pgeocode Documentation

Release 0.3.0

Roman Yurchak

Apr 07, 2021

Contents:

CHAPTER 1

pgeocode

Postal code geocoding and distance calculations

pgeocode is a Python library for high performance off-line querying of GPS coordinates, region name and municipality name from postal codes. Distances between postal codes as well as general distance queries are also supported. The used GeoNames database includes postal codes for 83 countries.

Currently, only queries within the same country are supported.

For additional documentation see pgeocode.readthedocs.io.

1.1 Installation

pgeocode requires Python 3.6+ as well as `numpy` and `pandas` packages. It can be installed with,

```
pip install pgeocode
```

or

```
conda install -c conda-forge pgeocode
```

1.2 Quickstart

Postal code queries

```
>>> import pgeocode

>>> nomi = pgeocode.Nominatim('fr')
>>> nomi.query_postal_code("75013")
postal_code               75013
```

(continues on next page)

(continued from previous page)

```
country_code                FR
place_name             Paris 13
state_name        Île-de-France
state_code                   11
county_name               Paris
county_code                  75
community_name            Paris
community_code              751
latitude                48.8322
longitude                2.3561
accuracy                      5

>>> nomi.query_postal_code(["75013", "69006"])
      postal_code place_name            state_name  latitude  longitude
0       75013   Paris 13         Île-de-France   48.8322     2.3561
1       69006    Lyon 06  Auvergne-Rhône-Alpes   45.7679     4.8506
```

Distance calculations

```
>>> dist = pgeocode.GeoDistance('fr')
>>> dist.query_postal_code("75013", "69006")
389.156
>>> dist.query_postal_code(["75013", "75014", "75015"], ["69006", "69005", "69004"])
array([ 389.15648697,  390.12577967,  390.49857655])
```

1.3 Geocoding format

The result of a geo-localistion query is a `pandas.DataFrame` with the following columns,

- `country_code`: iso country code, 2 characters
- `postal_code` : postal code
- `place_name` : place name (e.g. town, city etc)
- `state_name` : 1. order subdivision (state)
- `state_code` : 1. order subdivision (state)
- `county_name` : 2. order subdivision (county/province)
- `county_code` : 2. order subdivision (county/province)
- `community_name` : 3. order subdivision (community)
- `community_code` : 3. order subdivision (community)
- `latitude` : estimated latitude (wgs84)
- `longitude` : estimated longitude (wgs84)
- `accuracy` : accuracy of lat/lng from 1=estimated to 6=centroid

1.4 Configuration and defaults

Storage directory

Defaults to `~/pgeocode_data`, it is the directory where data is downloaded for later consumption. It can be changed using the environment variable `PGEOCODE_DATA_DIR`, i.e. `export PGEOCODE_DATA_DIR=/tmp/pgeocode_data`.

Data sources

Data sources are provided as a list in the `pgeocode.DOWNLOAD_URL` variable. The default value is,

```
DOWNLOAD_URL = [
    "https://download.geonames.org/export/zip/{country}.zip",
    "https://symerio.github.io/postal-codes-data/data/geonames/{country}.txt",
]
```

Data sources are tried from first to last until one works. Here the second link is a mirror of the first.

It is also possible to extend this variable with third party data sources, as long as they follow the same format. See for instance postal-codes-data repository for examples of data files.

1.5 License

The pgeocode package is distributed under the 3-clause BSD license.

1.6 Supported countries

The list of countries available in the GeoNames database, with the corresponding country codes, are given below,

