *spherical excess* E of a (spherical) triangle using [Girard's](https://mathworld.wolfram.com/GirardsSphericalExcessFormula.html) formula. |  | |
|  | |  |  | | --- | --- | | [**excessKarney**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessKarney)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the surface area of a (spherical) quadrilateral bounded by a segment of a great circle, two meridians and the equator using [Karney's](http://osgeo-org.1560.x6.nabble.com/Area-of-a-spherical-polygon-td3841625.html" \t "_top) method. |  | |
|  | |  |  | | --- | --- | | [**excessKarney\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessKarney_)(phi2, phi1, lam21) Compute the *spherical excess* E of a (spherical) quadrilateral bounded by a segment of a great circle, two meridians and the equator using [Karney's](http://osgeo-org.1560.x6.nabble.com/Area-of-a-spherical-polygon-td3841625.html" \t "_top) method. |  | |
|  | |  |  | | --- | --- | | [**excessLHuilier**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessLHuilier)(a, b, c) Compute the *spherical excess* E of a (spherical) triangle using [L'Huilier's](https://mathworld.wolfram.com/LHuiliersTheorem.html" \t "_top) Theorem. |  | |
|  | |  |  | | --- | --- | | [**excessQuad**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessQuad)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the surface area of a (spherical) quadrilateral bounded by a segment of a great circle, two meridians and the equator. |  | |
|  | |  |  | | --- | --- | | [**excessQuad\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#excessQuad_)(phi2, phi1, lam21) Compute the *spherical excess* E of a (spherical) quadrilateral bounded by a segment of a great circle, two meridians and the equator. |  | |
|  | |  |  | | --- | --- | | [**f2e2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2e2)(f) Return e2, the *1st eccentricity squared* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**f2e22**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2e22)(f) Return e22, the *2nd eccentricity squared* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**f2e32**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2e32)(f) Return e32, the *3rd eccentricity squared* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**f2f2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2f2)(f) Return f2, the *2nd flattening* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**f2f\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2f_)(f) Return f\_, the *inverse flattening* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**f2n**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f2n)(f) Return n, the *3rd flattening* for a given *flattening*. |  | |
|  | |  |  | | --- | --- | | [**fStr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fStr)(floats, prec=6, fmt='f', ints=False, sep=', ') DEPRECATED, use function [fstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstr). |  | |
|  | |  |  | | --- | --- | | [**fStrzs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fStrzs)(floatstr) DEPRECATED, use function [pygeodesy.fstrzs](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstrzs). |  | |
|  | |  |  | | --- | --- | | [**f\_2f**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#f_2f)(f\_) Return f, the *flattening* for a given *inverse flattening*. |  | |
|  | |  |  | | --- | --- | | [**facos1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#facos1)(x) Fast approximation of [pygeodesy.acos1](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.utily-module.html#acos1)(**x**). |  | |
|  | |  |  | | --- | --- | | [**false2f**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#false2f)(value, name=**'**value**'**, false=True, Error=<type 'exceptions.ValueError'>) DEPRECATED, use function [falsed2f](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.deprecated-module.html#falsed2f). |  | |
|  | |  |  | | --- | --- | | [**falsed2f**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#falsed2f)(falsed=True, Error=<type 'exceptions.ValueError'>, \*\*name\_value) DEPRECATED, use class [Easting](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.units.Easting-class.html) or [Northing](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.units.Northing-class.html). |  | |
|  | |  |  | | --- | --- | | [**fasin1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fasin1)(x) Fast approximation of [pygeodesy.asin1](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.utily-module.html#asin1)(**x**). |  | |
|  | |  |  | | --- | --- | | **fatan**(x) Fast approximation of atan(**x**). |  | |
|  | |  |  | | --- | --- | | [**fatan1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fatan1)(x) Fast approximation of atan(**x**) for 0 <= **x** <= 1, *unchecked*. |  | |
|  | |  |  | | --- | --- | | [**fatan2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fatan2)(y, x) Fast approximation of atan2(**y**, **x**). |  | |
|  | |  |  | | --- | --- | | [**fathom2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fathom2m)(fathoms) Convert *UK* fathom to meter. |  | |
|  | |  |  | | --- | --- | | [**favg**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#favg)(v1, v2, f=0.5) Return the average of two values. |  | |
|  | |  |  | | --- | --- | | [**fdot**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fdot)(a, \*b) Return the precision dot product *sum(a[i] \* b[i] for i=0..len(a))*. |  | |
|  | |  |  | | --- | --- | | [**fdot3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fdot3)(a, b, c, start=0) Return the precision dot product *start + sum(a[i] \* b[i] \* c[i] for i=0..len(a))*. |  | |
|  | |  |  | | --- | --- | | [**fhorner**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fhorner)(x, \*cs) Evaluate the polynomial *sum(cs[i] \* x\*\*i for i=0..len(cs))* using the Horner form. |  | |
|  | |  |  | | --- | --- | | [**fidw**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fidw)(xs, ds, beta=2) Interpolate using using [Inverse Distance Weighting](https://wikipedia.org/wiki/Inverse_distance_weighting) (IDW). |  | |
|  | |  |  | | --- | --- | | [**flatLocal**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#flatLocal)(lat1, lon1, lat2, lon2, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, wrap=False) Compute the distance between two (ellipsoidal) points using the [ellipsoidal Earth to plane projection](https://wikipedia.org/wiki/Geographical_distance#Ellipsoidal_Earth_projected_to_a_plane) aka [Hubeny](https://www.ovg.at/de/vgi/files/pdf/3781/" \t "_top) formula. |  | |
|  | |  |  | | --- | --- | | [**flatLocal\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#flatLocal_)(phi2, phi1, lam21, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**) Compute the *angular* distance between two (ellipsoidal) points using the [ellipsoidal Earth to plane projection](https://wikipedia.org/wiki/Geographical_distance#Ellipsoidal_Earth_projected_to_a_plane) aka [Hubeny](https://www.ovg.at/de/vgi/files/pdf/3781/" \t "_top) formula. |  | |
