Geocoder Documentation

Release 1.17.3

Denis Carriere

September 05, 2016

Contents

Release v1.17.3. (*Installation*)

Simple and consistent geocoding library written in Python.

Many online providers such as Google & Bing have geocoding services, these providers do not include Python libraries and have different JSON responses between each other.

It can be very difficult sometimes to parse a particular geocoding provider since each one of them have their own JSON schema.

Here is a typical example of retrieving a Lat & Lng from Google using Python, things shouldn't be this hard.

```
>>> import requests
>>> url = 'https://maps.googleapis.com/maps/api/geocode/json'
>>> params = {'sensor': 'false', 'address': 'Mountain View, CA'}
>>> r = requests.get(url, params=params)
>>> results = r.json()['results']
>>> location = results[0]['geometry']['location']
>>> location['lat'], location['lng']
(37.3860517, -122.0838511)
```

Now lets use Geocoder to do the same task.

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.latlng
(37.3860517, -122.0838511)
```

CHAPTER 1

Testimonials

Tobias Siebenlist Geocoder: great geocoding library by @DenisCarriere.

mcbetz Very good companion for Geocoder. Glad to see Python getting more geo libraries for Non-GIS users.

CHAPTER 2

API Documentation

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Overview

2.1.1 Installation

PyPi Install

To install Geocoder, simply:

```
$ pip install geocoder
```

GitHub Install

Installing the latest version from Github:

```
$ git clone https://github.com/DenisCarriere/geocoder
$ cd geocoder
$ python setup.py install
```

2.1.2 Examples

Many properties are available once the geocoder object is created.

Forward Geocoding

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.geojson
>>> g.json
>>> g.wkt
>>> g.osm
...
```

Reverse Geocoding

```
>>> g = geocoder.google([45.15, -75.14], method='reverse')
>>> g.city
>>> g.state
>>> g.state_long
>>> g.country
>>> g.country_long
...
```

House Addresses

```
>>> g = geocoder.google("453 Booth Street, Ottawa ON")
>>> g.housenumber
>>> g.postal
>>> g.street
>>> g.street_long
...
```

IP Addresses

```
>>> import geocoder
>>> g = geocoder.ip('199.7.157.0')
>>> g = geocoder.ip('me')
>>> g.latlng
>>> g.city
```

Command Line Interface

Basic usesage with CLI

```
$ geocode "Ottawa, ON" --provider bing
```

Saving results into a file

```
$ geocode "Ottawa, ON"  >> ottawa.geojson
```

Reverse geocoding with CLI

```
$ geocode "45.15, -75.14" --provider google --method reverse
```

Using JQ to query out a specific attribute

```
$ geocode "453 Booth Street" -p canadapost --output json | jq .postal
```

2.2 QGIS Field Calculator

Using the QGIS Field Calculator this will output WKT format.

2.2.1 Output Field

- **Name:** wkt
- **Type:** Text, unlimited length (text)

2.2.2 Function Editor

```
import geocoder

@qgsfunction(group='Geocoder')
def geocode(location, feature, parent):
    g = geocoder.google(location)
    return g.wkt
```

2.2.3 Expression

Find the **geocode** expression in the **Geocoder** function list, the final result will look something like this:

```
geocode("address")
```

Once the wkt field is added, you can then save your document as a CSV format and in the **Layer Options** define the **GEOMETRY = AS_WKT**.

CHAPTER 3

Providers

Detailed information about each individual provider that are within Geocoder.

3.1 ArcGIS

The World Geocoding Service finds addresses and places in all supported countries from a single endpoint. The service can find point locations of addresses, business names, and so on. The output points can be visualized on a map, inserted as stops for a route, or loaded as input for a spatial analysis. an address, retrieving imagery metadata, or creating a route.

3.1.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.arcgis('Redlands, CA')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'Redlands, CA' --provider arcgis
```

Parameters

- *location*: Your search location you want geocoded.
- *method*: (default=geocode) Use the following:
  - geocode

References

- ArcGIS Geocode API

3.2 Baidu

Baidu Maps Geocoding API is a free open the API, the default quota one million times / day.

3.2.1 Geocoding

```
>>> import geocoder # pip install geocoder
>>> g = geocoder.baidu('', key='<API KEY>')
>>> g.json
...
```

Command Line Interface

```
$ geocode '' --provider baidu
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export BAIDU_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: Baidu API key.
- *method*: (default=geocode) Use the following:
  - geocode

References

- API Reference
- Get Baidu key

3.3 Bing

The Bing™ Maps REST Services Application Programming Interface (API) provides a Representational State Transfer (REST) interface to perform tasks such as creating a static map with pushpins, geocoding an address, retrieving imagery metadata, or creating a route. Using Geocoder you can retrieve Bing's geocoded data from Bing Maps REST Services.

