Geocoder Documentation

Release 2.0.0

Andrey Shpak

May 25, 2022

CONTENTS

Release v2.0.0.

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.

```python
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.

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

CHAPTER

ONE

API DOCUMENTATION

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

1.1 Features

As base for many geocoders *geocoder3* project provide some functions, available in any supported geocoder. Some functions are implementation of same function included in some providers by default. In this case function property class can be replaced in provider results definition.

1.1.1 Confidence Score Calculation

Confidence score based on OpenCage API implementation, but available in any supported geocoder. For geocoders without default confidence score support this property calculated by same definition in `geocoder3` internal process.

What is Confidence Score

The OpenCage Geocoder will always attempt to find a match for as many parts of a query as it can, but this isn't always possible to do. Where a partial match is made, for example a street name can be matched but a specific house number on that street cannot be matched, the geocoder will still return a result but the granularity of the match will not be as high as if the house number was matched.

The confidence that the geocoder has in a match returned to the confidence field. This contains a value between 0 and 10, where 0 reflects no confidence and 10 reflects high confidence.

Confidence is calculated by measuring the distance in kilometres between the South West and North East corners of each results bounding box; a smaller distance represents a high confidence while a large distance represents a lower confidence.

The best way to think of our confidence score is as a measure of how confident we are that centre point coordinates returned for the result precisely reflect the result. So for example, if you search for "Berlin, Germany", we know exactly where that is, but it has a confidence of only 4, as Berlin is a large city (and Bundesland, but that's another story). The coordinates we return are in the centre of the bounding box, but it would be valid to consider anywhere in that box to be "Berlin", hence the relatively low confidence score.

Score	Description
10	less than 0.25 km distance
9	less than 0.5 km distance
8	less than 1 km distance
7	less than 5 km distance
6	less than 7.5 km distance
5	less than 10 km distance
4	less than 15 km distance
3	less than 20 km distance
2	less than 25 km distance
1	25 km or greater distance
0	unable to determine a bounding box

1.1.2 Well Known Text(WKT) output

`geocoder3` will try to generate WKT output for any retrieved result, was it provided by provider or not. WKT available for each result if multiple results was retrieved.

