MPHYG001: Research Software Engineering with Python

Assessment 1: Packaging Greengraph

Jan Zmazek, 16105492

January 5, 2017

1 Usage of the entry point

This Python package uses Google Maps to plot "urbanisation" (green pixels intensity) between two specified locations. The program accepts four input arguments; first location name, second location name, number of steps between locations and name of the output file.

1.1 Installation

The package can be installed using pip:

```
$ pip install git+git://github.com/janzmazek/greengraph.git
```

or if you want a local copy of the repository

```
$ git clone https://github.com/janzmazek/greengraph.git
$ python setup.py install
```

1.2 Usage

The package can be run using the command line with command

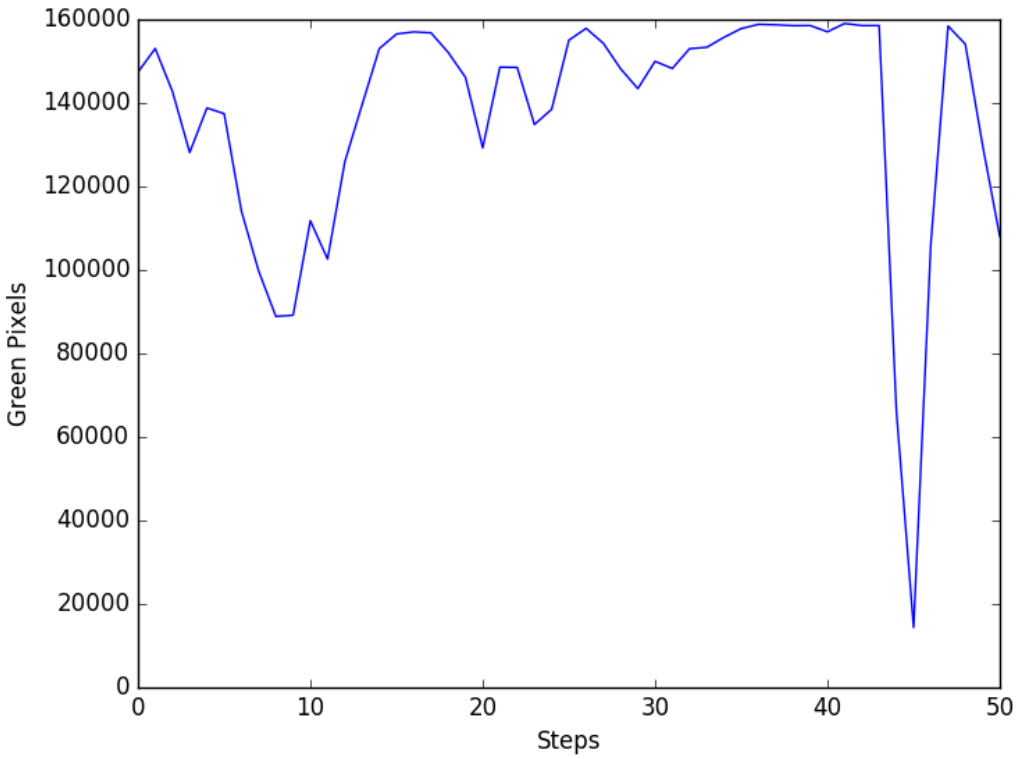
```
$ greengraph.
```

To specify input arguments, use

- `--from` (or `-f`): starting location,
- `--to` (or `-t`): ending location,
- `--steps` (or `-s`): number of steps between the locations,
- `--out` or (`-o`): name of the output file.

Example:

```
$ greengraph -f 'Ljubljana' -t 'London' -s 50 -o 'plot.png'
```



2 Discussion

As the main references for this assignment I used:

1. lecture notes (`http://github-pages.ucl.ac.uk/rsd-engineeringcourse/`),
2. tutorial on Python packaging (`https://packaging.python.org/`),
3. `unittest` documentation (`https://docs.python.org/3/library/unittest.html`).

The first problem that I encountered was regarding the use of external libraries. Getting code to work I had to replace library `StringIO` (which is gone in Python 3) with the library `io`. I also had to import library `requests` that was missing in the appended code for an unknown reason.

When writing tests, I encountered a problem when asserting two arrays are equal. One solution would be using `numpy.testing` library. But since I did not want to mix testing environments, I solved this by asserting values element by element, which is not the best solution as it is not very fast.

The extensive preparation of the work for release is in my opinion advantageous. Detailed documentation and clear examples of use give users a better insight of what the code does and increases the probability of users to download and use the program. Writing comments and understandable code (especially sensible variables) are especially good for developers who want to contribute to your code. Although testing the code can be quite time consuming, I think it is important, especially in bigger projects with large amounts of code.

Using package managers like `pip` makes installing a package very easy. I believe that this is especially advantageous for less experienced users as Python packages are installed or uninstalled using just the commands:

```
$ pip install some-package-name
$ pip uninstall some-package-name
```

`pip` installation also checks and installs dependencies. Another reason for using `pip` is that you can check for updates on all installed packages and update all of them automatically.

I think the advantages of using package indexes like `PyPI` are similar. It makes the documentation and description of packages easily accessible on website `pypi.python.org`, so packages are easy to find. Having a package on `PyPI` website probably makes users trust the code and thus installing it.

If we want some package to be maintained and updated, we often want to build a community of contributors and users. To achieve this, the code should be useful and well written in my opinion, which can be achieved, as mentioned earlier, by writing good comments and a clear syntax. The main contributor should be available for questions, respond to comments and revise and approve pull requests.