3.3.1 Geocoding

```
>>> import geocoder # pip install geocoder
>>> g = geocoder.bing('Mountain View, CA', key='<API KEY>')
>>> g.json
...
```

3.3.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.bing([45.15, -75.14], method='reverse')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'Mountain View, CA' --provider bing
$ geocode '45.15, -75.14' --provider bing --method reverse
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export BING_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: use your own API Key from Bing.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- Bing Maps REST Services

3.4 CanadaPost

The next generation of address finders, AddressComplete uses intelligent, fast searching to improve data accuracy and relevancy. Simply start typing a business name, address or Postal Code and AddressComplete will suggest results as you go. Using Geocoder you can retrieve CanadaPost's geocoded data from Addres Complete API.

3.4.1 Geocoding (Postal Code)

```
>>> import geocoder
>>> g = geocoder.canadapost('453 Booth Street, Ottawa', key='<API KEY>')
>>> g.postal
'K1R 7K9'
>>> g.json
...
```

Command Line Interface

```
$ geocode '453 Booth Street, Ottawa' --provider canadapost | jq .postal
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export CANADAPOST_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: (optional) API Key from CanadaPost Address Complete.
- *language*: (default=en) Output language preference.
- *country*: (default=ca) Geofenced query by country.
- *method*: (default=geocode) Use the following:
  - geocode

References

- Addres Complete API

3.5 Google

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map. Using Geocoder you can retrieve google's geocoded data from Google Geocoding API.

3.5.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.json
...
```

3.5.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.google([45.15, -75.14], method='reverse')
>>> g.json
...
```

3.5.3 Timezone

```
>>> import geocoder
>>> g = geocoder.google([45.15, -75.14], method='timezone')
>>> g.timeZoneName
'Eastern Daylight Time'
>>> g.timeZoneId
'America/Toronto'
>>> g.dstOffset
3600
>>> g.rawOffset
-18000
```

3.5.4 Component Filtering

```
>>> g = geocoder.google("Santa Cruz", components="country:ES")
```

Read me at Google's Geocoding API

https://developers.google.com/maps/documentation/geocoding/intro#ComponentFiltering

3.5.5 Elevation

```
>>> import geocoder
>>> g = geocoder.google([45.15, -75.14], method='elevation')
>>> g.meters
71.0
>>> g.feet
232.9
>>> g.resolution
38.17580795288086
```

Command Line Interface

```
$ geocode 'Mountain View, CA' --provider google
$ geocode '45.15, -75.14' --provider google --method reverse
$ geocode '45.15, -75.14' --provider google --method timezone
$ geocode '45.15, -75.14' --provider google --method elevation
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export GOOGLE_API_KEY=<Secret API Key>
$ export GOOGLE_CLIENT=<Secret Client>
$ export GOOGLE_CLIENT_SECRET=<Secret Client Secret>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: Your Google developers free key.
- *language*: 2-letter code of preferred language of returned address elements.
- *client*: Google for Work client ID. Use with client_secret. Cannot use with key parameter
- *client_secret*: Google for Work client secret. Use with client.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse
  - timezone
  - elevation

References

- Google Geocoding API

3.6 Mapbox

The Mapbox Geocoding API lets you convert location text into geographic coordinates (1600 Pennsylvania Ave NW → -77.0366,38.8971).

3.6.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.mapbox('San Francisco, CA', access_token='<TOKEN>')
>>> g.json
...
```

3.6.2 Reverse Geocoding

```
>>> import geocoder
>>> latlng = [45.3, -105.1]
>>> g = geocoder.mapbox(latlng, method='reverse')
>>> g.json
...
```

3.6.3 Geocoding with Proximity

Request feature data that best matches input and is biased to the given {latitude} and {longitude} coordinates. In the above example, a query of "200 Queen Street" returns a subset of all relevant addresses in the world. By adding the proximity option, this subset can be biased towards a given area, returning a more relevant set of results.