```python
import geocoder
g = geocoder.google('New York City')
g.wkt
# 'POINT(-74.0111421 40.7069226)'
```

What is WKT

Well-known text (WKT) is a text markup language for representing vector geometry objects on a map, spatial reference systems of spatial objects and transformations between spatial reference systems. A binary equivalent, known as well-known binary (WKB), is used to transfer and store the same information on databases, such as PostGIS, Microsoft SQL Server and DB2. The formats were originally defined by the Open Geospatial Consortium (OGC) and described in their Simple Feature Access and Coordinate Transformation Service specifications.

Wikipedia WKT

1.2 Supported providers

1.2.1 Base provider definition

Base classes of provider definition responsible for minimum set of methods and properties, that should be implemented or overridden in all nested providers.

This set of methods and properties guarantees working of all project *features* and minimum similarity of result of any provider usage.

Each provider itself can extend supported and extracted properties, available in direct instance access. For list of such extracted properties please read documentation for exact provider.

Base Multiple Results Query class

class geocoder.base.**MultipleResultsQuery**(*location*, *url=None*, *key=None*, *timeout=None*, *proxies=None*, *session=None*, *headers=None*, *params=None*, ***kwargs*)

Base results and query manager container

This class responsible for checking correct new provider files creation before it will be implemented in project. Such checks done in `__init_subclass__()` method and will not allow to initialize project without fix.

Class variables:

Some class variables are mandatory for all nested subclasses.

Variables

- **cls._URL** (*str*) – Default URL for provider, can be overwritten with *url* input parameter
- **cls._RESULT_CLASS** (OneResult) – Provider's individual result class.
- **cls._KEY** (*str*) – Provider's default api_key. Usually map to ENV variable responsible for key parsing. Can be overwritten with **key** parameter on instance creation. Shows actually used key when requested from instance.
- **cls._KEY_MANDATORY** (*bool*) – Special mark for check of mandatory presence of api key, for providers with mandatory key requirement
- **cls._METHOD** (*str*) – Provider's internal method, that should match with api.py `options` definition.
- **cls._PROVIDER** (*str*) – Provider's internal name, that should match with api.py `options` definition.
- **cls._TIMEOUT** (*float*) – Default timeout for `requests.request()` configuration, can be overwritten on instance creation or instance calling
- **cls._GEOCODER3_READY** (*bool*) – Temporary value, representing is provider tested and finished migration to geocoder3. On default value will generate warning on any provider call.

Instance variables:

After creation each instance of *MultipleResultsQuery* has the following mandatory variables. For some providers this list can be extended by provider implementation.

Variables

- **self.results_list** (*list[OneResult]*) – Hold all answers from provider in parsed state
- **self.url** (*str*) – Final request url that will be/was used during request
- **self.location** (*str*) – Object to geocode/reverse geocode
- **self.timeout** (*float*) – Final request timeout that was used during request
- **self.proxies** (*Optional[dict]*) – Final request proxies that was used during request
- **self.session** (*requests.Session*) – `requests.Session` object, that was used
- **self.headers** (*dict*) – Final request headers that was used during request
- **self.params** (*dict*) – Final request query params that was used during request
- **self.status_code** (*Optional[int]*) – `requests.Response` final HTTP answer code or *None* if request is not made yet, or `requests` failed during request

- **`self.raw_response`** (*`requests.Response`*) – Contain raw `requests.Response` from provider
- **`self.raw_json`** (*`Union[dict, list]`*) – Contain raw `requests.Response.json()` from provider
- **`self.error`** (*`str`*) – `requests` detailed error, if was raised during request
- **`self.is_called`** (*`bool`*) – *False* on instance initialization, become *True* after calling of *`__call__()`* method(i.e. instance call)