|  | |  |  | | --- | --- | | [**flatPolar**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#flatPolar)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the distance between two (spherical) points using the [polar coordinate flat-Earth](https://wikipedia.org/wiki/Geographical_distance#Polar_coordinate_flat-Earth_formula) formula. |  | |
|  | |  |  | | --- | --- | | [**flatPolar\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#flatPolar_)(phi2, phi1, lam21) Compute the *angular* distance between two (spherical) points using the [polar coordinate flat-Earth](https://wikipedia.org/wiki/Geographical_distance#Polar_coordinate_flat-Earth_formula) formula. |  | |
|  | |  |  | | --- | --- | | [**fmean**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fmean)(xs) Compute the accurate mean *sum(xs[i] for i=0..len(xs)) / len(xs)*. |  | |
|  | |  |  | | --- | --- | | [**fmean\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fmean_)(\*xs) Compute the accurate mean *sum(xs[i] for i=0..len(xs)) / len(xs)*. |  | |
|  | |  |  | | --- | --- | | [**fpolynomial**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fpolynomial)(x, \*cs) Evaluate the polynomial *sum(cs[i] \* x\*\*i for i=0..len(cs))*. |  | |
|  | |  |  | | --- | --- | | [**fpowers**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fpowers)(x, n, alts=0) Return a series of powers *[x\*\*i for i=1..n]*. |  | |
|  | |  |  | | --- | --- | | [**fprod**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fprod)(iterable, start=1.0) Iterable product, like math.prod or numpy.prod. |  | |
|  | |  |  | | --- | --- | | [**fractional**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fractional)(points, fi, j=None, wrap=None, LatLon=None, Vector=None, \*\*kwds) Return the point at a given *fractional* index. |  | |
|  | |  |  | | --- | --- | | [**frange**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#frange)(start, number, step=1) Generate a range of floats. |  | |
|  | |  |  | | --- | --- | | [**frechet\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#frechet_)(points1, points2, distance=None, units='') Compute the *discrete* [Fréchet](https://wikipedia.org/wiki/Frechet_distance) distance between two paths given as sets of points. |  | |
| value | |  |  | | --- | --- | | [**freduce**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#freduce)(function, sequence, initial=...) Apply a function of two arguments cumulatively to the items of a sequence, from left to right, so as to reduce the sequence to a single value. |  | |
|  | |  |  | | --- | --- | | [**fstr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fstr)(floats, prec=6, fmt='F', ints=False, sep=', ', strepr=None) Convert one or more floats to string, optionally stripped of trailing zero decimals. |  | |
|  | |  |  | | --- | --- | | [**fstrzs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fstrzs)(efstr, ap1z=False) Strip trailing zero decimals from a float string. |  | |
|  | |  |  | | --- | --- | | [**fsum**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fsum)(iterable) Return an accurate floating point sum of values in the iterable. |  | |
|  | |  |  | | --- | --- | | [**fsum1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fsum1)(iterable) Precision summation, primed with 1.0. |  | |
|  | |  |  | | --- | --- | | [**fsum1\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fsum1_)(\*xs) Precision summation of a few arguments, primed with 1.0. |  | |
|  | |  |  | | --- | --- | | [**fsum\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#fsum_)(\*xs) Precision summation of all positional arguments. |  | |
|  | |  |  | | --- | --- | | [**ft2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#ft2m)(feet, usurvey=False) Convert *International* or *US Survey* feet to meter. |  | |
|  | |  |  | | --- | --- | | [**furlong2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#furlong2m)(furlongs) Convert a furlong to meter. |  | |
|  | |  |  | | --- | --- | | [**geoidHeight2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#geoidHeight2)(lat, lon, model=0, timeout=2.0) Get the NAVD88 geoid height at an NAD83 location. |  | |
|  | |  |  | | --- | --- | | [**gnomonic**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#gnomonic)(lat0, lon0, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, exact=False, geodsolve=False, name='') Return a [GnomonicExact](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.GnomonicExact-class.html) or (if *Karney*'s [geographiclib](https://pypi.org/project/geographiclib" \t "_top) package is installed) a [GnomonicKarney](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.GnomonicKarney-class.html), otherwise a [Gnomonic](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.azimuthal.Gnomonic-class.html) instance. |  | |
|  | |  |  | | --- | --- | | [**grades**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#grades)(rad) Convert radians to *grades* (aka *gons* or *gradians*). |  | |
|  | |  |  | | --- | --- | | [**grades2degrees**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#grades2degrees)(gon) Convert *grades* (aka *gons* or *gradians*) to degrees. |  | |
|  | |  |  | | --- | --- | | [**grades2radians**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#grades2radians)(gon) Convert *grades* (aka *gons* or *gradians*) to radians. |  | |
|  | |  |  | | --- | --- | | [**grades400**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#grades400)(rad) Convert radians to *grades* (aka *gons* or *gradians*) and wrap *[0..+400)*. |  | |
|  | |  |  | | --- | --- | | [**halfs2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#halfs2)(str2) Split a string in 2 halfs. |  | |
|  | |  |  | | --- | --- | | [**hartzell**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hartzell)(pov, los=None, earth=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, \*\*LatLon\_and\_kwds) Compute the intersection of a Line-Of-Sight from a Point-Of-View in space with the surface of the earth. |  | |
|  | |  |  | | --- | --- | | [**hausdorff\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hausdorff_)(model, target, both=False, early=True, seed=None, units='', distance=None, point=<function \_point at 0x7fd078574dd0>) Compute the directed or symmetric [Hausdorff](https://wikipedia.org/wiki/Hausdorff_distance" \t "_top) distance between 2 sets of points with or without [early breaking](https://publik.tuwien.ac.at/files/PubDat_247739.pdf) and [random sampling](https://publik.tuwien.ac.at/files/PubDat_247739.pdf). |  | |
|  | |  |  | | --- | --- | | [**haversine**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#haversine)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the distance between two (spherical) points using the [Haversine](https://www.movable-type.co.uk/scripts/latlong.html) formula. |  | |
|  | |  |  | | --- | --- | | [**haversine\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#haversine_)(phi2, phi1, lam21) Compute the *angular* distance between two (spherical) points using the [Haversine](https://www.movable-type.co.uk/scripts/latlong.html) formula. |  | |