```
>>> import geocoder
>>> latlng = [45.3, -66.1]
>>> g = geocoder.mapbox("200 Queen Street", proximity=latlng)
>>> g.address
"200 Queen St, Saint John, E2L 2X1, New Brunswick, Canada"
>>> g = geocoder.mapbox("200 Queen Street")
"200 Queen St W, Toronto, M5T 1T9, Ontario, Canada"
...
```

Command Line Interface

```
$ geocode 'San Francisco, CA' --provider mapbox --out geojson
$ geocode '45.15, -75.14' --provider mapbox --method reverse
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export MAPBOX_ACCESS_TOKEN=<Secret Access Token>
```

Parameters

- *location*: Your search location you want geocoded.
- *proximity*: Search nearby [lat, lng].
- *access_token*: Use your own access token from Mapbox.
- *country*: Filtering by country code {cc} ISO 3166 alpha 2.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- Mabpox Geocoding API
- Get Mabpox Access Token

3.7 MapQuest

The geocoding service enables you to take an address and get the associated latitude and longitude. You can also use any latitude and longitude pair and get the associated address. Three types of geocoding are offered: address, reverse, and batch. Using Geocoder you can retrieve MapQuest's geocoded data from Geocoding Service.

3.7.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.mapquest('San Francisco, CA', key='<API KEY>')
>>> g.json
...
```

3.7.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.mapquest([45.15, -75.14], method='reverse', key='<API KEY>')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'San Francisco, CA' --provider mapquest --out geojson
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export MAPQUEST_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *method*: (default=geocode) Use the following:
  - geocode

References

- Mapquest Geocoding Service
- Get Free API Key

3.8 MaxMind

MaxMind's GeoIP2 products enable you to identify the location, organization, connection speed, and user type of your Internet visitors. The GeoIP2 databases are among the most popular and accurate IP geolocation databases available. Using Geocoder you can retrieve Maxmind's geocoded data from MaxMind's GeoIP2.

3.8.1 Geocoding (IP Address)

```
>>> import geocoder
>>> g = geocoder.maxmind('199.7.157.0')
>>> g.latlng
[45.413140, -75.656703]
>>> g.city
'Toronto'
>>> g.json
...
```

3.8.2 Geocode your own IP

To retrieve your own IP address, simply have *''* or *'me'* as the input.

```
>>> import geocoder
>>> g = geocoder.maxmind('me')
>>> g.latlng
[45.413140, -75.656703]
>>> g.ip
'199.7.157.0'
>>> g.json
...
```

Command Line Interface

```
$ geocode '8.8.8.8' --provider maxmind | jq .
```

Parameters

- *location*: Your search IP Address you want geocoded.
- *location*: (optional) *'me'* will return your current IP address's location.
- *method*: (default=geocode) Use the following:
  - geocode

References

- MaxMind's GeoIP2

3.9 Opencage

OpenCage Geocoder simple, easy, and open geocoding for the entire world Our API combines multiple geocoding systems in the background. Each is optimized for different parts of the world and types of requests.We aggregate the best results from open data sources and algorithms so you don't have to. Each is optimized for different parts of the world and types of requests. Using Geocoder you can retrieve OpenCage's geocoded data from OpenCage Geocoding Services.

3.9.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.opencage('San Francisco, CA', key='<API Key>')
>>> g.json
...
```

3.9.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.opencage([45.15, -75.14], method='reverse')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'San Francisco, CA' --provider opencage --out geojson --key '<API Key>' | jq .
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export OPENCAGE_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: (optional) use your own API Key from OpenCage.
- *method*: (default=geocode) Use the following:
  - geocode

References

- OpenCage Geocoding Services

3.10 OpenStreetMap

Nominatim (from the Latin, 'by name') is a tool to search OSM data by name and address and to generate synthetic addresses of OSM points (reverse geocoding). Using Geocoder you can retrieve OSM's geocoded data from Nominatim.

3.10.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.osm('New York city')
>>> g.json
...
```

3.10.2 Nominatim Server

Setting up your own offline Nominatim server is possible, using Ubuntu 14.04 as your OS and following the Nominatim Install instructions. This enables you to request as much geocoding as your little heart desires!

```
>>> url = 'http://localhost/nominatim/'
>>> url = 'localhost'
>>> g = geocoder.osm("New York City", url=url)
>>> g.json
...
```

3.10.3 OSM Addresses

The addr tag is the prefix for several 'addr:'* keys to describe addresses.

