GeoPy Documentation

Release 1.10.0

GeoPy Contributors

April 05, 2015

Contents

geopy is a Python 2 and 3 client for several popular geocoding web services.

geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

geopy is tested against CPython 2.7, CPython 3.2, CPython 3.4, PyPy, and PyPy3.

CHAPTER 1

Geocoders

Each geolocation service you might use, such as Google Maps, Bing Maps, or Yahoo BOSS, has its own class in `geopy.geocoders` abstracting the service's API. Geocoders each define at least a `geocode` method, for resolving a location from a string, and may define a `reverse` method, which resolves a pair of coordinates to an address. Each Geocoder accepts any credentials or settings needed to interact with its service, e.g., an API key or locale, during its initialization.

To geolocate a query to an address and coordinates:

```
>>> from geopy.geocoders import Nominatim
>>> geolocator = Nominatim()
>>> location = geolocator.geocode("175 5th Avenue NYC")
>>> print(location.address)
Flatiron Building, 175, 5th Avenue, Flatiron, New York, NYC, New York, ...
>>> print((location.latitude, location.longitude))
(40.7410861, -73.9896297241625)
>>> print(location.raw)
{'place_id': '9167009604', 'type': 'attraction', ...}
```

To find the address corresponding to a set of coordinates:

```
>>> from geopy.geocoders import Nominatim
>>> geolocator = Nominatim()
>>> location = geolocator.reverse("52.509669, 13.376294")
>>> print(location.address)
Potsdamer Platz, Mitte, Berlin, 10117, Deutschland, European Union
>>> print((location.latitude, location.longitude))
(52.5094982, 13.3765983)
>>> print(location.raw)
{'place_id': '654513', 'osm_type': 'node', ...}
```

Locators' `geolocate` and `reverse` methods require the argument `query`, and also accept at least the argument `exactly_one`, which is `True`. Geocoders may have additional attributes, e.g., Bing accepts `user_location`, the effect of which is to bias results near that location. `geolocate` and `reverse` methods may return three types of values:

- When there are no results found, returns `None`.
- When the method's `exactly_one` argument is `True` and at least one result is found, returns a `geopy.location.Location` object, which can be iterated over as:

  (address<String>, (latitude<Float>, longitude<Float>))

  Or can be accessed as *Location.address*, *Location.latitude*, *Location.longitude*, *Location.altitude*, and *Location.raw*. The last contains the geocoder's unparsed response for this result.

- When `exactly_one` is False, and there is at least one result, returns a list of `geopy.location.Location` objects, as above:

  [Location, [...]]

If a service is unavailable or otherwise returns a non-OK response, or doesn't receive a response in the allotted timeout, you will receive one of the Exceptions detailed below.

Every geocoder accepts an argument `format_string` that defaults to '%s' where the input string to geocode is interpolated. For example, if you only need to geocode locations in Cleveland, Ohio, you could do:

```
>>> from geopy.geocoders import GeocoderDotUS
>>> geolocator = GeocoderDotUS(format_string="%s, Cleveland OH")
>>> address, (latitude, longitude) = geolocator.geocode("11111 Euclid Ave")
>>> print(address, latitude, longitude)
11111 Euclid Ave, Cleveland, OH 44106 41.506784 -81.608148
```

geopy.geocoders.**get_geocoder_for_service**(*service*)

For the service provided, try to return a geocoder class.

```
>>> from geopy.geocoders import get_geocoder_for_service
>>> get_geocoder_for_service("nominatim")
geopy.geocoders.osm.Nominatim
```

If the string given is not recognized, a `geopy.exc.GeocoderNotFound` exception is raised.

class geopy.geocoders.**ArcGIS**(*username=None*, *password=None*, *referer=None*, *token_lifetime=60*, *scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using the ERSI ArcGIS API. Documentation at: https://developers.arcgis.com/rest/geocode/api-reference/overview-world-geocoding-service.htm

__init__(*username=None*, *password=None*, *referer=None*, *token_lifetime=60*, *scheme='https'*, *timeout=1*, *proxies=None*)

Create a ArcGIS-based geocoder.

New in version 0.97.

Parameters

- **username** (*string*) – ArcGIS username. Required if authenticated mode is desired.
- **password** (*string*) – ArcGIS password. Required if authenticated mode is desired.
- **referer** (*string*) – Required if authenticated mode is desired. 'Referer' HTTP header to send with each request, e.g., 'http://www.example.com'. This is tied to an issued token, so fielding queries for multiple referrers should be handled by having multiple ArcGIS geocoder instances.
- **token_lifetime** (*int*) – Desired lifetime, in minutes, of an ArcGIS-issued token.
- **scheme** (*string*) – Desired scheme. If authenticated mode is in use, it must be 'https'.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *exactly_one=True*, *timeout=None*, *distance=None*, *wkid=4326*)

Given a point, find an address.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s".) – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*bool*) – Return one result, or a list?
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.
- **distance** (*int*) – Distance from the query location, in meters, within which to search. ArcGIS has a default of 100 meters, if not specified.
- **wkid** (*string*) – WKID to use for both input and output coordinates.

class `geopy.geocoders.`**Baidu**(*api_key*, *scheme='http'*, *timeout=1*, *proxies=None*)

Geocoder using the Baidu Maps v2 API. Documentation at: http://developer.baidu.com/map/webservice-geocoding.htm

__init__(*api_key*, *scheme='http'*, *timeout=1*, *proxies=None*)

Initialize a customized Baidu geocoder using the v2 API.