|  | |  |  | | --- | --- | | [**heightOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#heightOf)(angle, distance, radius=6371008.77141) Determine the height above the (spherical) earth' surface after traveling along a straight line at a given tilt. |  | |
|  | |  |  | | --- | --- | | [**horizon**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#horizon)(height, radius=6371008.77141, refraction=False) Determine the distance to the horizon from a given altitude above the (spherical) earth. |  | |
|  | |  |  | | --- | --- | | [**hstr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hstr)(height, prec=2, fmt='%+.\*f', ints=False, m='') Return a string for the height value. |  | |
|  | |  |  | | --- | --- | | [**hubeny**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hubeny)(lat1, lon1, lat2, lon2, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, wrap=False) Compute the distance between two (ellipsoidal) points using the [ellipsoidal Earth to plane projection](https://wikipedia.org/wiki/Geographical_distance#Ellipsoidal_Earth_projected_to_a_plane) aka [Hubeny](https://www.ovg.at/de/vgi/files/pdf/3781/" \t "_top) formula. |  | |
|  | |  |  | | --- | --- | | [**hubeny\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hubeny_)(phi2, phi1, lam21, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**) Compute the *angular* distance between two (ellipsoidal) points using the [ellipsoidal Earth to plane projection](https://wikipedia.org/wiki/Geographical_distance#Ellipsoidal_Earth_projected_to_a_plane) aka [Hubeny](https://www.ovg.at/de/vgi/files/pdf/3781/" \t "_top) formula. |  | |
|  | |  |  | | --- | --- | | **hypot**(x, y) Return the Euclidean distance, sqrt(x\*x + y\*y). |  | |
|  | |  |  | | --- | --- | | [**hypot1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hypot1)(x) Compute the norm *sqrt(1 + x\*\*2)*. |  | |
|  | |  |  | | --- | --- | | [**hypot2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hypot2)(x, y) Compute the *squared* norm *x\*\*2 + y\*\*2*. |  | |
|  | |  |  | | --- | --- | | [**hypot2\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hypot2_)(\*xs) Compute the *squared* norm sum(x\*\*2 for x in **xs**). |  | |
|  | |  |  | | --- | --- | | [**hypot3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hypot3)(x, y, z) DEPRECATED, use function [pygeodesy.hypot\_](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.fmath-module.html" \l "hypot_). |  | |
|  | |  |  | | --- | --- | | [**hypot\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#hypot_)(\*xs) Compute the norm *sqrt(sum(x\*\*2 for x in xs))*. |  | |
|  | |  |  | | --- | --- | | [**inStr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#inStr)(inst, \*args, \*\*kwds) DEPRECATED, use function [pygeodesy.instr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "instr). |  | |
|  | |  |  | | --- | --- | | [**instr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#instr)(inst, \*args, \*\*kwds) Return the string representation of an instantiation. |  | |
|  | |  |  | | --- | --- | | [**intersection3d3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#intersection3d3)(start1, end1, start2, end2, eps=2.22044604925e-16, useZ=True, \*\*Vector\_and\_kwds) Compute the intersection point of two lines, each defined by or through a start and end point (3-D). |  | |
|  | |  |  | | --- | --- | | [**intersections2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#intersections2)(center1, radius1, center2, radius2, sphere=True, \*\*Vector\_and\_kwds) Compute the intersection of two spheres or circles, each defined by a (3-D) center point and a radius. |  | |
|  | |  |  | | --- | --- | | [**isNumpy2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isNumpy2)(obj) Check for an **Numpy2LatLon** points wrapper. |  | |
|  | |  |  | | --- | --- | | [**isPoints2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isPoints2)(obj) Check for an **LatLon2psxy** points wrapper. |  | |
|  | |  |  | | --- | --- | | [**isTuple2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isTuple2)(obj) Check for an **Tuple2LatLon** points wrapper. |  | |
|  | |  |  | | --- | --- | | [**isantipode**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isantipode)(lat1, lon1, lat2, lon2, eps=2.22044604925e-16) Check whether two points are antipodal, on diametrically opposite sides of the earth. |  | |
|  | |  |  | | --- | --- | | [**isantipode\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isantipode_)(phi1, lam1, phi2, lam2, eps=2.22044604925e-16) Check whether two points are antipodal, on diametrically opposite sides of the earth. |  | |
|  | |  |  | | --- | --- | | [**isbool**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isbool)(obj) Check whether an object is boolean. |  | |
|  | |  |  | | --- | --- | | [**isclass**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isclass)(obj) Return True if **obj** is a class. |  | |
|  | |  |  | | --- | --- | | [**isclockwise**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isclockwise)(points, adjust=False, wrap=True) Determine the direction of a path or polygon. |  | |
|  | |  |  | | --- | --- | | [**iscolinearWith**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#iscolinearWith)(point, point1, point2, eps=2.22044604925e-16, useZ=True) Check whether a point is colinear with two other (2- or 3-D) points. |  | |
|  | |  |  | | --- | --- | | [**isconvex**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isconvex)(points, adjust=False, wrap=True) Determine whether a polygon is convex. |  | |
|  | |  |  | | --- | --- | | [**isconvex\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isconvex_)(points, adjust=False, wrap=True) Determine whether a polygon is convex *and clockwise*. |  | |
|  | |  |  | | --- | --- | | [**isenclosedBy**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isenclosedBy)(point, points, wrap=False) Determine whether a point is enclosed by a polygon. |  | |
|  | |  |  | | --- | --- | | [**isenclosedby**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isenclosedby)(point, points, wrap=False) DEPRECATED, use function [pygeodesy.isenclosedBy](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html" \l "isenclosedBy). |  | |
|  | |  |  | | --- | --- | | [**isfinite**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isfinite)(obj) Check for Inf and NaN values. |  | |
|  | |  |  | | --- | --- | | **isidentifier**(obj) Return True if **obj** is a valid Python identifier. |  | |
| bool | |  |  | | --- | --- | | **isinf**(x) Check if float x is infinite (positive or negative). |  | |
|  | |  |  | | --- | --- | | [**isint**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isint)(obj, both=False) Check for int type or an integer float value. |  | |
|  | |  |  | | --- | --- | | **iskeyword**(x, y) y in x. |  | |
| bool | |  |  | | --- | --- | | **isnan**(x) Check if float x is not a number (NaN). |  | |
|  | |  |  | | --- | --- | | [**isnear0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isnear0)(x, eps0=4.93038065763e-32) Is **x** near zero? |  | |
|  | |  |  | | --- | --- | | [**isnear1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isnear1)(x, eps0=4.93038065763e-32) Is **x** near one? |  | |