Andorra (AD), Argentina (AR), American Samoa (AS), Austria (AT), Australia (AU), Åland Islands (AX), Bangladesh (BD), Belgium (BE), Bulgaria (BG), Bermuda (BM), Brazil (BR), Belarus (BY), Canada (CA), Switzerland (CH), Colombia (CO), Costa Rica (CR), Czechia (CZ), Germany (DE), Denmark (DK), Dominican Republic (DO), Algeria (DZ), Spain (ES), Finland (FI), Faroe Islands (FO), France (FR), United Kingdom of Great Britain and Northern Ireland (GB), French Guiana (GF), Guernsey (GG), Greenland (GL), Guadeloupe (GP), Guatemala (GT), Guam (GU), Croatia (HR), Hungary (HU), Ireland (IE), Isle of Man (IM), India (IN), Iceland (IS), Italy (IT), Jersey (JE), Japan (JP), Liechtenstein (LI), Sri Lanka (LK), Lithuania (LT), Luxembourg (LU), Latvia (LV), Monaco (MC), Republic of Moldova (MD), Marshall Islands (MH), The former Yugoslav Republic of Macedonia (MK), Northern Mariana Islands (MP), Martinique (MQ), Malta (MT), Mexico (MX), Malaysia (MY), New Caledonia (NC), Netherlands (NL), Norway (NO), New Zealand (NZ), Philippines (PH), Pakistan (PK), Poland (PL), Saint Pierre and Miquelon (PM), Puerto Rico (PR), Portugal (PT), Réunion (RE), Romania (RO), Russian Federation (RU), Sweden (SE), Slovenia (SI), Svalbard and Jan Mayen Islands (SJ), Slovakia (SK), San Marino (SM), Thailand (TH), Turkey (TR), Ukraine (UA), United States of America (US), Uruguay (UY), Holy See (VA), United States Virgin Islands (VI), Wallis and Futuna Islands (WF), Mayotte (YT), South Africa (ZA)

See GeoNames database for more information.

CHAPTER 2

Contributing

At present `pgeocode.py` is a single module to faciliate vendoring, please add your changes to this file.

2.1 Testing

Unit tests can be run with,

```
pip install pytest pytest-httpserver
```

```
pytest
```

You can also run `tox` to run the unit test will all the supported python versions.

2.2 Code style

Pgeocode uses black and flake8 for the code style.

To apply them automatically, install pre-commmit

```
pip install pre-commit
pre-commit install
```

then fix any style errors that occurs when you commit. You may run it twice to apply black and isort. It is also possible to run them manually with,

```
pre-commit run -a
```

CHAPTER 3

API Reference

pgeocode.Nominatim(country, unique)	Query geographical location from a city name or a postal code
pgeocode.GeoDistance(country, errors)	Distance calculation from a city name or a postal code
pgeocode.haversine_distance(x, y)	Haversine (great circle) distance

3.1 pgeocode.Nominatim

class pgeocode.**Nominatim**(*country: str = 'fr'*, *unique: bool = True*)

Query geographical location from a city name or a postal code

Parameters

- **country** (*str, default='fr'*) – country code. See the documentation for a list of supported countries.
- **unique** (*bool, default=True*) – Create unique postcode index, merging all places with the same postcode into a single entry

__init__(*country: str = 'fr'*, *unique: bool = True*)

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

Methods

__init__(country, unique)	Initialize self.
query_location(name)	Get locations information from a community/minicipality name
query_postal_code(codes)	Get locations information from postal codes

query_location(*name*)

Get locations information from a community/minicipality name

query_postal_code(*codes*)
Get locations information from postal codes

Parameters **codes** (*array, list or int*) – an array of strings containing postal codes

Returns **df** – a pandas.DataFrame with the relevant information

Return type pandas.DataFrame

3.2 pgeocode.GeoDistance

class pgeocode.**GeoDistance**(*country: str = 'fr', errors: str = 'ignore'*)
Distance calculation from a city name or a postal code

Parameters

- **data_path** (*str*) – path to the dataset
- **error** (*str, default='ignore'*) – how to handle not found elements. One of 'ignore' (return NaNs), 'error' (raise an exception), 'nearest' (find from nearest valid points)

__init__(*country: str = 'fr', errors: str = 'ignore'*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(country, errors)	Initialize self.
query_location(name)	Get locations information from a community/minicipality name
query_postal_code(x, y)	Get distance (in km) between postal codes

query_location(*name*)
Get locations information from a community/minicipality name

query_postal_code(*x*, *y*)
Get distance (in km) between postal codes

Parameters

- **x** (*array, list or int*) – a list of postal codes
- **y** (*array, list or int*) – a list of postal codes

Returns **d** – the calculated distances

Return type array or int

3.3 pgeocode.haversine_distance

pgeocode.**haversine_distance**(*x*, *y*)
Haversine (great circle) distance

Calculate the great circle distance between two points on the earth (specified in decimal degrees)

Parameters

- **x** (*array, shape=(n_samples, 2)*) – the first list of coordinates (degrees)

- **y** (*array: shape=(n_samples, 2)*) – the second list of coordinates (degress)

Returns **d** – the distance between corrdinates (km)

Return type array, shape=(n_samples,)

References

https://en.wikipedia.org/wiki/Great-circle_distance

Index

Symbols

G

H

N

Q