New in version 1.0.0.

Parameters

- **api_key** (*string*) – The API key required by Baidu Map to perform geocoding requests. API keys are managed through the Baidu APIs console (http://lbsyun.baidu.com/apiconsole/key).
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is http and only http support.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *timeout=None*)
Given a point, find an address.

Parameters

- **query** (geopy.point.Point, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class geopy.geocoders.**Bing**(*api_key*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using the Bing Maps Locations API. Documentation at: https://msdn.microsoft.com/en-us/library/ff701715.aspx

__init__(*api_key*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)
Initialize a customized Bing geocoder with location-specific address information and your Bing Maps API key.

Parameters

- **api_key** (*string*) – Should be a valid Bing Maps API key.
- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, Mountain View, CA'. The default is just '%s'.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.

  New in version 0.97.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception.

  New in version 0.97.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on urllib2.ProxyHandler.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *user_location=None*, *timeout=None*, *culture=None*, *include_neighborhood=None*, *include_country_code=False*)
Geocode an address.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **user_location** (geopy.point.Point) – Prioritize results closer to this location.

  New in version 0.96.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

- **culture** (*string*) – Affects the language of the response, must be a two-letter country code.

  New in version 1.4.0.

- **include_neighborhood** (*boolean*) – Sets whether to include the neighborhood field in the response.

  New in version 1.4.0.

- **include_country_code** (*boolean*) – Sets whether to include the two-letter ISO code of the country in the response (field name 'countryRegionIso2').

  New in version 1.4.0.

reverse(*query*, *exactly_one=True*, *timeout=None*)

Reverse geocode a point.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s".) – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*bool*) – Return one result, or a list?
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

class `geopy.geocoders.`**DataBC**(*scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using the Physical Address Geocoder from DataBC. Documentation at:
http://www.data.gov.bc.ca/dbc/geographic/locate/geocoding.page

__init__(*scheme='https'*, *timeout=1*, *proxies=None*)

Create a DataBC-based geocoder.

Parameters

- **scheme** (*string*) – Desired scheme.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *max_results=25*, *set_back=0*, *location_descriptor='any'*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **max_results** (*int*) – The maximum number of resutls to request.
- **set_back** (*float*) – The distance to move the accessPoint away from the curb (in meters) and towards the interior of the parcel. location_descriptor must be set to accessPoint for set_back to take effect.
- **location_descriptor** (*string*) – The type of point requested. It can be any, accessPoint, frontDoorPoint, parcelPoint, rooftopPoint and routingPoint.

- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**GeocodeFarm**(*api_key=None*, *format_string='%s'*, *timeout=1*, *proxies=None*)

Geocoder using the GeocodeFarm API. Documentation at: https://www.geocode.farm/geocoding/free-api-documentation/

__init__(*api_key=None*, *format_string='%s'*, *timeout=1*, *proxies=None*)

Create a geocoder for GeocodeFarm.

New in version 0.99.

Parameters

- **api_key** (*string*) – The API key required by GeocodeFarm to perform geocoding requests.
- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, Mountain View, CA'. The default is just '%s'.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *exactly_one=True*, *timeout=None*)

Returns a reverse geocoded location.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*bool*) – Return one result or a list of results, if available. GeocodeFarm's API will always return at most one result.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**GeocoderDotUS**(*username=None*, *password=None*, *format_string='%s'*, *timeout=1*, *proxies=None*)

GeocoderDotUS geocoder, documentation at: http://geocoder.us/

Note that GeocoderDotUS does not support SSL.

__init__(*username=None*, *password=None*, *format_string='%s'*, *timeout=1*, *proxies=None*)

Parameters

- **username** (*string*) –
- **password** (*string*) –
- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, Mountain View, CA'. The default is just '%s'.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising an `geopy.exc.GeocoderTimedOut` exception.

  New in version 0.97.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

class `geopy.geocoders.`**GeoNames**(*country_bias=None*, *username=None*, *timeout=1*, *proxies=None*)

GeoNames geocoder, documentation at: http://www.geonames.org/export/geonames-search.html

Reverse geocoding documentation at: http://www.geonames.org/maps/us-reverse-geocoder.html

__init__(*country_bias=None*, *username=None*, *timeout=1*, *proxies=None*)

Parameters

- **country_bias** (*string*) –
- **username** (*string*) –
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.

  New in version 0.97.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

class `geopy.geocoders.`**GoogleV3**(*api_key=None*, *domain='maps.googleapis.com'*, *scheme='https'*, *client_id=None*, *secret_key=None*, *timeout=1*, *proxies=None*)

Geocoder using the Google Maps v3 API. Documentation at: https://developers.google.com/maps/documentation/geocoding/

__init__(*api_key=None*, *domain='maps.googleapis.com'*, *scheme='https'*, *client_id=None*, *secret_key=None*, *timeout=1*, *proxies=None*)

Initialize a customized Google geocoder.

API authentication is only required for Google Maps Premier customers.

Parameters

- **api_key** (*string*) – The API key required by Google to perform geocoding requests. API keys are managed through the Google APIs console (https://code.google.com/apis/console).

