googlemaps - Google Maps and Local Search APIs

class **GoogleMaps**

An easy-to-use Python wrapper for the Google Maps and Local Search APIs.

Geocoding: convert a postal address to latitude and longitude

```
>>> from googlemaps import GoogleMaps
>>> gmaps = GoogleMaps(api_key)
>>> address = 'Constitution Ave NW & 10th St NW, Washington, DC'
>>> lat, lng = gmaps.address_to_latlng(address)
>>> print lat, lng
38.8921021 -77.0260358
```

Reverse Geocoding: find the nearest address to (lat, lng)

```
>>> destination = gmaps.latlng_to_address(38.887563, -77.019929)
>>> print destination
Independence and 6th SW, Washington, DC 20024, USA
```

Local Search: find places matching a query near a given location

```
>>> local = gmaps.local_search('cafe near ' + destination)
>>> print local['responseData']['results'][0]['titleNoFormatting']
Vie De France Bakery & Cafe
```

Directions: turn-by-turn directions, distance, time, etc. from point A to point B

```
>>> directions = gmaps.directions(address, destination)
>>> print directions['Directions']['Distance']['meters']
1029
>>> print directions['Directions']['Duration']['seconds']
106
>>> for step in directions['Directions']['Routes'][0]['Steps']:
...     print step['descriptionHtml']
Head <b>east</b> on <b>Constitution Ave NW</b> toward <b>9th St NW</b>
Take the 2nd <b>right</b> onto <b>7th St NW</b>
Turn <b>left</b> at <b>Independence Ave SW</b>
```

This software is in no way associated with or endorsed by Google Inc. Use of the Google
Terms of Service: http://code.google.com/apis/maps/terms.html. Note in particular that yo
Maps API key to use this service, and that there are rate limits to the number of requests yo

GoogleMaps methods

GoogleMaps.**__init__**(*api_key=''*, *referrer_url=''*)

Create a new **GoogleMaps** object using the given *api_key* and *referrer_url*.

Parameters:
- *api_key* (string) – Google Maps API key
- *referrer_url* (string) – URL of the website using or displaying information fro

Google requires API users to register for an API key before using the geocoding s
http://code.google.com/apis/maps/signup.html. If you are not geocoding, you do not need ar

Use of Google Local Search requires a referrer URL, generally the website where the retri
If you are not using Local Search, you do not need a referrer URL.

Geocoding

GoogleMaps.**address_to_latlng**(*address*)

Given a string *address*, return a *(latitude, longitude)* pair.

This is a simplified wrapper for **geocode()**.

Parameter: *address* (string) – The postal address to geocode.
Returns: *(latitude, longitude)* of *address*.
Return type: (float, float)
Raises GoogleMapsError:

If the address could not be geocoded.

GoogleMaps.**geocode**(*query*, *sensor='false'*, *oe='utf8'*, *ll=''*, *spn=''*, *gl=''*)

Given a string address *query*, return a dictionary of information about that location, including

Interesting bits:

```
>>> gmaps = GoogleMaps(api_key)
>>> address = '350 Fifth Avenue New York, NY'
>>> result = gmaps.geocode(address)
>>> placemark = result['Placemark'][0]
>>> lng, lat = placemark['Point']['coordinates'][0:2]    # Note these are backwards
>>> print lat, lng
40.6721118 -73.9838823
>>> details = placemark['AddressDetails']['Country']['AdministrativeArea']
>>> street = details['Locality']['Thoroughfare']['ThoroughfareName']
>>> city = details['Locality']['LocalityName']
>>> state = details['AdministrativeAreaName']
>>> zipcode = details['Locality']['PostalCode']['PostalCodeNumber']
>>> print ', '.join((street, city, state, zipcode))
350 5th Ave, Brooklyn, NY, 11215
```

More documentation on the format of the return value can be found at Google's geocoder
Some places have a *'SubAdministrativeArea'* and some don't; sometimes a *'Locality'* will ha
some don't.)

Parameters:
- *query* (string) – Address of location to be geocoded.
- *sensor* (string) – `'true'` if the address is coming from, say, a GPS device.
- *oe* (string) – Output Encoding; best left at `'utf8'`.
- *ll* (string) – *lat,lng* of the viewport center as comma-separated string; Viewport Biasing
- *spn* (string) – The "span" of the viewport; must be used with *ll*.
- *gl* (string) – Two-character ccTLD for country code biasing.

Returns: geocoder return value dictionary

Return type: dict

Raises GoogleMapsError:
if there is something wrong with the query.

More information on the types and meaning of the parameters can be found at the Google H

Reverse Geocoding

GoogleMaps.**latlng_to_address**(*lat*, *lng*)

Given a latitude *lat* and longitude *lng*, return the closest address.

This is a simplified wrapper for `reverse_geocode()`.