This format is meant to be saved as a CSV and imported into JOSM.

```
>>> g = geocoder.osm('11 Wall Street, New York')
>>> g.osm
{
    "x": -74.010865,
    "y": 40.7071407,
    "addr:country": "United States of America",
    "addr:state": "New York",
    "addr:housenumber": "11",
    "addr:postal": "10005",
    "addr:city": "NYC",
    "addr:street": "Wall Street"
}
```

Command Line Interface

```
$ geocode 'New York city' --provider osm --out geojson | jq .
$ geocode 'New York city' -p osm -o osm
$ geocode 'New York city' -p osm --url localhost
```

Parameters

- *location*: Your search location you want geocoded.
- *url*: Custom OSM Server (ex: localhost)
- *method*: (default=geocode) Use the following:
  - geocode

References

- Nominatim
- Nominatim Install
- addr tag

3.11 FreeGeoIP.net

freegeoip.net provides a public HTTP API for software developers to search the geolocation of IP addresses. It uses a database of IP addresses that are associated to cities along with other relevant information like time zone, latitude and longitude.

You're allowed up to 10,000 queries per hour by default. Once this limit is reached, all of your requests will result in HTTP 403, forbidden, until your quota is cleared.

3.11.1 Geocoding (IP Address)

```
>>> import geocoder
>>> g = geocoder.freegeoip('99.240.181.199')
>>> g.json
...
```

Command Line Interface

```
$ geocode '99.240.181.199' --provider freegeoip
```

Parameters

- *location*: Your search location you want geocoded.
- *method*: (default=geocode) Use the following:
  - geocode

References

- API Reference

3.12 GeocodeFarm

Geocode.Farm is one of the few providers that provide this highly specialized service for free. We also have affordable paid plans, of course, but our free services are of the same quality and provide the same results. The major difference between our affordable paid plans and our free API service is the limitations. On one of our affordable paid plans your limit is set based on the plan you signed up for, starting at 25,000 query requests per day (API calls). On our free API offering, you are limited to 250 query requests per day (API calls).

3.12.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.geocodefarm('Mountain View, CA')
>>> g.json
...
```

3.12.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.geocodefarm([45.15, -75.14], method='reverse')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'Mountain View, CA' --provider geocodefarm
$ geocode '45.15, -75.14' --provider geocodefarm --method reverse
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export GEOCODEFARM_API_KEY=<Secret API Key>
```

Parameters

- *location*: The string to search for. Usually a street address. If reverse then should be a latitude/longitude.
- *key*: (optional) API Key. Only Required for Paid Users.
- *lang*: (optional) 2 digit lanuage code to return results in. Currently only "en"(English) or "de"(German) supported.
- *country*: (optional) The country to return results in. Used for biasing purposes and may not fully filter results to this specific country.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- GeocodeFarm API Documentation

3.13 Geocoder.ca

Geocoder.ca - A Canadian and US location geocoder. Using Geocoder you can retrieve Geolytica's geocoded data from Geocoder.ca.

3.13.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.geolytica('Ottawa, ON')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'Ottawa, ON' --provider geolytica
```

Parameters

- *location*: Your search location you want geocoded.
- *auth*: The authentication code for unthrottled service.
- *strictmode*: Optionally you can prevent geocoder from making guesses on your input.
- *strict*: Optional Parameter for enabling strict parsing of free form location input.
- *method*: (default=geocode) Use the following:
  - geocode
- *auth*: (optional) The authentication code for unthrottled service (premium API)

References

- API Reference

3.14 GeoOttawa

This data was collected in the field using GPS software on handheld computers. Not all information has been verified for accuracy and therefore should only be used in an advisory capacity. Forestry Services reserves the right to revise the data pursuant to further inspection/review. If you find any errors or omissions, please report them to 3-1-1.

3.14.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.ottawa('453 Booth Street')
>>> g.json
...
```

Command Line Interface

```
$ geocode '453 Booth Street' --provider ottawa
```

Parameters

- *location*: Your search location you want geocoded.
- *method*: (default=geocode) Use the following:
  - geocode

References

- GeoOttawa Map

3.15 HERE

Send a request to the geocode endpoint to find an address using a combination of country, state, county, city, postal code, district, street and house number. Using Geocoder you can retrieve geocoded data from the HERE Geocoder REST API.