  New in version 0.98.2.
- **domain** (*string*) – Should be the localized Google Maps domain to connect to. The default is 'maps.google.com', but if you're geocoding address in the UK (for example), you may want to set it to 'maps.google.co.uk' to properly bias results.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.

  New in version 0.97.
- **client_id** (*string*) – If using premier, the account client id.
- **secret_key** (*string*) – If using premier, the account secret key.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *timeout=None*, *bounds=None*, *region=None*, *components=None*, *language=None*, *sensor=False*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.
- **bounds** (*list or tuple*) – The bounding box of the viewport within which to bias geocode results more prominently.

- **region** (*string*) – The region code, specified as a ccTLD ("top-level domain") two-character value.
- **components** (*dict*) – Restricts to an area. Can use any combination of: route, locality, administrative_area, postal_code, country.

  New in version 0.97.1.
- **language** (*string*) – The language in which to return results.
- **sensor** (*bool*) – Whether the geocoding request comes from a device with a location sensor.

reverse(*query*, *exactly_one=False*, *timeout=None*, *language=None*, *sensor=False*)

Given a point, find an address.

Parameters

- **query** (geopy.point.Point, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*boolean*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception.

  New in version 0.97.
- **language** (*string*) – The language in which to return results.
- **sensor** (*boolean*) – Whether the geocoding request comes from a device with a location sensor.

timezone(*location*, *at_time=None*, *timeout=None*)

This is an unstable API.

Finds the timezone a *location* was in for a specified *at_time*, and returns a pytz timezone object.

New in version 1.2.0.

Parameters

- **location** (geopy.point.Point, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you want a timezone.
- **at_time** – The time at which you want the timezone of this location. This is optional, and defaults to the time that the function is called in UTC.

Return type pytz timezone

class geopy.geocoders.**IGNFrance**(*api_key*, *username=None*, *password=None*, *referer=None*, *domain='wxs.ign.fr'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using the IGN France GeoCoder OpenLS API. Documentation at: http://api.ign.fr/tech-docs-js/fr/developpeur/search.html

__init__(*api_key*, *username=None*, *password=None*, *referer=None*, *domain='wxs.ign.fr'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Initialize a customized IGN France geocoder.

Parameters

- **api_key** (*string*) – The API key required by IGN France API to perform geocoding requests. You can get your key here: http://api.ign.fr. Mandatory. For authentication with referer and with username/password, the api key always differ.

- **username** (*string*) – When making a call need HTTP simple authentication username. Mandatory if no referer set
- **password** (*string*) – When making a call need HTTP simple authentication password. Mandatory if no referer set
- **referer** (*string*) – When making a call need HTTP referer. Mandatory if no password and username
- **domain** (*string*) – Currently it is 'wxs.ign.fr', can be changed for testing purposes for developer API e.g gpp3-wxs.ign.fr at the moment.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *query_type='StreetAddress'*, *maximum_responses=25*, *is_freeform=False*, *filtering=None*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The query string to be geocoded.
- **query_type** (*string*) – The type to provide for geocoding. It can be PositionOfInterest, StreetAddress or CadastralParcel. StreetAddress is the default choice if none provided.
- **maximum_responses** (*int*) – The maximum number of responses to ask to the API in the query body.
- **is_freeform** (*string*) – Set if return is structured with freeform structure or a more structured returned. By default, value is False.
- **filtering** (*string*) – Provide string that help setting geocoder filter. It contains an XML string. See examples in documentation and ignfrance.py file in directory tests.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *reverse_geocode_preference=('StreetAddress',)*, *maximum_responses=25*, *filtering=''*, *exactly_one=False*, *timeout=None*)

Given a point, find an address.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **reverse_geocode_preference** (*list*) – Enable to set expected results type. It can be StreetAddress or PositionOfInterest. Default is set to StreetAddress
- **maximum_responses** (*int*) – The maximum number of responses to ask to the API in the query body.

- **filtering** (*string*) – Provide string that help setting geocoder filter. It contains an XML string. See examples in documentation and ignfrance.py file in directory tests.
- **exactly_one** (*boolean*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**LiveAddress**(*auth_id*, *auth_token*, *candidates=1*, *scheme='https'*, *timeout=1*, *proxies=None*)

Initialize a customized LiveAddress geocoder provided by SmartyStreets. More information regarding the LiveAddress API can be found here:

https://smartystreets.com/products/liveaddress-api

__init__(*auth_id*, *auth_token*, *candidates=1*, *scheme='https'*, *timeout=1*, *proxies=None*)

Initialize a customized SmartyStreets LiveAddress geocoder.

Parameters

- **auth_id** (*string*) – Valid *Auth ID* from SmartyStreets.

  New in version 1.5.0.
- **auth_token** (*string*) – Valid *Auth Token* from SmartyStreets.
- **candidates** (*int*) – An integer between 1 and 10 indicating the max number of candidate addresses to return if a valid address could be found.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.

  New in version 0.97.

  Changed in version 1.8.0.

  LiveAddress now requires *https*. Specifying *scheme=http* will result in a `geopy.exc.ConfigurationError`.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising an `geopy.exc.GeocoderTimedOut` exception.