|  | |  |  | | --- | --- | | [**isneg0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isneg0)(x) Check for NEG0, negative 0.0. |  | |
|  | |  |  | | --- | --- | | [**isnon0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isnon0)(x, eps0=4.93038065763e-32) Is **x** non-zero? |  | |
|  | |  |  | | --- | --- | | [**isodd**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isodd)(x) Is **x** odd? |  | |
|  | |  |  | | --- | --- | | [**ispolar**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#ispolar)(points, wrap=False) Check whether a polygon encloses a pole. |  | |
|  | |  |  | | --- | --- | | [**isscalar**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isscalar)(obj) Check for scalar types. |  | |
|  | |  |  | | --- | --- | | [**issequence**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#issequence)(obj, \*excluded) Check for sequence types. |  | |
|  | |  |  | | --- | --- | | [**isstr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#isstr)(obj) Check for string types. |  | |
|  | |  |  | | --- | --- | | [**issubclassof**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#issubclassof)(Sub, \*Supers) Check whether a class is a sub-class of some class(es). |  | |
|  | |  |  | | --- | --- | | [**iterNumpy2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#iterNumpy2)(obj) Iterate over Numpy2 wrappers or other sequences exceeding the threshold. |  | |
|  | |  |  | | --- | --- | | [**iterNumpy2over**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#iterNumpy2over)(n=None) Get or set the [iterNumpy2](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.iters-module.html#iterNumpy2) threshold. |  | |
|  | |  |  | | --- | --- | | [**joined**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#joined)(\*words, \*\*sep) DEPRECATED, use NN(...), NN.join\_ or **sep**.join. |  | |
|  | |  |  | | --- | --- | | [**joined\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#joined_)(\*words, \*\*sep) DEPRECATED, use \_SPACE\_(...), \_SPACE\_.join\_ or **sep**.join, sep=" ". |  | |
|  | |  |  | | --- | --- | | [**latDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#latDMS)(deg, form=**'**dms**'**, prec=2, sep='') Convert latitude to a string, optionally suffixed with N or S. |  | |
|  | |  |  | | --- | --- | | [**latlon2n\_xyz**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#latlon2n_xyz)(lat, lon, name='') Convert lat-, longitude to n-vector (normal to the earth's surface) X, Y and Z components. |  | |
|  | |  |  | | --- | --- | | [**latlonDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#latlonDMS)(lls, form=**'**dms**'**, prec=None, sep=None) Convert one or more LatLon instances to strings. |  | |
|  | |  |  | | --- | --- | | [**latlonDMS\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#latlonDMS_)(\*lls, \*\*form\_prec\_sep) Convert one or more LatLon instances to strings. |  | |
|  | |  |  | | --- | --- | | [**len2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#len2)(items) Make built-in function len work for generators, iterators, etc. |  | |
|  | |  |  | | --- | --- | | [**limiterrors**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#limiterrors)(raiser=None) Get/set the throwing of [LimitError](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.errors.LimitError-class.html)s. |  | |
|  | |  |  | | --- | --- | | [**lonDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#lonDMS)(deg, form=**'**dms**'**, prec=2, sep='') Convert longitude to a string, optionally suffixed with E or W. |  | |
|  | |  |  | | --- | --- | | [**luneOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#luneOf)(lon1, lon2, closed=False, LatLon=<class 'pygeodesy.points.LatLon\_'>, \*\*LatLon\_kwds) Generate an ellipsoidal or spherical [lune](https://wikipedia.org/wiki/Spherical_lune)-shaped path or polygon. |  | |
|  | |  |  | | --- | --- | | [**m2NM**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2NM)(meter) Convert meter to nautical miles (NM). |  | |
|  | |  |  | | --- | --- | | [**m2SM**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2SM)(meter) Convert meter to statute miles (SM). |  | |
|  | |  |  | | --- | --- | | [**m2chain**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2chain)(meter) Convert meter to *UK* chains. |  | |
|  | |  |  | | --- | --- | | [**m2degrees**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2degrees)(distance, radius=6371008.77141, lat=0) Convert a distance to an angle along the equator or along the parallel at an other (geodetic) latitude. |  | |
|  | |  |  | | --- | --- | | [**m2fathom**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2fathom)(meter) Convert meter to *UK* fathoms. |  | |
|  | |  |  | | --- | --- | | [**m2ft**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2ft)(meter, usurvey=False) Convert meter to *International* or *US Survey* feet (ft). |  | |
|  | |  |  | | --- | --- | | [**m2furlong**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2furlong)(meter) Convert meter to furlongs. |  | |
|  | |  |  | | --- | --- | | [**m2km**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2km)(meter) Convert meter to kilo meter (km). |  | |
|  | |  |  | | --- | --- | | [**m2radians**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2radians)(distance, radius=6371008.77141, lat=0) Convert a distance to an angle along the equator or along the parallel at an other (geodetic) latitude. |  | |
|  | |  |  | | --- | --- | | [**m2toise**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2toise)(meter) Convert meter to French [toises](https://wikipedia.org/wiki/Toise" \t "_top). |  | |
|  | |  |  | | --- | --- | | [**m2yard**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#m2yard)(meter) Convert meter to *UK* yards. |  | |
|  | |  |  | | --- | --- | | [**machine**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#machine)() Return standard platform.machine, but distinguishing Intel from Intel *emulation* on Apple Silicon (on macOS only). |  | |
|  | |  |  | | --- | --- | | [**map1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#map1)(fun1, \*xs) Apply each argument to a single-argument function and return a tuple of results. |  | |
|  | |  |  | | --- | --- | | [**map2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#map2)(func, \*xs) Apply arguments to a function and return a tuple of results. |  | |
|  | |  |  | | --- | --- | | [**meeus2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#meeus2)(point1, point2, point3, circum=False, useZ=True) Return the radius and *Meeus*' Type of the smallest circle *through* or *containing* three (2- or 3-D) points. |  | |
|  | |  |  | | --- | --- | | [**modulename**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#modulename)(clas, prefixed=None) Return the class name optionally prefixed with the module name. |  | |
|  | |  |  | | --- | --- | | [**n2e2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#n2e2)(n) Return e2, the *1st eccentricity squared* for a given *3rd flattening*. |  | |
|  | |  |  | | --- | --- | | [**n2f**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#n2f)(n) Return f, the *flattening* for a given *3rd flattening*. |  | |
