MPHYG001: Coursework #1

Due on Monday, January 11, 2016

Maria Ruxandra Robu

Greengraph is a Python package that plots the distribution of green pixels found in the maps between two given locations. For a detailed description of the main functions, refer to the documentation files generated with Sphinx. The package can be installed from a git repository and used from the command line or as a library. For more details on the usage, please view the README file.

Task 1

Document the usage of your entry point.

The command line entry point is of the form:

```
$ getGreenGraph --from startPosition --to endPosition --steps numberSteps --out
    filenameOutput.ext
# example of usage
$ getGreenGraph --from London --to Cambridge --steps 10 --out outGraph.png
```

For more details on how to use the command line and to view the default values for the arguments, try:

```
$ getGreenGraph --help
usage: getGreenGraph [-h] [--from STARTPOS] [--to ENDPOS] [--steps STEPS]
                [--out OUT]

Greengraph package - Generates a graph with the proportion of green pixels
between two locations

optional arguments:
  -h, --help   show this help message and exit
  --from STARTPOS Start position, default = London
  --to ENDPOS  End position, default = Cambridge
  --steps STEPS Number of steps between start and end, default = 10
  --out OUT    Name of output image, default = outGraph.png
```

Similarly, the package can be used as a library in any Python module. Here is an example of a call to get a graph of the distribution of green pixels between two given locations:

```
greengraph.getGraph.plotGreenDistribution('London', 'Cambridge', 10, 'outGraph.png')
```

For more details, please see the documentation of the code generated with Sphinx.

Task 2

Discuss problems encountered in completing your work.

Most of the difficulties were encountered in the testing section, in understanding how to use fixtures and mocks and how to apply them to this project.

I tried to patch the htttp get request to the internet in the Map() class. In the original code, the return value is used in the constructor to save the image and get the numpy array with the pixels. I could not figure out how to set up the return_value in such a way that the constructor does not throw an exception when it tries to load the image. So, I created a new function outside of the class (getMapAt()), which returns the response

Task 2 continued on next page...

from the request.get. The function is called only from the constructor of the class Map(). In this case, the patch worked and it tests the arguments with which it is called in test_mapClass.test_defaultRequestParam().

On the other hand, creating a mock for the Greengraph() class did not present any problems, since the connection to the internet was established in a method (geolocate(self, place)) and not in the constructor. In this test case (test_graphClass.test_geolocate()), a MagicMock() was created for the class member geocoder.

I encountered another issue when I wanted to test if the method count_green() from the Map() class works with an image that has no green pixels. When a Map object is created, the image of the current latitude and longitude is set from a connection to Google Maps. I wanted to create a mock for the class, so the constructor does not send the get request. I tried to write a test function with a patch decorator like this:

```
from mock import patch
@patch('greengraph.map.Map')
def getMapMock(mockMap):
  # patch decorator
  mockMap.pixels=someTestImage
   numGreenPix=mockMap.count_green()
```

However, I could not make it work. So, I added the method setDummyImage(self, testImg) to the class Map(). In this way, the count_green() method was tested for an image with no green pixels. The test image was generated using fixtures. A setup_module(module) function creates a image with ones on the red and blue channels and zeros on the green one. This image is saved in .csv files and loaded in the necessary test files. A teardown_module(module) function was also added, which can delete the .csv files after all the functions in the testing module have been run. Normally, these setup_module and teardown_module function are used for opening and closing database and internet connections. However, their use was adapted to the Greengraph project in this situation.

Task 3

Discuss the advantages and costs involved in preparing work for release, the use of package managers like pip and package indexes like PyPI.

Package indexes, like PyPi ease the distribution of code within a community of developers. These packages can be easily managed through the use of pip from the command line. Given these well established distribution tools, most developers would start searching for libraries in there. So, a clear advantage of deploying a package on PyPi would be the increased visibility and the potential greater reach. Furthermore, it makes the installation with the package manager pip easier.

```
# package from a random website
$ pip install https://myWebsite.com/PythonCode/myPackages/myProject301.tar
# package from GitHub
$ pip install git+https://github.com/myUsername/myProject301.git
# package from PyPi
$ pip install myProject301
```

In order to deploy a package on PyPi, a setup.py file has to be defined. Similarly, files like the README.rst, LICENSE.txt and code documentation are recommended since they make the usage of the code easier. These files are considered best practice for releasing packages that are easy to integrate in any project. Hence,

Task 3 continued on next page...

through its extensive use in the Python developers community, the code uploaded on PyPi has a greater chance of being well documented.

One of the disadvantages of code distributed through these methods is that the developers have to have some experience with the tools, how to install the packages and if necessary, the correct versions of the dependencies. Consequently, PyPi and pip would not be the ideal choices when packages need to be deployed to end-users (in which case an executable would be needed).

Task 4

Discuss further steps you would need to take to build a community of users for a project.

One of the most important parts in building a community of users for a project is to document the code thoroughly. This would allow developers to easily install and use the package. Furthermore, it will encourage further development on top of the existing code, if the modules are designed to allow extensions.

Another important step would be for the main developers to be active in resolving issues in reasonable time. Maintaining the package up to date is essential in keeping the existing users and make the community stronger.

Moreover, the main contributors to the package should be active on websites like Stack Overflow where developers ask questions related to their product. In this way, they can increase their visibility and bring new people in the community.