  New in version 0.97.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.

class `geopy.geocoders.`**NaviData**(*api_key=None*, *domain='api.navidata.pl'*, *timeout=1*, *proxies=None*)

Geocoder using the NaviData API. Documentation at:

http://www.navidata.pl

__init__(*api_key=None*, *domain='api.navidata.pl'*, *timeout=1*, *proxies=None*)

New in version 1.8.0.

Initialize NaviData geocoder. Please note that 'scheme' parameter is not supported: at present state, all NaviData traffic use plain http.

Parameters

- **api_key** (*string*) – The commercial API key for service. None required if you use the API for non-commercial purposes.
- **domain** (*string*) – Currently it is 'api.navidata.pl', can be changed for testing purposes.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The query string to be geocoded; this must be URL encoded.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *exactly_one=True*, *timeout=None*)

Given a point, find an address.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*boolean*) – Return one result or a list of results, if available. Currently this has no effect (only one address is returned by API).
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**Nominatim**(*format_string='%s'*, *view_box=(-180, -90, 180, 90)*, *country_bias=None*, *timeout=1*, *proxies=None*, *domain='nominatim.openstreetmap.org'*, *scheme='https'*)

Nominatim geocoder for OpenStreetMap servers. Documentation at: https://wiki.openstreetmap.org/wiki/Nominatim

Note that Nominatim does not support SSL.

__init__(*format_string='%s'*, *view_box=(-180, -90, 180, 90)*, *country_bias=None*, *timeout=1*, *proxies=None*, *domain='nominatim.openstreetmap.org'*, *scheme='https'*)

Parameters

- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, Mountain View, CA'. The default is just '%s'.
- **view_box** (*tuple*) – Coordinates to restrict search within.
- **country_bias** (*string*) – Bias results to this country.

- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

- **domain** (*string*) – Should be the localized Openstreetmap domain to connect to. The default is 'nominatim.openstreetmap.org', but you can change it to a domain of your own.

  New in version 1.8.2.

- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.

  New in version 1.8.2.

geocode(*query*, *exactly_one=True*, *timeout=None*, *addressdetails=False*, *language=False*, *geometry=None*)

Geocode a location query.

Parameters

- **query** – The address, query or structured query to geocode you wish to geocode.

  For a structured query, provide a dictionary whose keys are one of: *street*, *city*, *county*, *state*, *country*, or *postalcode*. For more information, see Nominatim's documentation for "structured requests":

  https://wiki.openstreetmap.org/wiki/Nominatim

- **exactly_one** (*bool*) – Return one result or a list of results, if available.

- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

- **addressdetails** (*bool*) – If you want in *Location.raw* to include addressdetails such as city_district, etc set it to True

- **language** (*string*) – Preferred language in which to return results. Either uses standard RFC2616 accept-language string or a simple comma-separated list of language codes.

- **geometry** (*string*) – If present, specifies whether the geocoding service should return the result's geometry in *wkt*, *svg*, *kml*, or *geojson* formats. This is available via the *raw* attribute on the returned `geopy.location.Location` object.

  New in version 1.3.0.

reverse(*query*, *exactly_one=True*, *timeout=None*, *language=False*)

Returns a reverse geocoded location.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.

- **exactly_one** (*bool*) – Return one result or a list of results, if available.

- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

- **language** (*string*) – Preferred language in which to return results. Either uses standard RFC2616 accept-language string or a simple comma-separated list of language codes.

class geopy.geocoders.**OpenCage**(*api_key*, *domain='api.opencagedata.com'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using the Open Cage Data API. Documentation at: http://geocoder.opencagedata.com/api.html

..versionadded:: 1.1.0

__init__(*api_key*, *domain='api.opencagedata.com'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Initialize a customized Open Cage Data geocoder.

Parameters

- **api_key** (*string*) – The API key required by Open Cage Data to perform geocoding requests. You can get your key here: https://developer.opencagedata.com/
- **domain** (*string*) – Currently it is 'api.opencagedata.com', can be changed for testing purposes.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on urllib2.ProxyHandler.

geocode(*query*, *bounds=None*, *country=None*, *language=None*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The query string to be geocoded; this must be URL encoded.
- **language** (*string*) – an IETF format language code (such as *es* for Spanish or pt-BR for Brazilian Portuguese); if this is omitted a code of *en* (English) will be assumed by the remote service.
- **bounds** (*string*) – Provides the geocoder with a hint to the region that the query resides in. This value will help the geocoder but will not restrict the possible results to the supplied region. The bounds parameter should be specified as 4 coordinate points forming the south-west and north-east corners of a bounding box. For example, *bounds=-0.563160,51.280430,0.278970,51.683979*.
- **country** (*string*) – Provides the geocoder with a hint to the country that the query resides in. This value will help the geocoder but will not restrict the possible results to the supplied country. The country code is a 3 character code as defined by the ISO 3166-1 Alpha 3 standard.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *language=None*, *exactly_one=False*, *timeout=None*)

Given a point, find an address.

Parameters

- **query** (geopy.point.Point, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.

- **language** (*string*) – The language in which to return results.
- **exactly_one** (*boolean*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**`OpenMapQuest`**(*api_key=None*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Geocoder using MapQuest Open Platform Web Services. Documentation at:
http://developer.mapquest.com/web/products/open/geocoding-service

`__init__`(*api_key=None*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)

Initialize an Open MapQuest geocoder with location-specific address information. No API Key is needed by the Nominatim based platform.