|  | |  |  | | --- | --- | | [**n2f\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#n2f_)(n) Return f\_, the *inverse flattening* for a given *3rd flattening*. |  | |
|  | |  |  | | --- | --- | | [**n\_xyz2latlon**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#n_xyz2latlon)(x, y, z, name='') Convert n-vector components to lat- and longitude in degrees. |  | |
|  | |  |  | | --- | --- | | [**n\_xyz2philam**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#n_xyz2philam)(x, y, z, name='') Convert n-vector components to lat- and longitude in radians. |  | |
|  | |  |  | | --- | --- | | [**nameof**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nameof)(inst) Get the name of an instance. |  | |
|  | |  |  | | --- | --- | | [**nearestOn**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nearestOn)(point, point1, point2, within=True, useZ=True, Vector=None, \*\*Vector\_kwds) Locate the point between two points closest to a reference (2- or 3-D). |  | |
|  | |  |  | | --- | --- | | [**nearestOn3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nearestOn3)(point, points, closed=False, wrap=False, \*\*options) DEPRECATED, use function [pygeodesy.nearestOn5](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html#nearestOn5). |  | |
|  | |  |  | | --- | --- | | [**nearestOn4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nearestOn4)(point, points, closed=False, wrap=False, \*\*options) DEPRECATED, use function [pygeodesy.nearestOn5](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html#nearestOn5). |  | |
|  | |  |  | | --- | --- | | [**nearestOn5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nearestOn5)(point, points, closed=False, wrap=False, LatLon=None, \*\*options) Locate the point on a path or polygon closest to a reference point. |  | |
|  | |  |  | | --- | --- | | [**nearestOn6**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#nearestOn6)(point, points, closed=False, useZ=True, \*\*Vector\_and\_kwds) Locate the point on a path or polygon closest to a reference point. |  | |
|  | |  |  | | --- | --- | | [**neg**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#neg)(x) Negate x unless zero or NEG0. |  | |
|  | |  |  | | --- | --- | | [**neg\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#neg_)(\*xs) Negate all of xs with [neg](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.basics-module.html#neg). |  | |
|  | |  |  | | --- | --- | | [**norm2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#norm2)(x, y) Normalize a 2-dimensional vector. |  | |
|  | |  |  | | --- | --- | | [**normDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#normDMS)(strDMS, norm=None) Normalize all degree, minute and second DMS symbols in a string to the default DMS symbols '\xc2\xb0', '\xe2\x80\xb2' and '\xe2\x80\xb3'. |  | |
|  | |  |  | | --- | --- | | [**norm\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#norm_)(\*xs) Normalize all n-dimensional vector components. |  | |
|  | |  |  | | --- | --- | | [**notImplemented**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#notImplemented)(inst, \*args, \*\*kwds) Raise a NotImplementedError for a missing method or property. |  | |
|  | |  |  | | --- | --- | | [**notOverloaded**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#notOverloaded)(inst, \*args, \*\*kwds) Raise an AssertionError for a method or property not overloaded. |  | |
|  | |  |  | | --- | --- | | [**opposing**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#opposing)(bearing1, bearing2, margin=None) Compare the direction of two bearings given in degrees. |  | |
|  | |  |  | | --- | --- | | [**opposing\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#opposing_)(radians1, radians2, margin=None) Compare the direction of two bearings given in radians. |  | |
|  | |  |  | | --- | --- | | [**pairs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#pairs)(items, prec=6, fmt='F', ints=False, sep='=') Convert items to *name=value* strings, with floats handled like [fstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstr). |  | |
|  | |  |  | | --- | --- | | [**parse3d**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parse3d)(str3d, sep=',', Vector=<class 'pygeodesy.vector3d.Vector3d'>, \*\*Vector\_kwds) Parse an "x, y, z" string. |  | |
|  | |  |  | | --- | --- | | [**parse3llh**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parse3llh)(strllh, height=0, sep=',', clipLat=90, clipLon=180) Parse a string "lat lon [h]" representing lat-, longitude in degrees and optional height in meter. |  | |
|  | |  |  | | --- | --- | | [**parseDDDMMSS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseDDDMMSS)(strDDDMMSS, suffix=**'**NSEW**'**, sep='', clip=0, sexagecimal=False) Parse a lat- or longitude represention forms as [D]DDMMSS in degrees. |  | |
|  | |  |  | | --- | --- | | [**parseDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseDMS)(strDMS, suffix=**'**NSEW**'**, sep='', clip=0) Parse a lat- or longitude representation in degrees. |  | |
|  | |  |  | | --- | --- | | [**parseDMS2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseDMS2)(strLat, strLon, sep='', clipLat=90, clipLon=180) Parse a lat- and a longitude representions "lat, lon" in degrees. |  | |
|  | |  |  | | --- | --- | | [**parseETM5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseETM5)(strUTM, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Etm=<class 'pygeodesy.etm.Etm'>, falsed=True, name='') Parse a string representing a UTM coordinate, consisting of "zone[band] hemisphere easting northing". |  | |
|  | |  |  | | --- | --- | | [**parseMGRS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseMGRS)(strMGRS, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Mgrs=<class 'pygeodesy.mgrs.Mgrs'>, name='') Parse a string representing a MGRS grid reference, consisting of "zoneBand, grid, easting, northing". |  | |
|  | |  |  | | --- | --- | | [**parseOSGR**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseOSGR)(strOSGR, Osgr=<class 'pygeodesy.osgr.Osgr'>, name='') Parse a string representing an OSGR grid reference, consisting of "[grid] easting northing". |  | |
|  | |  |  | | --- | --- | | [**parseRad**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseRad)(strRad, suffix=**'**NSEW**'**, clip=0) Parse a string representing angle in radians. |  | |
|  | |  |  | | --- | --- | | [**parseUPS5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseUPS5)(strUPS, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Ups=<class 'pygeodesy.ups.Ups'>, falsed=True, name='') Parse a string representing a UPS coordinate, consisting of "[zone][band] pole easting northing" where **zone** is pseudo zone "00"|"0"|"" and band is 'A'|'B'|'Y'|'Z'|''. |  | |
|  | |  |  | | --- | --- | | [**parseUTM**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseUTM)(strUTM, datum=\_UTM, Utm=\_UTM, name='') DEPRECATED, use function [parseUTM5](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.utm-module.html#parseUTM5). |  | |