- **`self.current_result`** (`OneResult`) – Mapping to result, that are used for direct attributes retrieval in *`__getattr__()`*

Init parameters:

For initialization parameters, please check *`MultipleResultsQuery.__init__()`* method documentation.

`__call__`(*timeout=None*, *proxies=None*, *session=None*)

Query remote server and parse results

Any keyword argument of *`__call__()`* will have precedence over same argument in *`__init__()`* method.

Parameters

- **`timeout`** (*`Union[None, float, Tuple[float, float], Tuple[float, None]]`*) – Max request answer wait time
- **`proxies`** (*`Optional[MutableMapping[str, str]]`*) – Proxies for `requests.request()`
- **`session`** (*`Optional[requests.Session]`*) – Custom `requests.Session` for request

`__getattr__`(*name*)

Allow direct access to `MultipleResultsQuery.current_result` attributes from direct calling of *`MultipleResultsQuery`*

Called when an attribute lookup has not found the attribute in the usual places (i.e. it is not an instance attribute nor is it found in the class tree for self).

Note: If the attribute is found through the normal mechanism, *`__getattr__()`* is not called.

Parameters **`name`** – Attribute name for lookup

Raises **`RuntimeError`** – If provider query was not made and `current_result` is still empty. (From *`has_data()`*)

`__init__`(*location*, *url=None*, *key=None*, *timeout=None*, *proxies=None*, *session=None*, *headers=None*, *params=None*, ***kwargs*)

Initialize a *`MultipleResultsQuery`* object.

For class and instance variables description please refer to class docstrings.

Parameters

- **`location`** – Query content for geocode or reverse geocoding
- **`url`** (*`Optional[str]`*) – Overwrite for default provider service url
- **`key`** (*`Optional[str]`*) – API Key data for provider usage, if required. Passed to *`_get_api_key()`*, which result passed to *`_build_headers()`* and *`_build_params()`*,

and may be passed to other custom provider's implementation methods. Check exact provider docs.

- **timeout** (*Union[None, float, Tuple[float, float], Tuple[float, None]]*) – Max request answer wait time
- **proxies** (*Optional[MutableMapping[str, str]]*) – Proxies for `requests.request()`
- **session** (*Optional[requests.Session]*) – Custom `requests.Session` for request
- **headers** (*Optional[MutableMapping[str, str]]*) – Additional headers for `requests.request()`
- **params** (*Optional[dict]*) – Additional query parameters
- **kwargs** – Any other keyword arguments, that will be passed to internal *_build_headers()*, *_build_params()*, *_before_initialize()* or other custom provider's implementation methods. Check exact provider docs

Raises

- **ValueError** – When provided custom `url` is not well-formatted
- **ValueError** – If api key was not provided, but mandatory for provider use

classmethod __init_subclass__(***kwargs*)

Responsible for setup check for *MultipleResultsQuery* subclasses.

Raises

- **ValueError** – When subclass not define `cls._URL` value.
- **ValueError** – When subclass incorrectly define `cls._RESULT_CLASS` value.
- **ValueError** – When subclass incorrectly define `cls._METHOD` value.

_adapt_results(*json_response*)

Allow children classes to format json_response into *_parse_results()* expected format

This required for correct iteration in *_parse_results()*

Parameters **json_response** – Raw json from provider, usually same as in `raw_json`, by default invoked inside *_parse_results()*

_before_initialize(*location*, ***kwargs*)

Hook for children class to finalize their setup before the query

Parameters

- **location** – Query content for geocode or reverse geocoding
- **kwargs** – All kwargs from *__init__()* method

_build_headers(*provider_key*, ***kwargs*)

Generate default query headers for provider

Parameters

- **provider_key** – Finalized api_key, from *_get_api_key()* method
- **kwargs** – All kwargs from *__init__()* method

`_build_params`(*location*, *provider_key*, ***kwargs*)

Generate default query parameters mapping for provider

Parameters

- **`location`** – Query content for geocode or reverse geocoding
- **`provider_key`** – Finalized api_key, from *`_get_api_key()`* method
- **`kwargs`** – All kwargs from *`__init__()`* method

`_catch_errors`(*json_response*)

Checks the JSON returned from the provider and flag errors if necessary

`_connect`()

Responsible for handling external request and connection errors