Parameters

- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, Mountain View, CA'. The default is just '%s'.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.

  New in version 0.97.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.

  New in version 0.97.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

`geocode`(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

  New in version 0.97.

class `geopy.geocoders.`**`YahooPlaceFinder`**(*consumer_key*, *consumer_secret*, *timeout=1*, *proxies=None*)

Geocoder that utilizes the Yahoo! BOSS PlaceFinder API. Documentation at:
https://developer.yahoo.com/boss/geo/docs/

`__init__`(*consumer_key*, *consumer_secret*, *timeout=1*, *proxies=None*)

Parameters

- **consumer_key** (*string*) – Key provided by Yahoo.

- **consumer_secret** (*string*) – Secret corresponding to the key provided by Yahoo.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.
- **proxies** (*dict*) – If specified, routes this geocoder"s requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

  New in version 0.96.

geocode(*query*, *exactly_one=True*, *timeout=None*, *min_quality=0*, *reverse=False*, *valid_country_codes=None*, *with_timezone=False*)
Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **min_quality** (*int*) –
- **reverse** (*bool*) –
- **valid_country_codes** (*list or tuple*) –
- **with_timezone** (*bool*) – Include the timezone in the response's *raw* dictionary (as *timezone*).

class `geopy.geocoders.`**What3Words**(*api_key*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)

What3Words geocoder, documentation at: http://what3words.com/api/reference

__init__(*api_key*, *format_string='%s'*, *scheme='https'*, *timeout=1*, *proxies=None*)
Initialize a What3Words geocoder with 3-word or OneWord-address and What3Words API key.

New in version 1.5.0.

Parameters

- **api_key** (*string*) – Key provided by What3Words.
- **format_string** (*string*) – String containing '%s' where the string to geocode should be interpolated before querying the geocoder. For example: '%s, piped.gains.jungle'. The default is just '%s'.
- **scheme** (*string*) – Use 'https' or 'http' as the API URL's scheme. Default is https. Note that SSL connections' certificates are not verified.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *lang='en'*, *exactly_one=True*, *timeout=None*)
Geocode a "3 words" or "OneWord" query.

Parameters

- **query** (*string*) – The 3-word or OneWord-address you wish to geocode.

- **lang** (*string*) – two character language codes as supported by the API (http://what3words.com/api/reference/languages).
- **exactly_one** (*bool*) – Parameter has no effect for this geocoder. Due to the address scheme there is always exactly one result.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization. .. versionadded:: 0.97

reverse(*query*, *lang='en'*, *exactly_one=True*, *timeout=None*)

Given a point, find the 3 word address.

Parameters

- **query** (`geopy.point.Point`, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the 3 word address.
- **lang** (*string*) – two character language codes as supported by the API (http://what3words.com/api/reference/languages).
- **exactly_one** (*bool*) – Parameter has no effect for this geocoder. Due to the address scheme there is always exactly one result.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

class `geopy.geocoders.`**Yandex**(*api_key=None*, *lang=None*, *timeout=1*, *proxies=None*)

Yandex geocoder, documentation at: http://api.yandex.com/maps/doc/geocoder/desc/concepts/input_params.xml

__init__(*api_key=None*, *lang=None*, *timeout=1*, *proxies=None*)

Create a Yandex-based geocoder.

New in version 1.5.0.

Parameters

- **api_key** (*string*) – Yandex API key (not obligatory) http://api.yandex.ru/maps/form.xml
- **lang** (*string*) – response locale, the following locales are supported: "ru_RU" (default), "uk_UA", "be_BY", "en_US", "tr_TR"
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception.
- **proxies** (*dict*) – If specified, routes this geocoder's requests through the specified proxy. E.g., {"https": "192.0.2.0"}. For more information, see documentation on `urllib2.ProxyHandler`.

geocode(*query*, *exactly_one=True*, *timeout=None*)

Geocode a location query.

Parameters

- **query** (*string*) – The address or query you wish to geocode.
- **exactly_one** (*bool*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a `geopy.exc.GeocoderTimedOut` exception. Set this only if you wish to override, on this call only, the value set during the geocoder's initialization.

reverse(*query*, *exactly_one=False*, *timeout=None*)
Given a point, find an address.

Parameters

- **query** (geopy.point.Point, list or tuple of (latitude, longitude), or string as "%(latitude)s, %(longitude)s") – The coordinates for which you wish to obtain the closest human-readable addresses.
- **exactly_one** (*boolean*) – Return one result or a list of results, if available.
- **timeout** (*int*) – Time, in seconds, to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception.

CHAPTER 2

Calculating Distance

New in version 0.93.

Geopy can calculate geodesic distance between two points using the [Vincenty distance](https://en.wikipedia.org/wiki/Vincenty's_formulae) or [great-circle distance](https://en.wikipedia.org/wiki/Great-circle_distance) formulas, with a default of Vincenty available as the function *geopy.distance.distance*.

Great-circle distance (`great_circle`) uses a spherical model of the earth, using the average great-circle radius of 6372.795 kilometers, resulting in an error of up to about 0.5%. The radius value is stored in `distance.EARTH_RADIUS`, so it can be customized (it should always be in kilometers, however).