|  | |  |  | | --- | --- | | [**parseUTM5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseUTM5)(strUTM, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Utm=<class 'pygeodesy.utm.Utm'>, falsed=True, name='') Parse a string representing a UTM coordinate, consisting of "zone[band] hemisphere easting northing". |  | |
|  | |  |  | | --- | --- | | [**parseUTMUPS5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseUTMUPS5)(strUTMUPS, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Utm=<class 'pygeodesy.utm.Utm'>, Ups=<class 'pygeodesy.ups.Ups'>, name='') Parse a string representing a UTM or UPS coordinate, consisting of "zone[band] hemisphere/pole easting northing". |  | |
|  | |  |  | | --- | --- | | [**parseWM**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#parseWM)(strWM, radius=6378137.0, Wm=<class 'pygeodesy.webmercator.Wm'>, name='') Parse a string "e n [r]" representing a WM coordinate, consisting of easting, northing and an optional radius. |  | |
|  | |  |  | | --- | --- | | [**perimeterOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#perimeterOf)(points, closed=False, adjust=True, radius=6371008.77141, wrap=True) Approximate the perimeter of a path or polygon. |  | |
|  | |  |  | | --- | --- | | [**perimeterof**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#perimeterof)(points, closed=False, adjust=True, radius=6371008.77141, wrap=True) DEPRECATED, use function [perimeterOf](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.points-module.html" \l "perimeterOf). |  | |
|  | |  |  | | --- | --- | | [**philam2n\_xyz**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#philam2n_xyz)(phi, lam, name='') Convert lat-, longitude to n-vector (normal to the earth's surface) X, Y and Z components. |  | |
|  | |  |  | | --- | --- | | [**pierlot**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#pierlot)(point1, point2, point3, alpha12, alpha23, useZ=False, Clas=None, \*\*Clas\_kwds) 3-Point resection using [Pierlot](http://www.telecom.ulg.ac.be/publi/publications/pierlot/Pierlot2014ANewThree" \t "_top)'s method ToTal. |  | |
|  | |  |  | | --- | --- | | [**points2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#points2)(points, closed=True, base=None, Error=<class 'pygeodesy.errors.PointsError'>) Check a path or polygon represented by points. |  | |
|  | |  |  | | --- | --- | | [**polygon**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#polygon)(points, closed=True, base=None) DEPRECATED, use function [points2](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.iters-module.html#points2). |  | |
|  | |  |  | | --- | --- | | [**precision**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#precision)(form, prec=None) Set the default precison for a given F\_ form. |  | |
|  | |  |  | | --- | --- | | [**print\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#print_)(\*args, \*\*nl\_nt\_prefix\_end\_file\_flush\_sep) Python 3-style print function. |  | |
|  | |  |  | | --- | --- | | [**printf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#printf)(fmt, \*args, \*\*nl\_nt\_prefix\_end\_file\_flush\_sep) C-style printf function. |  | |
|  | |  |  | | --- | --- | | [**property\_doc\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#property_doc_)(doc) Decorator for a standard property with basic documentation. |  | |
|  | |  |  | | --- | --- | | [**quadOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#quadOf)(latS, lonW, latN, lonE, closed=False, LatLon=<class 'pygeodesy.points.LatLon\_'>, \*\*LatLon\_kwds) Generate a quadrilateral path or polygon from two points. |  | |
|  | |  |  | | --- | --- | | **radians**(x) Convert angle x from degrees to radians. |  | |
|  | |  |  | | --- | --- | | [**radians2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radians2m)(rad, radius=6371008.77141, lat=0) Convert an angle to a distance along the equator or along the parallel at an other (geodetic) latitude. |  | |
|  | |  |  | | --- | --- | | [**radiansPI**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radiansPI)(deg) Convert and wrap degrees to radians *[-PI..+PI]*. |  | |
|  | |  |  | | --- | --- | | [**radiansPI2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radiansPI2)(deg) Convert and wrap degrees to radians *[0..+2PI)*. |  | |
|  | |  |  | | --- | --- | | [**radiansPI\_2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radiansPI_2)(deg) Convert and wrap degrees to radians *[-3PI/2..+PI/2]*. |  | |
|  | |  |  | | --- | --- | | [**radical2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radical2)(distance, radius1, radius2) Compute the *radical ratio* and *radical line* of two [intersecting circles](https://mathworld.wolfram.com/Circle-CircleIntersection.html). |  | |
|  | |  |  | | --- | --- | | [**radii11**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#radii11)(point1, point2, point3, useZ=True) Return the radii of the In-, *Soddy* and Tangent circles of a (2- or 3-D) triangle. |  | |
|  | |  |  | | --- | --- | | [**randomrangenerator**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#randomrangenerator)(seed) Return a seeded random range function generator. |  | |
|  | |  |  | | --- | --- | | [**rangerrors**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#rangerrors)(raiser=None) Get/set the throwing of [RangeError](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.errors.RangeError-class.html)s. |  | |
|  | |  |  | | --- | --- | | [**reprs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#reprs)(objs, prec=6, fmt='F', ints=False) Convert objects to repr strings, with floats handled like [fstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstr). |  | |
|  | |  |  | | --- | --- | | [**scalar**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#scalar)(value, low=2.22044604925e-16, high=1.0, name=**'**scalar**'**, Error=<type 'exceptions.ValueError'>) DEPRECATED, use class [Number\_](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.units.Number_-class.html) or [Scalar\_](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.units.Scalar_-class.html). |  | |
|  | |  |  | | --- | --- | | [**signOf**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#signOf)(x) Return sign of x as int. |  | |
|  | |  |  | | --- | --- | | [**simplify1**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplify1)(points, distance=0.001, radius=6371008.77141, indices=False, \*\*options) Basic simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**simplify2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplify2)(points, pipe, radius=6371008.77141, shortest=False, indices=False, \*\*options) DEPRECATED, use function [pygeodesy.simplifyRW](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.simplify-module.html" \l "simplifyRW). |  | |