`classmethod` **`_get_api_key`**(*key=None*)

Retrieves API Key from method argument first, then from Environment variables

Parameters **key** (*`Optional[str]`*) – Custom API Key data for provider usage, if required. Passed from *`__init__()`* method.

Raises `ValueError` – If api key was not provided, but mandatory for provider use

`_parse_results`(*json_response*)

Responsible for parsing original json and separating it to *`OneResult`* objects

`debug`()

Display debug information for instance of *`MultipleResultsQuery`*

`rate_limited_get`(*url*, ***kwargs*)

By default, simply wraps a `requests.get()` request

`property` **`geojson`**

Output all answers as GeoJSON FeatureCollection

`property` **`has_data`**

Status of geocoding if request was made

Raises `RuntimeError` – When external request was not made before property call

`property` **`status`**

Specify current summary status of instance

Possible statuses:

- "External request was not made"
- "OK" - when request was made, and any result retrieved
- `requests` error text representation, if request faced error
- "ERROR - No results found"
- "ERROR - Unhandled Exception"

Base One Result class

`class geocoder.base.OneResult(json_content)`

Container for one (JSON) object returned by provider

Class variables:

Variables

- `cls._TO_EXCLUDE` – List of properties and attributes to exclude in `OneResult._parse_json_with_fieldnames()`
- `cls._GEOCODER3_READY` (*bool*) – Temporary value, representing is provider tested and finished migration to geocoder3. On default value will bypass some internal checks.

Instance variables:

After creation each instance of `OneResult` has the following mandatory variables. For some providers this list can be extended by provider implementation.

Variables

- `self.object_raw_json` – Raw json for object, passed by `MultipleResultsQuery._parse_results()`
- `self.object_json` – Result of `OneResult._parse_json_with_fieldnames()`
- `self.fieldnames` – Fieldnames list generated in `OneResult._parse_json_with_fieldnames()`

Init parameters:

For initialization parameters, please check `OneResult.__init__()` method documentation.

`__init__(json_content)`

Initialize `OneResult` object and parse input json

Parameters `json_content` (*dict*) – Dictionary, passed by `MultipleResultsQuery.__call__()`

`_parse_json_with_fieldnames()`

Parse the instance object with all attributes/methods defined in the class, except for the ones defined starting with '_' or flagged in `cls._TO_EXCLUDE`.

The final result is stored in `self.object_json` and `self.fieldnames`

`debug()`

Display debug information for instance of `OneResult`

`abstract property address`

Object simple string address.

`property bbox`

Output answer as GeoJSON bbox if it can be calculated/retrieved.

`property bounds`

Output answer as Google Maps API bounds if it can be calculated/retrieved.

`property confidence`

Is as a measure of how confident we are that centre point coordinates returned for the result precisely reflect the result.

property east

Return optional east coordinate of bbox, if available.

property geojson

Output answer as GeoJSON Feature

property geometry

Output answer as GeoJSON Point

abstract property lat

Latitude of the object

property latlng

Optional list of latitude and longitude values.

abstract property lng

Longitude of the object

property north

Return optional north coordinate of bbox, if available.

property northeast

Return north-east list of coordinates for bounds, if available.

property ok

Status of retrieving location/IP coordinates or reverse geocoding.

Usually should be replaced in reverse results class.

property south

Return optional south coordinate of bbox, if available.

property southwest

Return south-west list of coordinates for bounds, if available.

property status

Specify current summary status of instance

property west

Return optional west coordinate of bbox, if available.

property wkt

Output coordinates in well-known text format, no SRID data.

property x

Longitude of the object

property xy

Optional list of longitude and latitude values.

property y

Latitude of the object

1.2.2 OpenStreetMap(Nominatim)

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.

Simple usage

OpenStreetMap does not require any keys for work. So you can begin without any setup.