Vincenty distance (`vincenty`) uses a more accurate ellipsoidal model of the earth. This is the default distance formula, and is thus aliased as `distance.distance`. There are multiple popular ellipsoidal models, and which one will be the most accurate depends on where your points are located on the earth. The default is the WGS-84 ellipsoid, which is the most globally accurate. geopy includes a few other models in the distance.ELLIPSOIDS dictionary:

```
              model             major (km)   minor (km)     flattening
ELLIPSOIDS = {'WGS-84':        (6378.137,    6356.7523142,  1 /
              'GRS-80':        (6378.137,    6356.7523141,  1 /
              'Airy (1830)':   (6377.563396, 6356.256909,   1 /
              'Intl 1924':     (6378.388,    6356.911946,   1 / 297.0),
              'Clarke (1880)': (6378.249145, 6356.51486955, 1 / 293.465),
              'GRS-67':        (6378.1600,   6356.774719,   1 / 298.25),
              }
```

Here's an example usage of distance.vincenty:

```
>>> from geopy.distance import vincenty
>>> newport_ri = (41.49008, -71.312796)
>>> cleveland_oh = (41.499498, -81.695391)
>>> print(vincenty(newport_ri, cleveland_oh).miles)
538.3904451566326
```

Using great-circle distance:

```
>>> from geopy.distance import great_circle
>>> newport_ri = (41.49008, -71.312796)
>>> cleveland_oh = (41.499498, -81.695391)
>>> print(great_circle(newport_ri, cleveland_oh).miles)
537.1485284062816
```

You can change the ellipsoid model used by the Vincenty formula like so:

```
>>> distance.vincenty(ne, cl, ellipsoid='GRS-80').miles
```

The above model name will automatically be retrieved from the ELLIPSOIDS dictionary. Alternatively, you can specify the model values directly:

```
>>> distance.vincenty(ne, cl, ellipsoid=(6377., 6356., 1 / 297.)).miles
```

Distances support simple arithmetic, making it easy to do things like calculate the length of a path:

```
>>> d = distance.distance
>>> _, wa = g.geocode('Washington, DC')
>>> _, pa = g.geocode('Palo Alto, CA')
>>> print((d(ne, cl) + d(cl, wa) + d(wa, pa)).miles)
3276.157156868931
```

class geopy.distance.**vincenty**(**args*, ***kwargs*)

Calculate the geodesic distance between two points using the formula devised by Thaddeus Vincenty, with an accurate ellipsoidal model of the earth.

Set which ellipsoidal model of the earth to use by specifying an `ellipsoid` keyword argument. The default is 'WGS-84', which is the most globally accurate model. If `ellipsoid` is a string, it is looked up in the *ELLIPSOIDS* dictionary to obtain the major and minor semiaxes and the flattening. Otherwise, it should be a tuple with those values. See the comments above the *ELLIPSOIDS* dictionary for more information.

Example:

```
>>> from geopy.distance import vincenty
>>> newport_ri = (41.49008, -71.312796)
>>> cleveland_oh = (41.499498, -81.695391)
>>> print(vincenty(newport_ri, cleveland_oh).miles)
538.3904451566326
```

Note: This implementation of Vincenty distance fails to converge for some valid points. In some cases, a result can be obtained by increasing the number of iterations (*iterations* keyword argument, given in the class *__init__*, with a default of 20). It may be preferable to use `great_circle`, which is marginally less accurate, but always produces a result.

class geopy.distance.**great_circle**(**args*, ***kwargs*)

Use spherical geometry to calculate the surface distance between two geodesic points. This formula can be written many different ways, including just the use of the spherical law of cosines or the haversine formula.

Set which radius of the earth to use by specifying a 'radius' keyword argument. It must be in kilometers. The default is to use the module constant *EARTH_RADIUS*, which uses the average great-circle radius.

Example:

```
>>> from geopy.distance import great_circle
>>> newport_ri = (41.49008, -71.312796)
>>> cleveland_oh = (41.499498, -81.695391)
>>> great_circle(newport_ri, cleveland_oh).miles
537.1485284062816
```

CHAPTER 3

Data

class geopy.location.**Location**(*address=''*, *point=None*, *raw=None*)

Contains a parsed geocoder response. Can be iterated over as (location<String>, (latitude<float>, longitude<Float)). Or one can access the properties *address*, *latitude*, *longitude*, or *raw*. The last is a dictionary of the geocoder's response for this item.

New in version 0.98.

address

Location as a formatted string returned by the geocoder or constructed by geopy, depending on the service.

Return type unicode

altitude

Location's altitude.

Return type float or None

latitude

Location's latitude.

Return type float or None

longitude

Location's longitude.

Return type float or None

raw

Location's raw, unparsed geocoder response. For details on this, consult the service's documentation.

Return type dict or None

class geopy.point.**Point**

A geodetic point with latitude, longitude, and altitude.

Latitude and longitude are floating point values in degrees. Altitude is a floating point value in kilometers. The reference level is never considered and is thus application dependent, so be consistent! The default for all values is 0.

Points can be created in a number of ways...