|  | |  |  | | --- | --- | | [**simplifyRDP**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplifyRDP)(points, distance=0.001, radius=6371008.77141, shortest=False, indices=False, \*\*options) Ramer-Douglas-Peucker (RDP) simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**simplifyRDPm**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplifyRDPm)(points, distance=0.001, radius=6371008.77141, shortest=False, indices=False, \*\*options) Modified Ramer-Douglas-Peucker (RDPm) simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**simplifyRW**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplifyRW)(points, pipe=0.001, radius=6371008.77141, shortest=False, indices=False, \*\*options) Reumann-Witkam (RW) simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**simplifyVW**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplifyVW)(points, area=0.001, radius=6371008.77141, attr=None, indices=False, \*\*options) Visvalingam-Whyatt (VW) simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**simplifyVWm**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#simplifyVWm)(points, area=0.001, radius=6371008.77141, attr=None, indices=False, \*\*options) Modified Visvalingam-Whyatt (VWm) simplification of a path of LatLon points. |  | |
|  | |  |  | | --- | --- | | [**sincos2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sincos2)(rad) Return the sine and cosine of an angle in radians. |  | |
|  | |  |  | | --- | --- | | [**sincos2\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sincos2_)(\*rads) Return the sine and cosine of angle(s) in {Cradians}. |  | |
|  | |  |  | | --- | --- | | [**sincos2d**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sincos2d)(deg) Return the sine and cosine of an angle in degrees. |  | |
|  | |  |  | | --- | --- | | [**sincos2d\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sincos2d_)(\*degs) Return the sine and cosine of angle(s) in degrees. |  | |
|  | |  |  | | --- | --- | | [**snellius3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#snellius3)(a, b, degC, alpha, beta) Snellius' surveying using [Snellius Pothenot](https://wikipedia.org/wiki/Snellius%E2%80%93Pothenot_problem" \t "_top). |  | |
|  | |  |  | | --- | --- | | [**soddy4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#soddy4)(point1, point2, point3, eps=8.881784197e-16, useZ=True) Return the radius and center of the inner *Soddy* circle of a (2- or 3-D) triangle. |  | |
|  | |  |  | | --- | --- | | [**splice**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#splice)(iterable, n=2, \*\*fill) Split an iterable into n slices. |  | |
|  | |  |  | | --- | --- | | [**sqrt0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sqrt0)(x2) Compute the square root iff **x2** > EPS02. |  | |
|  | |  |  | | --- | --- | | [**sqrt3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#sqrt3)(x2) Compute the square root, *cubed* *sqrt(x)\*\*3* or *sqrt(x\*\*3)*. |  | |
|  | |  |  | | --- | --- | | [**strs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#strs)(objs, prec=6, fmt='F', ints=False) Convert objects to str strings, with floats handled like [fstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "fstr). |  | |
|  | |  |  | | --- | --- | | [**tanPI\_2\_2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tanPI_2_2)(rad) Compute the tangent of half angle, 90 degrees rotated. |  | |
|  | |  |  | | --- | --- | | [**tan\_2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tan_2)(rad, \*\*semi) Compute the tangent of half angle. |  | |
|  | |  |  | | --- | --- | | [**tand**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tand)(deg, \*\*error\_kwds) Return the tangent of an angle in degrees. |  | |
|  | |  |  | | --- | --- | | [**tand\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tand_)(\*degs, \*\*error\_kwds) Return the tangent of angle(s) in degrees. |  | |
|  | |  |  | | --- | --- | | [**thomas**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#thomas)(lat1, lon1, lat2, lon2, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, wrap=False) Compute the distance between two (ellipsoidal) points using [Thomas'](https://apps.dtic.mil/dtic/tr/fulltext/u2/703541.pdf) formula. |  | |
|  | |  |  | | --- | --- | | [**thomas\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#thomas_)(phi2, phi1, lam21, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**) Compute the *angular* distance between two (ellipsoidal) points using [Thomas'](https://apps.dtic.mil/dtic/tr/fulltext/u2/703541.pdf) formula. |  | |
|  | |  |  | | --- | --- | | [**tienstra**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tienstra)(pointA, pointB, pointC, alpha, beta=None, gamma=None, useZ=False, Clas=None, \*\*Clas\_kwds) 3-Point resection using [Tienstra](https://wikipedia.org/wiki/Tienstra_formula" \t "_top)'s formula. |  | |
|  | |  |  | | --- | --- | | [**toCss**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toCss)(latlon, cs0=None, height=None, Css=<class 'pygeodesy.css.Css'>, name='') Convert an (ellipsoidal) geodetic point to a Cassini-Soldner location. |  | |
|  | |  |  | | --- | --- | | [**toDMS**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toDMS)(deg, form=**'**dms**'**, prec=2, sep='', ddd=2, neg='-', pos='') Convert *signed* degrees to string, without suffix. |  | |
|  | |  |  | | --- | --- | | [**toEtm8**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toEtm8)(latlon, lon=None, datum=None, Etm=<class 'pygeodesy.etm.Etm'>, falsed=True, name='', zone=None, \*\*cmoff) Convert a lat-/longitude point to an ETM coordinate. |  | |
|  | |  |  | | --- | --- | | [**toLcc**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toLcc)(latlon, conic=Conic(name='WRF\_Lb', lat0=40, lon0=-97, par1=33, par2=45, E0=0**...**, height=None, Lcc=<class 'pygeodesy.lcc.Lcc'>, name='', \*\*Lcc\_kwds) Convert an (ellipsoidal) geodetic point to a *Lambert* location. |  | |
|  | |  |  | | --- | --- | | [**toMgrs**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toMgrs)(utm, Mgrs=<class 'pygeodesy.mgrs.Mgrs'>, name='', \*\*Mgrs\_kwds) Convert a UTM coordinate to an MGRS grid reference. |  | |
|  | |  |  | | --- | --- | | [**toOsgr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toOsgr)(latlon, lon=None, datum=Datum(name='WGS84', ellipsoid=Ellipsoids.WGS84, transform=Tran**...**, Osgr=<class 'pygeodesy.osgr.Osgr'>, name='', \*\*Osgr\_kwds) Convert a lat-/longitude point to an OSGR coordinate. |  | |
|  | |  |  | | --- | --- | | [**toUps8**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toUps8)(latlon, lon=None, datum=None, Ups=<class 'pygeodesy.ups.Ups'>, pole='', falsed=True, strict=True, name='') Convert a lat-/longitude point to a UPS coordinate. |  | |
|  | |  |  | | --- | --- | | [**toUtm**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toUtm)(latlon, lon=None, datum=None, Utm=\_UTM, cmoff=True, name='') DEPRECATED, use function [pygeodesy.toUtm8](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.utm-module.html#toUtm8). |  | |
|  | |  |  | | --- | --- | | [**toUtm8**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toUtm8)(latlon, lon=None, datum=None, Utm=<class 'pygeodesy.utm.Utm'>, falsed=True, name='', zone=None, \*\*cmoff) Convert a lat-/longitude point to a UTM coordinate. |  | |
|  | |  |  | | --- | --- | | [**toUtmUps8**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toUtmUps8)(latlon, lon=None, datum=None, falsed=True, Utm=<class 'pygeodesy.utm.Utm'>, Ups=<class 'pygeodesy.ups.Ups'>, pole='', name='', \*\*cmoff) Convert a lat-/longitude point to a UTM or UPS coordinate. |  | |