```
import geocoder

g = geocoder.osm('New York city')
print(g.latlng)
# [40.7127281, -74.0060152]
print(g[0].latlng)
# Same as g.latlng: [40.7127281, -74.0060152]
```

This provider may return multiple results by setting the parameter `max_results` to the desired number. By default, 1 entry retrieved. Multiple results contained as internal sequence. You can check any result, by direct member object calling like in normal lists. Without member number mention, object with index 0 is always called.

```
import geocoder

g = geocoder.osm('New York city', max_results=3)
print(g[0].latlng)
# [40.7127281, -74.0060152]
print(g.latlng)
# Same as g[0].latlng: [40.7127281, -74.0060152]
print(g[1].latlng)
# Other result: [40.75126905, -73.98482021795536]
```

Custom or Local Nominatim Server

Setting up your own local offline Nominatim server is possible, using the following the Nominatim Install instructions. This enables you to request as much geocoding as your need.

Also, usage of any custom Nominatim Server is possible with setting `url` parameter. `url` should point to direct `/search` endpoint, check example below.

```
import geocoder

g = geocoder.osm("New York City", url="http://localhost/nominatim/search")
print(g[0].latlng)
# [40.7127281, -74.0060152]
```

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.

```
import geocoder
g = geocoder.osm('11 Wall Street, New York')
print(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
```

Helper method parameters

Helper method is recommended way to use providers, if no class extension required. During project modification this public API will be last thing for non-compatible changes.

`geocoder.osm`(*query*, *method='geocode'*, ***kwargs*)

OSM Provider

Provider supported methods:

- `geocode`
- `details`
- `reverse`

Parameters

- **`query`** – Your search location you want geocoded.
- **`method`** – One of provider's supported methods, defaults to `geocode`.
- **`url`** – Custom OSM Server URL location (ex: http://nominatim.openstreetmap.org/search)

Working class API

class `geocoder.providers.OsmQuery`(*location*, *url=None*, *key=None*, *timeout=None*, *proxies=None*, *session=None*, *headers=None*, *params=None*, ***kwargs*)

Bases: `geocoder.base.MultipleResultsQuery`

Nominatim API Reference: https://nominatim.org/release-docs/develop/api/Overview/

add(*value*)

Special method implementation for custom `MutableSequence` subclass

Not expected to be nested or changed in subclasses.

append(*value*)

S.append(value) – append value to the end of the sequence

clear() → None -- remove all items from S

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

debug()

Display debug information for instance of `MultipleResultsQuery`

extend(*values*)

S.extend(iterable) – extend sequence by appending elements from the iterable

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

insert(*index*, *value*)

Special method implementation for custom `MutableSequence` subclass

Not expected to be nested or changed in subclasses.

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

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

rate_limited_get(*url*, ***kwargs*)

By default, simply wraps a `requests.get()` request

remove(*value*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reverse()

S.reverse() – reverse *IN PLACE*

property geojson

Output all answers as GeoJSON FeatureCollection

property has_data

Status of geocoding if request was made

Raises `RuntimeError` – When external request was not made before property call

property status

Specify current summary status of instance

Possible statuses:

- "External request was not made"
- "OK" - when request was made, and any result retrieved
- `requests` error text representation, if request faced error
- "ERROR - No results found"
- "ERROR - Unhandled Exception"

Returned object properties

class geocoder.providers.**OsmResult**(*json_content*)

Bases: *geocoder.base.OneResult*

debug()

Display debug information for instance of `OneResult`

property accuracy

property address

Full comma-separated address

property allotments

place=allotments

Dacha or cottage settlement, which is located outside other inhabited locality. This value is used mainly in Russia and other countries of the former Soviet Union, where a lot of such unofficial settlements exist

property bbox

Output answer as GeoJSON bbox if it can be calculated/retrieved.

property bounds

Output answer as Google Maps API bounds if it can be calculated/retrieved.

property city

place=city

The largest urban settlements in the territory, normally including the national, state and provincial capitals. These are defined by charter or other governmental designation in some territories and are a matter of judgement in others. Should normally have a population of at least 100,000 people and be larger than nearby towns.

See place=suburb and place=neighbourhood on how to tag divisions within a city. The outskirts of urban settlements may or may not match the administratively declared boundary of the city.

property confidence

Is as a measure of how confident we are that centre point coordinates returned for the result precisely reflect the result.

property country

admin_level=2

property country_code

admin_level=2

property county

admin_level=6

property district

admin_level=5/6

property east

Return optional east coordinate of bbox, if available.

property farm

place=farm

A farm that has its own name. If the farm is not a part of bigger settlement use place=isolated_dwelling. See also landuse=farmyard

property geojson

Output answer as GeoJSON Feature

property geometry