With longitude, latitude, and altitude:

```
>>> p1 = Point(41.5, -81, 0)
>>> p2 = Point(latitude=41.5, longitude=-81)
```

With a sequence of 0 to 3 values (longitude, latitude, altitude):

```
>>> p1 = Point([41.5, -81, 0])
>>> p2 = Point((41.5, -81))
```

Copy another *Point* instance:

```
>>> p2 = Point(p1)
>>> p2 == p1
True
>>> p2 is p1
False
```

Give a string containing at least latitude and longitude:

```
>>> p1 = Point('41.5,-81.0')
>>> p2 = Point('41.5 N -81.0 W')
>>> p3 = Point('-41.5 S, 81.0 E, 2.5km')
>>> p4 = Point('23 26m 22s N 23 27m 30s E 21.0mi')
>>> p5 = Point('''3 26' 22" N 23 27' 30" E''')
```

Point values can be accessed by name or by index:

```
>>> p = Point(41.5, -81.0, 0)
>>> p.latitude == p[0]
True
>>> p.longitude == p[1]
True
>>> p.altitude == p[2]
True
```

When unpacking (or iterating), a (latitude, longitude, altitude) tuple is returned:

```
>>> latitude, longitude, altitude = p
```

static __new__ (*latitude=None*, *longitude=None*, *altitude=None*)

Parameters

- **latitude** (*float*) – Latitude of point.
- **longitude** (*float*) – Longitude of point.
- **altitude** (*float*) – Altitude of point.

classmethod from_point (*point*)

Create and return a new `Point` instance from another `Point` instance.

classmethod from_sequence (*seq*)

Create and return a new `Point` instance from any iterable with 0 to 3 elements. The elements, if present, must be latitude, longitude, and altitude, respectively.

classmethod from_string (*string*)

Create and return a `Point` instance from a string containing latitude and longitude, and optionally, altitude.

Latitude and longitude must be in degrees and may be in decimal form or indicate arcminutes and arcseconds (labeled with Unicode prime and double prime, ASCII quote and double quote or 'm' and 's'). The degree symbol is optional and may be included after the decimal places (in decimal form) and before the arcminutes and arcseconds otherwise. Coordinates given from south and west (indicated by S and W suffixes) will be converted to north and east by switching their signs. If no (or partial) cardinal directions are given, north and east are the assumed directions. Latitude and longitude must be separated by at least whitespace, a comma, or a semicolon (each with optional surrounding whitespace).

Altitude, if supplied, must be a decimal number with given units. The following unit abbrevations (case-insensitive) are supported:

- `km` (kilometers)
- `m` (meters)
- `mi` (miles)
- `ft` (feet)
- `nm`, `nmi` (nautical miles)

Some example strings the will work include:

- 41.5;-81.0
- 41.5,-81.0
- 41.5 -81.0
- 41.5 N -81.0 W
- -41.5 S;81.0 E
- 23 26m 22s N 23 27m 30s E
- 23 26’ 22” N 23 27’ 30” E
- UT: N 39°20’ 0” / W 74°35’ 0’‘

CHAPTER 4

Exceptions

class geopy.exc.**GeopyError**
Geopy-specific exceptions are all inherited from GeopyError.

class geopy.exc.**ConfigurationError**
When instantiating a geocoder, the arguments given were invalid. See the documentation of each geocoder's *__init__* for more details.

class geopy.exc.**GeocoderServiceError**
There was an exception caused when calling the remote geocoding service, and no more specific exception could be raised by geopy. When calling geocoders' *geocode* or *reverse* methods, this is the most general exception that can be raised, and any non-geopy exception will be caught and turned into this. The exception's message will be that of the original exception.

class geopy.exc.**GeocoderQueryError**
Either geopy detected input that would cause a request to fail, or a request was made and the remote geocoding service responded that the request was bad.

class geopy.exc.**GeocoderQuotaExceeded**
The remote geocoding service refused to fulfill the request because the client has used its quota.

class geopy.exc.**GeocoderAuthenticationFailure**
The remote geocoding service rejects the API key or account credentials this geocoder was instantiated with.

class geopy.exc.**GeocoderInsufficientPrivileges**
The remote geocoding service refused to fulfill a request using the account credentials given.

class geopy.exc.**GeocoderTimedOut**
The call to the geocoding service was aborted because no response was receiving within the *timeout* argument of either the geocoding class or, if specified, the method call. Some services are just consistently slow, and a higher timeout may be needed to use them.

class geopy.exc.**GeocoderUnavailable**
Either it was not possible to establish a connection to the remote geocoding service, or the service responded with a code indicating it was unavailable.

class geopy.exc.**GeocoderParseError**
Geopy could not parse the service's response. This is a bug in geopy.

class geopy.exc.**GeocoderNotFound**
Caller requested the geocoder matching a string, e.g., "google" > GoogleV3, but no geocoder could be found.

CHAPTER 5

Logging

geopy will log geocoding URLs with a logger name *geopy* at level *DEBUG*, and for some geocoders, these URLs will include authentication information. If this is a concern, one can disable this logging by specifying a logging level of *NOTSET* or a level greater than *DEBUG* for logger name *geopy*. geopy does no logging above DEBUG.

CHAPTER 6

Changelog

6.1 1.10.0

2015-04-05

- **CHANGED: GeocodeFarm now uses version 3 of the service's API, which** allows use by unauthenticated users, multiple results, and SSL/TLS. You may need to obtain a new API key from GeocodeFarm, or use *None* for their free tier. Contributed by Eric Palakovich Carr.
- **ADDED: DataBC geocoder for use with the British Columbia government's** DataBC service. Contributed by Benjamin Trigona-Harany.
- **ADDED: Placefinder's geocode method now requests a timezone if the** *with_timezone* parameter is true. Contributed by willr.
- **FIXED: Nominatim specifies a *viewbox* parameter rather than the** apparently deprecated *view_box*.