Parameters:
- *lat* (float) – latitude
- *lng* (float) – longitude

Returns: Closest postal address to *(lat, lng)*, if any.

Return type: string

Raises GoogleMapsError:
if the coordinates could not be converted to an address.

GoogleMaps.**reverse_geocode**(*lat*, *lng*, *sensor='false'*, *oe='utf8'*, *ll=''*, *spn=''*, *gl=''*)

Converts a (latitude, longitude) pair to an address.

Interesting bits:

```
>>> gmaps = GoogleMaps(api_key)
>>> reverse = gmaps.reverse_geocode(38.887563, -77.019929)
>>> address = reverse['Placemark'][0]['address']
>>> print address
Independence and 6th SW, Washington, DC 20024, USA
>>> accuracy = reverse['Placemark'][0]['AddressDetails']['Accuracy']
>>> print accuracy
9
```

Parameters:
- *lat* (float) – latitude
- *lng* (float) – longitude

Returns: Reverse geocoder return value dictionary giving closest address(es) to *(lat, lr*

Return type: dict

Raises GoogleMapsError:

If the coordinates could not be reverse geocoded.

Keyword arguments and return value are identical to those of `geocode()`.

Directions

`GoogleMaps.`**`directions`**(*origin*, *destination*, ***kwargs*)

Get driving directions from *origin* to *destination*.

Interesting bits:

```
>>> gmaps = GoogleMaps(api_key)
>>> start = 'Constitution Ave NW & 10th St NW, Washington, DC'
>>> end   = 'Independence and 6th SW, Washington, DC 20024, USA'
>>> dirs  = gmaps.directions(start, end)
>>> time  = dirs['Directions']['Duration']['seconds']
>>> dist  = dirs['Directions']['Distance']['meters']
>>> route = dirs['Directions']['Routes'][0]
>>> for step in route['Steps']:
...     print step['Point']['coordinates'][1], step['Point']['coordinates'][0]
...     print step['descriptionHtml']
38.8921 -77.02604
Head <b>east</b> on <b>Constitution Ave NW</b> toward <b>9th St NW</b>
38.89208 -77.02191
Take the 2nd <b>right</b> onto <b>7th St NW</b>
38.88757 -77.02191
Turn <b>left</b> at <b>Independence Ave SW</b>
```

Parameters:
- *origin* (string) – Starting address
- *destination* (string) – Ending address
- *kwargs* – You can pass additional URL parameters as keyword arguments documented.

Returns: Dictionary containing driving directions.

Return type: dict

Raises GoogleMapsError:

If Google Maps was unable to find directions.

Local Search

`GoogleMaps.`**`local_search`**(*query*, *numresults=8*, ***kwargs*)

Searches Google Local for the string *query* and returns a dictionary of the results.

```
>>> gmaps = GoogleMaps(api_key)
>>> local = gmaps.local_search('sushi san francisco, ca')
>>> result = local['responseData']['results'][0]
>>> print result['titleNoFormatting']
Sushi Groove
>>> print result['streetAddress']
1916 Hyde St
>>> print result['phoneNumbers'][0]['number']
(415) 440-1905
```

For more information on the available data, see Google's documentation on AJAX res properties.

The return value of this method is slightly different than that documented by Google; it atte as possible, from several queries (up to *numresults*), into the `['responseData']['results']` a results referencing this array (such as `'cursor'`, `'currentPageIndex'`, `'moreResultsUrl'`) may n

This method may return fewer results than you ask for; Google Local returns a maximum of results.

Parameters:
- *query* (string) – String containing a search and a location, such as `'Sushi Sa`

- *numresults* (int) – Number of results to return, up to a maximum of **MAX_LOCA**
- *kwargs* – You can pass additional AJAX search arguments and they will be

Returns: A Google AJAX result structure.

Return type: dict

Raises GoogleMapsError:

If the query was malformed.

Installation

It's as easy as:

```
sudo easy_install googlemaps
```

For Python versions prior to 2.6, you may also need the simplejson module.

Not got root? **googlemaps** plays nice with virtualenv.

You can also download the source from sourceforge.net; `googlemaps.py` packs all this delicious f contained module that can be used as a script for command-line geocoding.

easy_install is available from PyPI if you don't have it already.

Notes

You will need your own Google Maps API key to use the geocoding functions of this module number of requests per day from a single IP address. If you make too many requests, or yo something else is wrong with your request, a **GoogleMapsError** will be raised containing a status the error; more information can be found at the linked reference.

All of the data returned by this module is in JSON-compatible format, making it easy to combine

Information

Author: John Kleint
Version: 1.0.2
License: Lesser Affero General Public License v3
Source: http://sourceforge.net/projects/py-googlemaps
Python Versions:
2.3 - 2.6+

This software comes with no warranty and is in no way associated with Google Inc. or the Goo Google does not approve or endorse this software. GOOGLE is a trademark of Google Inc.