|  | |  |  | | --- | --- | | [**toWm**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toWm)(latlon, lon=None, radius=6378137.0, Wm=<class 'pygeodesy.webmercator.Wm'>, name='', \*\*Wm\_kwds) Convert a lat-/longitude point to a WM coordinate. |  | |
|  | |  |  | | --- | --- | | [**toise2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#toise2m)(toises) Convert French [toises](https://wikipedia.org/wiki/Toise" \t "_top) to meter. |  | |
|  | |  |  | | --- | --- | | [**trfXform**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#trfXform)(reframe1, reframe2, epoch=None, xform=None, rates=None) Define a new Terrestrial Reference Frame (TRF) conversion. |  | |
|  | |  |  | | --- | --- | | [**triAngle**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#triAngle)(a, b, c) Compute one angle of a triangle. |  | |
|  | |  |  | | --- | --- | | [**triAngle4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#triAngle4)(a, b, c) Compute the angles of a triangle. |  | |
|  | |  |  | | --- | --- | | [**triSide**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#triSide)(a, b, radC) Compute one side of a triangle. |  | |
|  | |  |  | | --- | --- | | [**triSide2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#triSide2)(b, c, radB) Compute one side and the opposite angle of a triangle. |  | |
|  | |  |  | | --- | --- | | [**triSide4**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#triSide4)(radA, radB, c) Compute two sides and the height of a triangle. |  | |
|  | |  |  | | --- | --- | | [**trilaterate2d2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#trilaterate2d2)(x1, y1, radius1, x2, y2, radius2, x3, y3, radius3, eps=None, \*\*Vector\_and\_kwds) Trilaterate three circles, each given as a (2-D) center and a radius. |  | |
|  | |  |  | | --- | --- | | [**trilaterate3d2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#trilaterate3d2)(center1, radius1, center2, radius2, center3, radius3, eps=2.22044604925e-16, \*\*Vector\_and\_kwds) Trilaterate three spheres, each given as a (3-D) center and a radius. |  | |
|  | |  |  | | --- | --- | | [**tyr3d**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#tyr3d)(tilt=0, yaw=0, roll=0, Vector=<class 'pygeodesy.vector3d.Vector3d'>, \*\*Vector\_kwds) Convert an attitude oriention into a (3-D) direction vector. |  | |
|  | |  |  | | --- | --- | | **ub2str**(ub) Convert unicode or bytes to str. |  | |
|  | |  |  | | --- | --- | | [**unStr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unStr)(name, \*args, \*\*kwds) DEPRECATED, use function [pygeodesy.unstr](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.streprs-module.html" \l "unstr). |  | |
|  | |  |  | | --- | --- | | [**unroll180**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unroll180)(lon1, lon2, wrap=True) Unroll longitudinal delta and wrap longitude in degrees. |  | |
|  | |  |  | | --- | --- | | [**unrollPI**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unrollPI)(rad1, rad2, wrap=True) Unroll longitudinal delta and wrap longitude in radians. |  | |
|  | |  |  | | --- | --- | | [**unsign0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unsign0)(x) DEPRECATED, use function [pygeodesy.unsigned0](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.basics-module.html#unsigned0). |  | |
|  | |  |  | | --- | --- | | [**unsigned0**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unsigned0)(x) Return 0.0 unsigned. |  | |
|  | |  |  | | --- | --- | | [**unstr**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#unstr)(named, \*args, \*\*kwds) Return the string representation of an invokation. |  | |
|  | |  |  | | --- | --- | | [**upsZoneBand5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#upsZoneBand5)(lat, lon, strict=True, name='') Return the UTM/UPS zone number, *polar* Band letter, pole and clipped lat- and longitude for a given location. |  | |
|  | |  |  | | --- | --- | | [**utmZoneBand2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#utmZoneBand2)(lat, lon) DEPRECATED, use function [pygeodesy.utmZoneBand5](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy.utm-module.html#utmZoneBand5). |  | |
|  | |  |  | | --- | --- | | [**utmZoneBand5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#utmZoneBand5)(lat, lon, cmoff=False, name='') Return the UTM zone number, Band letter, hemisphere and (clipped) lat- and longitude for a given location. |  | |
|  | |  |  | | --- | --- | | [**utmupsValidate**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#utmupsValidate)(coord, falsed=False, MGRS=False, Error=<class 'pygeodesy.utmups.UTMUPSError'>) Check a UTM or UPS coordinate. |  | |
|  | |  |  | | --- | --- | | [**utmupsValidateOK**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#utmupsValidateOK)(coord, falsed=False, ok=True) Check a UTM or UPS coordinate. |  | |
|  | |  |  | | --- | --- | | [**utmupsZoneBand5**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#utmupsZoneBand5)(lat, lon, cmoff=False, name='') Return the UTM/UPS zone number, Band letter, hemisphere/pole and clipped lat- and longitude for a given location. |  | |
|  | |  |  | | --- | --- | | [**vincentys**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#vincentys)(lat1, lon1, lat2, lon2, radius=6371008.77141, wrap=False) Compute the distance between two (spherical) points using [Vincenty's](https://wikipedia.org/wiki/Great-circle_distance" \t "_top) spherical formula. |  | |
|  | |  |  | | --- | --- | | [**vincentys\_**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#vincentys_)(phi2, phi1, lam21) Compute the *angular* distance between two (spherical) points using [Vincenty's](https://wikipedia.org/wiki/Great-circle_distance" \t "_top) spherical formula. |  | |
|  | |  |  | | --- | --- | | [**wildberger3**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wildberger3)(a, b, c, alpha, beta, R3=<built-in function min>) Snellius' surveying using [Rational Trigonometry](https://wikipedia.org/wiki/Snellius%E2%80%93Pothenot_problem). |  | |
|  | |  |  | | --- | --- | | [**wrap180**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrap180)(deg) Wrap degrees to *[-180..+180]*. |  | |
|  | |  |  | | --- | --- | | [**wrap360**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrap360)(deg) Wrap degrees to *[0..+360)*. |  | |
|  | |  |  | | --- | --- | | [**wrap90**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrap90)(deg) Wrap degrees to *[-270..+90]*. |  | |
|  | |  |  | | --- | --- | | [**wrapPI**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrapPI)(rad) Wrap radians to *[-PI..+PI]*. |  | |
|  | |  |  | | --- | --- | | [**wrapPI2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrapPI2)(rad) Wrap radians to *[0..+2PI)*. |  | |
|  | |  |  | | --- | --- | | [**wrapPI\_2**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#wrapPI_2)(rad) Wrap radians to *[-3PI/2..+PI/2]*. |  | |
|  | |  | | --- | | [**yard2m**](https://mrjean1.github.io/PyGeodesy/docs/pygeodesy-module.html#yard2m)(yards) Convert *UK* yards to meter. | |