6.2 1.9.1

2015-02-17

- **FIXED: Fix support for GoogleV3 bounds parameter. Contributed by** Benjamin Trigona-Harany.

6.3 1.9.0

2015-02-12

- **CHANGED: MapQuest geocoder removed as the API it uses is now only available** to enterprise accounts. OpenMapQuest is a replacement for Nominatim-sourced data.
- **CHANGED: Nominatim now uses HTTPS by default and accepts a *scheme*** argument. Contributed by srounet.
- **ADDED: Nominatim now accepts a *domain* argument, which** allows using a different server than *nominatim.openstreetmap.org*. Contributed by srounet.
- **FIXED: Bing was not accessible from *get_geocoder_for_service*. Contributed** by Adrián López.

6.4 1.8.1

2015-01-28

- **FIXED: GoogleV3 geocoder did not send API keys for reverse and timezone** methods.

6.5 1.8.0

2015-01-21

- ADDED: NaviData geocoder added. Contributed by NaviData.
- **CHANGED: LiveAddress now requires HTTPS connections. If you set *scheme*** to be *http*, rather than the default *https*, you will now receive a *ConfigurationError*.

6.6 1.7.1

2015-01-05

- **FIXED: IGN France geocoder's address formatting better handles results** that do not have a building number. Contributed by Thomas Gratier.

6.7 1.7.0

2014-12-30

- ADDED: IGN France geocoder. Contributed by Thomas Gratier.
- FIXED: Bing checks the response body for error codes.

6.8 1.6.1

2014-12-12

- FIXED: What3Words validation loosened. Contributed by spatialbitz.
- FIXED: Point.format() includes altitude.

6.9 1.6.0

2014-12-08

- ADDED: Python 3.2 and PyPy3 compatibility. Contributed by Mike Toews.

6.10 1.5.0

2014-12-07

- ADDED: Yandex geocoder added. Contributed by htch.
- ADDED: What3Words geocoder added. Contributed by spatialbitz.
- **FIXED: LiveAddress geocoder made compatible with a change in the service's** authentication. An *auth_id* parameter was added to the geocoder's initialization. Contributed by Arsen Mamikonyan.

6.11 1.4.0

2014-11-08

- ADDED: Mapquest.reverse() method added. Contributed by Dody Suria Wijaya.
- ADDED: Bing's geocoder now accepts the optional arguments "culture", "includeNeighborhood", and "include". Contributed by oskholl.

6.12 1.3.0

2014-09-23

- ADDED: Nominatim.geocode() accepts a *geometry* argument for retrieving *wkt*, *svg*, *kml*, or *geojson* formatted geometries in results. Contributed by spatialbitz.

6.13 1.2.0

2014-09-22

- ADDED: GeoNames.reverse() added. Contributed by Emile Aben.
- ADDED: GoogleV3.timezone() added. This returns a pytz object giving the timezone in effect for a given location at a time (defaulting to now).

6.14 1.1.5

2014-09-07

- FIXED: YahooPlaceFinder is now compatible with the older requests_oauthlib version 0.4.0.

6.15 1.1.4

2014-09-06

- FIXED: Point.format() seconds precision in Python 3.

6.16 1.1.3

2014-08-30

- FIXED: Fix OpenCage AttributeError on empty result. Contributed by IsaacHaze.

6.17 1.1.2

2014-08-12

- FIXED: Update Point __repr__ method to format _items properly. Contributed by TristanH.

6.18 1.1.1

2014-08-06

- FIXED: Python 3 compatibility.

6.19 1.1.0

2014-07-31

- ADDED: OpenCage geocoder added. Contributed by Demeter Sztanko.
- ADDED: *geopy.geocoders.get_geocoder_for_service* allows library authors to dynamically get a geocoder.
- FIXED: YahooPlacefinder bugs causing geocoding failure.
- FIXED: LiveAddress API URL updated.
- FIXED: Location.__repr__ unicode encode error in Python 2.7.
- CHANGED: *geopy.geocoders* modules now strictly declare their exports.

6.20 1.0.1

2014-07-24

- FIXED: The Baidu Maps geocoder's *_check_status* method used a Python 2-specific print statement.

6.21 1.0.0

2014-07-23

- ADDED: Baidu Maps geocoder added. Contributed by Risent.
- ADDED: Nominatim geocoder now supports structured queries. Contributed by kpanic.
- ADDED: Nominatim geocoder now supports a *language* parameter. Contributed by Benjamin Henne.
- CHANGED: GoogleV3's *geocode* and *reverse* methods have different orders for keyword argument parameters. Geocoders are now standardized on *(query, exactly_one, timeout, ...)*.

- FIXED: Removed rounding of minutes which was causing a formatted point to always have zero seconds. Contributed by Jonathan Batchelor.

For changes in the 0.9 series, see the `0.9x changelog`.

CHAPTER 7

Indices and search

- *genindex*
- *search*

g

Symbols

A

B

C

D

F

G

I

L

N

O

P

R

T

V

W

Y