Output answer as GeoJSON Point

property hamlet

place=hamlet

A smaller rural community typically with less than 100-200 inhabitants, few infrastructure.

property house_number

property icon

property importance

property island

place=island

Identifies the coastline of an island (> 1 km2), also consider place=islet for very small islandsIdentifies the coastline of an island (> 1 km2), also consider place=islet for very small islands

property isolated_dwelling

place=isolated_dwelling

Smallest kind of human settlement. No more than 2 households.

property lat

Latitude of the object

property latlng

Optional list of latitude and longitude values.

property license

property lng

Longitude of the object

property locality

place=isolated_dwelling

For an unpopulated named place.

property municipality

admin_level=8

property neighborhood

place=neighborhood

A named part of a place=village, a place=town or a place=city. Smaller than place=suburb and place=quarter.

The tag can be used for any kind of landuse or mix of landuse (such as residential, commercial, industrial etc). Usage of this term depends greatly on local history, culture, politics, economy and organization of settlements. More specific rules are intentionally avoided.

Note: the British English spelling is used rather than the American English spelling of neighborhood.

property north

Return optional north coordinate of bbox, if available.

property northeast

Return north-east list of coordinates for bounds, if available.

property ok

Status of retrieving location/IP coordinates or reverse geocoding.

Usually should be replaced in reverse results class.

property osm_id

property osm_type

property place_id

property place_rank

property population

property postal

property quality

property quarter

place=quarter

A named part of a bigger settlement where this part is smaller than a suburb and bigger than a neighbourhood. This does not have to be an administrative entity.

The term quarter is sometimes used synonymously for neighbourhood.

property region

admin_level=3

property south

Return optional south coordinate of bbox, if available.

property southwest

Return south-west list of coordinates for bounds, if available.

property state

admin_level=4

property status

Specify current summary status of instance

property street

property suburb

place=suburb

A distinct section of an urban settlement (city, town, etc.) with its own name and identity. e.g.

- annexed towns or villages which were formerly independent,
- independent (or dependent) municipalities within a city or next to a much bigger town
- historical districts of settlements
- industrial districts or recreation areas within a settlements with specific names.

property town

place=town

A second tier urban settlement of local importance, often with a population of 10,000 people and good range of local facilities including schools, medical facilities etc and traditionally a market. In areas of low population, towns may have significantly lower populations.

See place=neighbourhood and possibly also place=suburb on how to tag divisions within a town.

property type

property village

place=village

A smaller distinct settlement, smaller than a town with few facilities available with people traveling to nearby towns to access these. Populations of villages vary widely in different territories but will nearly always be less than 10,000 people, often a lot less.

See place=neighbourhood on how to tag divisions within a larger village

property west

Return optional west coordinate of bbox, if available.

property wkt

Output coordinates in well-known text format, no SRID data.

property x

Longitude of the object

property xy

Optional list of longitude and latitude values.

property y

Latitude of the object

References

- Nominatim Project
- Nominatim Install
- [addr tag]

CHAPTER TWO

CONTRIBUTOR GUIDE

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

2.1 Development documentation

Contents:

2.1.1 Geocoder to geocoder3 migration

During initial migration of geocoder project to geocoder3 many changes in structure and approach was made. Here is list of noticeable changes, that should be considered during migration from geocoder to geocoder3. Also, please read changes in described in project releases section.

Some parts of text below describe new provider's approach. This is valid only to providers, that are marked as **Geocoder3 ready** in main readme.md file.

Default geocoding engine changed from Google to OpenStreetMap

As Google engine now requires mandatory API key, default engine changed to free OpenStreetMap.

Some helper functions removed from `api.py` and project

Functions for 'silence' geocoding or IP geocoding are removed from project. Please use exact provider and provider configuration for such requests. Functions itself located in `api.py`.

List of removed functions:

- `geocoder.elevation()`
- `geocoder.nokia()` - now called as `geocoder.here()`
- `geocoder.location()` - replaced with `geocoder.Location`, direct class sharing
- `geocoder.places()`
- `geocoder.reverse()`
- `geocoder.timezone()`

Provider's files relocated

All provider's definition files relocated from project main folder to `providers` module and related subdirectory inside. Directories structure respect provider implementation function.

If you use direct provider classes imports, please update import statements. If you use direct helper's functions from `geocoder` module - no changes expected.

`geocoder.base.OneResult` changes

- Property `self.raw` renamed to `self.object_raw_json` to be more explainable in inside content. This change affects all subclasses (all providers).