3.15.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.here('Espoo, Finland')
>>> g.json
...
```

3.15.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.google([45.15, -75.14], method='reverse')
>>> g.json
...
```

Using API Key

If you want to use your own *app_id* & *app_code*, you must register an app at the HERE Developer.

```
>>> g = geocoder.here('Espoo, Finland',
                      app_id='<YOUR APP ID>',
                      app_code='<YOUR APP CODE>')
```

Command Line Interface

```
$ geocode 'Espoo, Finland' --provider here
$ geocode '45.15, -75.14' --provider here --method reverse
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export APP_ID=<Secret APP ID>
$ export APP_CODE=<Secret APP Code>
```

Parameters

- *location*: Your search location you want geocoded.
- *app_code*: (optional) use your own Application Code from HERE.
- *app_id*: (optional) use your own Application ID from HERE.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- HERE Geocoder REST API

3.16 IP Info.io

Use the IPInfo.io IP lookup API to quickly and simply integrate IP geolocation into your script or website. Save yourself the hassle of setting up local GeoIP libraries and having to remember to regularly update the data.

3.16.1 Geocoding (IP Address)

```
>>> import geocoder
>>> g = geocoder.ipinfo('199.7.157.0')
>>> g.latlng
[45.413140, -75.656703]
>>> g.city
'Toronto'
>>> g.json
...
```

3.16.2 Geocode your own IP

To retrieve your own IP address, simply have *''* or *'me'* as the input.

```
>>> import geocoder
>>> g = geocoder.ipinfo('me')
>>> g.latlng
[45.413140, -75.656703]
>>> g.ip
'199.7.157.0'
```

```
>>> g.json
...
```

Command Line Interface

```
$ geocode '199.7.157.0' --provider ipinfo | jq .
```

Parameters

- *location*: Your search location you want geocoded.
- *location*: (optional) '' will return your current IP address's location.
- *method*: (default=geocode) Use the following:
  - geocode

References

- IpinfoIo

3.17 Tamu

The Texas A&M Geoservices Geocoding API provides output including Lat and Lon and numerous census data values.

An API key linked to an account with Texas A&M is required.

Tamu's API differs from the other geocoders in this package in that it requires the street address, city, state, and US zipcode to be passed in separately, rather than as a single string. Because of this requirement, the "location", "city", "state", and "zipcode" parameters are all required when using the Tamu provider. The "location" parameter should contain only the street address of the location.

3.17.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.tamu(
            '595 Market St',
            city='San Francisco',
            state='California',
            zipcode='94105',
            key='demo')
>>> g.json
...
```

Command Line Interface

```
$ geocode '595 Market St' --provider tamu --city San Francisco --state CA --zipcode 94105 --key <Secr
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export TAMU_API_KEY=<Secret API Key>
```

Parameters

- *location*: The street address of the location you want geocoded.
- *city*: The city of the location to geocode.
- *state*: The state of the location to geocode.
- *zipcode*: The zipcode of the location to geocode.
- *key*: use your own API Key from Tamu.
- *method*: (default=geocode) Use the following:
  - geocode

Census Output Fields

Note: "FIPS" stands for "Federal Information Processing System"

- *census_block*: Census Block value for location
- *census_tract*: Census Tract value for location
- *census_county_fips*: Census County FIPS value
- *census_cbsa_fips*: Census Core Base Statistical Area FIPS value
- *census_mcd_fips*: Census Minor Civil Division FIPS value
- *census_msa_fips*: Census Metropolitan Statistical Area FIPS value
- *census_place_fips*: Census Place FIPS value
- *census_state_fips*: Census State FIPS value
- *census_year*: Census Year from which these values originated

References

- Tamu Geocoding API

3.18 TomTom

The Geocoding API gives developers access to TomTom's first class geocoding service. Developers may call this service through either a single or batch geocoding request. This service supports global coverage, with house number level matching in over 50 countries, and address point matching where available. Using Geocoder you can retrieve TomTom's geocoded data from Geocoding API.

3.18.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.tomtom('San Francisco, CA', key='<API KEY>')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'San Francisco, CA' --provider mapbox --out geojson
```

Environment Variables

To make sure your API key is store safely on your computer, you can define that API key using your system's environment variables.

```
$ export TOMTOM_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: use your own API Key from TomTom.
- *method*: (default=geocode) Use the following:
  - geocode

References

- TomTom Geocoding API

3.19 What3Words

What3Words is a global grid of 57 trillion 3mx3m squares. Each square has a unique 3 word address that people can find and communicate quickly, easily, and without ambiguity.

Addressing the world

Everyone and everywhere now has an address

3.19.1 Geocoding (3 Words)

