SmartyStreets.py Documentation

Release 0.1.0

Ben Lopatin

October 21, 2014

Contents:

CHAPTER 1

SmartyStreets.py

A wrapper for the SmartyStreets address validation and geolocation API.

Other Python libraries exist but skip out on multiple address submission and make opinionated decisions about how to transform the return data.

Note: This project is not affiliated with the SmartyStreets service or company in any way.

- Free software: BSD license
- Documentation: https://smartystreets.readthedocs.org.

1.1 Features

SmartyStreets.py aims to provide a sane, tested, and feature complete wrapper to the SmartyStreets LiveAddress API, including address lookup for validation and geolocation, as well as zipcode lookup and validation.

1.2 Installation

SmartyStreets.py requires the requests library and will install it if it is found missing. **Installed versions < 2.0 will be upgraded.**:

```
pip install smartystreets.py
```

1.3 Basic usage

1.3.1 API client

Create a client instance with your key:

```
client = SmartyStreets(AUTH_ID, AUTH_TOKEN)
```

Create a client instance with SmartyStreets configuration options:

```
client = SmartyStreets(AUTH_ID, AUTH_TOKEN, standardize=True, invalid=False,
            logging=False)
```

These options correspond to the *x-standardize-only*, *x-include-invalid*, and *x-suppress-logging* headers for opening up results to standardized but not necessarily deliverable addresses, including invalid delivery addresses, and toggling SmartyStreets API logging, respectively.

Since the SmartyStreets API only permits up to 100 addresses to be looked up at once the client will raise an exception if more than 100 are provided. You can turn off this functionality using the *truncate_addresses* option, which will silently truncate the list to the first 100 addresses:

```
client = SmartyStreets(AUTH_ID, AUTH_TOKEN, truncate_addresses=True)
```

1.3.2 Address lookup

Simple address lookup:

```
client.street_address("100 Main St Richmond, VA")
```

Multiple simple street addresses:

```
client.street_addresses(["100 Main St Richmond, VA", "100 Main St Richmond, VA"])
```

Note that these are different function names.

You can also use dictionaries including detailed data:

```
client.street_address({
    'input_id': 'k1d8j',
    'street': '100 Main st',
    'city': 'Richmond',
    'state': 'VA',
    'candidates': 2,
})
```

And multiple detailed lookups:

```
client.street_addresses([
    {
        'input_id': 'k1d8j',
        'street': '100 Main st',
        'city': 'Richmond',
        'state': 'VA',
        'candidates': 2,
    }, {
        'input_id': 'z29ir',
        'street': '400 Main st',
        'city': 'Richmond',
        'state': 'VA',
        'candidates': 2,
    }
])
```

Note: You cannot mix the simple street lookup style and the detailed dictionary lookup style in the same API call. This is a library restriction.

1.3.3 Return data

Just as important as a clean interface for working with the API is a helpful way of working with the returned data.

Returned data is presented as either a single *SmartyAddress* or a *SmartyAddresses* collection. Each is based on builtin types so that you always have access to the underlying data exactly as it was returned, but with added convenience methods.

Address geolocation

```
>>> address = client.street_address("100 Main St Richmond, VA")
>>> address.location
(37.5436,-77.4453)
```

Address verification

```
>>> address.verified
True
```

Multiple addresses: input ID lookup

You can look up an address by the *input_id* parameter (provided you include one in the request):

```
>>> addresses = client.street_address([{'input_id': '123', 'street': ...}])
>>> addresses.get('123')
{'input_id': '123', 'street': ... }
```

The *get* method is used because the *SmartyAddresses* object's default lookup is against the list index.

1.3.4 Zipcode lookup

TODO

1.3.5 Response errors

The following documented response codes raise specific exceptions based on a *SmaryStreetsError* class.

- 400 Bad input. Required fields missing from input or are malformed.
- 401 Unauthorized. Addressuthentication failure; invalid credentials.
- 402 Payment required. No Addressuthenticationctive subscription found.
- 500 Internal server error. General service foundailure; retry request.

CHAPTER 2

Installation

At the command line:

```
$ pip install smartystreets.py
```

This will install the package *smartystreets* into your Python path, along with requests if it is not installed already, or the previously installed version is < 2.0.

Usage

To use SmartyStreets.py in a project:

```
from smartystreets.client import Client
```

You'll need to provide your *AUTH_ID* and *AUTH_TOKEN* to create a client instance:

```
myclient = Client(auth_id='jkjdakjdkfjkaj', auth_token='kjakj1kjd')
```

You'll use either the *street_address* or *street_addresses* method depending on *what kind of data you want back*. The first method takes only one address as an argument, and returns either a single *Address* instance or *None*. The pluralized *street_addresses* will take a list of 1 or more address inputs and return an *AddressCollection* regardless of how many addresses are submitted or returned.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/bennylope/smartystreets/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

SmartyStreets.py could always use more documentation, whether as part of the official SmartyStreets.py docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bennylope/smartystreets/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *smartystreets* for local development.

1. Fork the *smartystreets* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/smartystreets.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv smartystreets
   $ cd smartystreets/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 smartystreets tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/bennylope/smartystreets/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_smartystreets
```

CHAPTER 5

Credits

5.1 Development Lead

- Ben Lopatin <ben@wellfire.co>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.0 (2014-10-30)

- First release on PyPI.

CHAPTER 8

Indices and tables

- *genindex*
- *modindex*
- *search*