- Property `self.housenumber` renamed to `self.house_number`. Affect all nested provider's files. Property `self.house_number` removed and available only in concrete provider's implementation.
- Most default properties values replaced from empty string `""` to `None`. Empty dicts in some cases left untouched. Please verify properties signatures.
- List of default properties become much smaller, some secondary properties, not required to internal *`geocoder.base.OneResult`* work was removed. This will allow new providers faster implementation. Such properties may exist in concrete implementations. Removed:
  - `accuracy`
  - `quality`
  - `house_number`
  - `street`
  - `city`
  - `state`
  - `country`
  - `postal`
  - `osm`
  - `locality`
  - `province`
  - `street_number`
  - `road`
  - `route`
- All parts of *`geocoder.base.OneResult`* now have huge docstrings and documentation, explaining all behaviour and approach.
- Some internal instance variables and properties renamed to be more concrete. This affect all children classes. List of renames:
  - `json` to `object_json`

`geocoder.base.MultipleResultsQuery` changes

- Class will enforce correct setting of `cls._URL`, `cls._RESULT_CLASS`, `cls._METHOD`, `cls._PROVIDER` in nested classes on project initialization stage.
- Non-mandatory class variables `cls.method` and `cls.provider` renamed to `cls._METHOD`, `cls._PROVIDER` and become mandatory, related tests added.
- Internal class structure changed. Now *`geocoder.base.MultipleResultsQuery.__init__()`* does not make an external query, and only do object initialization. This allow to initialize any amount of objects in advance usage cases (in loops). Query made in *`geocoder.base.MultipleResultsQuery.__call__()`* method. This change does not change helpers behaviour. I.e. `geocoder.get_results()` and related functions already respect this change internally.
- Some internal instance variables and properties renamed to be more concrete. This affect all children classes. List of renames:
  - `response` to `raw_response`
  - `_list` to `results_list`
  - `ok` to `has_data`
- New instance variables/properties added:
  - `is_called` - Hold status of external request. I.e. was or not was made.
  - `raw_json` - Hold unmodified JSON from provider for whole answer.
- Removed functions:
  - `geocoder.base.MultipleResultsQuery.set_default_result()`
- *`geocoder.base.MultipleResultsQuery.__init__()`* method now have all default keyword arguments in signature, removing silent usage of `kwargs.get("something")`, this practice will be extended to all child classes.
- All parts of *`geocoder.base.MultipleResultsQuery`* now have huge docstrings and documentation, explaining all behaviour and approach.

All print statements replaced with logging module

List of affected files, functions and classes:

- base.py
  - *`geocoder.base.OneResult.debug()`*
  - *`geocoder.base.MultipleResultsQuery.debug()`*
- distance.py
  - `geocoder.distance.haversine()` - warnings
- bing_batch.py
  - `geocoder.providers.BingBatchResult.debug()`
- bing_batch_forward.py
  - `geocoder.providers.BingBatchForwardResult.debug()`
- bing_batch_reverse.py
  - `geocoder.providers.BingBatchReverse.debug()`

Removed(some temporary) project features

- OSM type CLI/Debug output removed as non-well documented

kwargs approach and naming changes

- There was a confusion between 'deprecated' `limit` and new `maxRows` provider's setting. All such cases renamed to self-explained `max_results`.
- Everywhere, where it was possible all `**kwargs` replaced with complete list of function settings, usually with expected input type and defaults, if defaults available.

`geocoder.providers.google` classes renamed

- `geocoder.providers.ElevationQuery` to `geocoder.providers.GoogleElevationQuery`
- `geocoder.providers.PlacesQuery` to `geocoder.providers.GooglevPlacesQuery`
- `geocoder.providers.ElevationResult` to `geocoder.providers.GoogleElevationResult`
- `geocoder.providers.PlacesResult` to `geocoder.providers.GooglevPlacesResult`

2.1.2 Documentation guide

New documentation standards

1. All new documentation should be in `Markdown` format. Except index files and special cases.
2. However, usage of RST blocks is not restricted.

Update of old documentation guide

Old documentation files, partly located in `docs/source.old` folder. You can take any file from there and update it for new version.

1. First check is information is anyhow connected to `geocoder3` project, or completely absolute.
2. If information is completely outdated - remove related file, and make a PR with description of delete reason.
3. If information has sense, please update a file, convert it to `Markdown` and include in current documentation.

2.1.3 Testing policy

Testing policy requirements are strict and straightforward:

1. All providers files should be covered with tests.
2. All internet exchange (requests) should be pre-recorded with vcr.py and included in pull request. This guarantee that tests are connection independent.
3. Main test engine is pytest.

2.1.4 Geocoder authors

Honored crew of authors of original library.

Lead Developer

- Denis Carriere - Creator of Python's Geocoder

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 compatibility
- flebel - Submitted Github Issues
- patrickyan - Submitted Github Issues
- esy - Submitted Github Issues
- Sergei Grabalin ( ) - Fixed Python2 Unicode Issues
- Matthieu Rigal - Added session support

2.1.5 Geocoder3 authors

Lead developer

- Andrey Shpak - Lead of python3 migration project

Contributors

Your name can be here…

INDEX