```
>>> import geocoder
>>> g = geocoder.w3w('embedded.fizzled.trial')
>>> g.json
...
```

3.19.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.w3w([45.15, -75.14], method='reverse')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'embedded.fizzled.trial' --provider w3w
$ geocode '45.15, -75.14' --provider w3w --method reverse
```

Environment Variables

For safe storage of your API key on your computer, you can define that API key using your system's environment variables.

```
$ export W3W_API_KEY=<Secret API Key>
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: use your own API Key from What3Words.
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- API Reference
- Get W3W key

3.20 Yahoo

Yahoo PlaceFinder is a geocoding Web service that helps developers make their applications location-aware by converting street addresses or place names into geographic coordinates (and vice versa). Using Geocoder you can retrieve Yahoo's geocoded data from Yahoo BOSS Geo Services.

3.20.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.yahoo('San Francisco, CA')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'San Francisco, CA' --provider yahoo --out geojson
```

Parameters

- *location*: Your search location you want geocoded.
- *method*: (default=geocode) Use the following:
  - geocode

References

- Yahoo BOSS Geo Services

3.21 Yandex

Yandex (Russian:) is a Russian Internet company which operates the largest search engine in Russia with about 60% market share in that country.

The Yandex home page has been rated as the most popular website in Russia.

3.21.1 Geocoding

```
>>> import geocoder
>>> g = geocoder.yandex('Moscow Russia')
>>> g.json
...
```

3.21.2 Reverse Geocoding

```
>>> import geocoder
>>> g = geocoder.yandex([55.95, 37.96], method='reverse')
>>> g.json
...
```

Command Line Interface

```
$ geocode 'Moscow Russia' --provider yandex --out geojson
$ geocode '55.95, 37.96' --provider yandex --method reverse
```

Parameters

- *location*: Your search location you want geocoded.
- *lang*: Chose the following language:
  - **ru-RU** — Russian (by default)

  - **uk-UA** — Ukrainian
  - **be-BY** — Belarusian
  - **en-US** — American English
  - **en-BR** — British English
  - **tr-TR** — Turkish (only for maps of Turkey)
- *kind*: Type of toponym (only for reverse geocoding):
  - **house** — house or building
  - **street** — street
  - **metro** — subway station
  - **district** — city district
  - **locality** — locality (city, town, village, etc.)
- *method*: (default=geocode) Use the following:
  - geocode
  - reverse

References

- Yandex API Reference

3.22 TGOS

TGOS Map is official map service of Taiwan.

3.22.1 Geocoding

```
>>> import geocoder # pip install geocoder
>>> g = geocoder.tgos('735', key='<API KEY>')
>>> g.json
...
```

Command Line Interface

```
$ geocode '735' --provider tgos
```

Parameters

- *location*: Your search location you want geocoded.
- *key*: Use your own API Key from TGOS.
- *method*: (default=geocode) Use the following:
  - geocode

- *language*: (default=taiwan) Use the following:
  - taiwan
  - english
  - chinese

References

- TGOS Maps API

CHAPTER 4

Contributor Guide

If you want to contribute to the project, this part of the documentation is for you.

4.1 Authors

4.1.1 Lead Developer

- Denis Carriere - Creator of Python's Geocoder

4.1.2 Contributors

A big thanks to all the people that help contribute:

- Virus Warnning - Implemented TGOS provider
- Kevin Brolly - Implemented GeocodeFarm provider
- Michael R. Okun - Implemented Tamu provider
- Palo Dravecky - Added Google for Work.
- Dunice Vadimh - Added IPInfo provider.
- Yed Podtrzitko - Cleaned up code & Added Six
- Thomas Gratier - Wrote an article about Geocoder vs. Geopy
- Max Arnold - Submitted Github Issue
- Thanh Ha - Cleaned up code & Unit Testing
- Mahdi Yusuf - Promoted by Pycoders Weekly, Issue #155 Nimoy
- Alex Pilon - Cleaned up code
- Philip Hubertus - Provided HERE improvements & documentation
- Antonio Lima - Improved code quality and introduced Rate Limits
- Alexander Lukanin - Improved Python 3 compatibilty
- flebel - Submitted Github Issues
- patrickyan - Submitted Github Issues
- esy - Submitted Github Issues

- Sergei Grabalin ( ) - Fixed Python2 Unicode Issues