postcodes Documentation

Release 0.1

Edward Robinson

December 04, 2016

Contents

Postcodes is a small library for getting information about, postcodes in the UK. At its core, the postcode data is provided by the Ordnance Survey OS OpenData initiative, but this library is actually a wrapper for a web-service provided by Stuart Harrison.

CHAPTER 1

Installation

If you use pip then installation is simply:

```
$ pip install postcodes
```

or, if you want the latest github version:

```
$ pip install git+git://github.com/e-dard/postcodes.git
```

You can also install Postcodes via Easy Install:

```
$ easy_install postcodes
```

CHAPTER 2

Features

Postcodes allows you to do the following:

- Lookup the postcode data associated with a specific postcode;
- Get the nearest postcode data associated to a specific geographical point;
- Get all of the postcode data within a specific distance to a geographical point;
- Get all of the postcode data within a specific distance to a known postcode.

As well as being a thin wrapper over the uk-postcodes web-service, Postcodes also provides a simple caching and validation layer, in the form of the *PostCoder* object, meaning you don't have to worry about keeping track of any previously requested data.

2.1 Usage

Postcodes is very simple. Simply create a new *PostCoder* object and away you go:

```
>>> from pprint import PrettyPrinter
>>> from postcodes import PostCoder
>>>
>>> pc = PostCoder()
>>> result = pc.get("SW1A 2TT")
>>> PrettyPrinter(indent=4).pprint(result['geo'])
{   u'easting': u'530283',
    u'geohash': u'http://geohash.org/gcpuvptqwyh4',
    u'lat': u'51.502308',
    u'lng': u'-0.124331',
    u'northing': u'179820'}
>>>
```

If for any reason you want to use your own caching or validation, you also have access to the functions in the *postcodes* module, which are also documented in the *API* section.

2.2 Returned Data

For each postcode, a Python dictionary is returned containing all the available data from the Ordanance Survey Code-Point Open dataset. For example, `postcodes.get("W1A 2TT")` returns:

```
{ u'administrative': { u'constituency': { u'code': u'',
                                          u'title': u'',
                                          u'uri': u''},
                      u'district': { u'snac': u'',
                                     u'title': u'',
                                     u'uri': u''},
                      u'ward': { u'snac': u'',
                                 u'title': u'',
                                 u'uri': u''}},
  u'geo': { u'easting': u'',
            u'geohash': u'',
            u'lat': u'',
            u'lng': u'',
            u'northing': u''},
  u'postcode': u''}
```

Values have been removed for brevity; all returned types are `unicode` stings.

CHAPTER 3

API Documentation

postcodes.**get**(*postcode*)
Request data associated with postcode.

Parameters **postcode** – the postcode to search for. The postcode may contain spaces (they will be removed).

Returns a dict of the nearest postcode's data or None if no postcode data is found.

postcodes.**get_nearest**(*lat*, *lng*)
Request the nearest postcode to a geographical point, specified by lat and lng.

Parameters

- **lat** – latitude of point.
- **lng** – longitude of point.

Returns a dict of the nearest postcode's data.

postcodes.**get_from_postcode**(*postcode*, *distance*)
Request all postcode data within distance miles of postcode.

Parameters

- **postcode** – the postcode to search for. The postcode may contain spaces (they will be removed).
- **distance** – distance in miles to postcode.

Returns a list of dicts containing postcode data within the specified distance or None if postcode is not valid.

postcodes.**get_from_geo**(*lat*, *lng*, *distance*)
Request all postcode data within distance miles of a geographical point specified by lat and lng.

Parameters

- **lat** – latitude of point.
- **lng** – longitude of point.
- **distance** – distance in miles to postcode.

Returns a list of dicts containing postcode data within the specified distance.

3.1 The PostCoder Object

class postcodes.**PostCoder**

The *PostCoder* object provides state for maintaining a cache of historical requests. It's the recommended way to interact with the underlying web-service.

Because *PostCoder* caches all previously requested postcode data it's fine to repeatedly request the same data as much as you like, and you don't need to worry about explicitly storing any data in your application.

Because the underlying data is not likely to change very much, if at all, cached postcode data never expires. However, if for some perverse reason you do want to skip the cache and make an explicit request for data then you can set `skip_cache=True` in all of the available methods.

get(*postcode*, *skip_cache=False*)

Calls *postcodes.get* and by default utilises a local cache.

Parameters **skip_cache** – optional argument specifying whether to skip the cache and make an explicit request. Given postcode data doesn't really change, it's unlikely you will ever want to set this to `True`.

get_nearest(*lat*, *lng*, *skip_cache=False*)

Calls *postcodes.get_nearest* but checks correctness of `lat` and `long`, and by default utilises a local cache.

Parameters **skip_cache** – optional argument specifying whether to skip the cache and make an explicit request.

Raises **IllegalPointException** – if the latitude or longitude are out of bounds.

Returns a dict of the nearest postcode's data.

get_from_postcode(*postcode*, *distance*, *skip_cache=False*)

Calls *postcodes.get_from_postcode* but checks correctness of `distance`, and by default utilises a local cache.

Parameters **skip_cache** – optional argument specifying whether to skip the cache and make an explicit request.

Raises **IllegalPointException** – if the latitude or longitude are out of bounds.

Returns a list of dicts containing postcode data within the specified distance.

get_from_geo(*lat*, *lng*, *distance*, *skip_cache=False*)

Calls *postcodes.get_from_geo* but checks the correctness of all arguments, and by default utilises a local cache.

Parameters **skip_cache** – optional argument specifying whether to skip the cache and make an explicit request.

Raises **IllegalPointException** – if the latitude or longitude are out of bounds.

Returns a list of dicts containing postcode data within the specified distance.

p

G

P