GreenGraph Coursework Report

Michael Vasmer
SN: 110003490
Module Code: MPHYG001

Introduction

This report is split into four sections: documentation of my entry point script, a discussion of the problems I encountered, a discussion of the pros and cons of packaging python projects and some ideas about building a community of users for the project.

Entry point script

The usage of my entry point script is as follows:
`greengraph [--help] [--start START] [--end END] [--steps STEPS] [--out OUT]`
The `--help` flag prints out the usage instructions. The `--start` flag allows the user to specify the starting location and the `--end` flag allows the user to specify the ending location. These default to London and Durham respectively. The `--steps` flag allows the user to specify the steps between the starting and ending locations (i.e. the number of data points on the graph). This defaults to 10. The `--out` flag allows the user to specify the output file name, which is saved as a Portable Network Graphics (PNG) file. This defaults to `graph.png`.
Example: `greengraph --start London --end Birmingham --steps 10 --out LonBir`
This would produce a file `LonBir.png` in the directory where the entry point script was invoked containing the graph.

Problems encountered whilst completing the coursework

I encountered two main problems whilst completing the coursework. The first of these was to do with testing if my packaged code was installing correctly. The first time I ran `python setup.py install` some of my other installed packages broke and I found it very difficult to remove all of the files for the greengraph package as I wasn't sure where they had been installed. I managed to avoid this problem after doing some research and using Virtual Environments. By using Virtual Environments I was able to test the installation of my package in various different contexts and if any problems occurred I could always just delete the Virtual Environment safe in the knowledge that my main Python installation was untouched.

The second problem I encountered was understanding mocking. Mocking is a concept I had not encountered before and I found it hard to grasp initially. The single biggest problem I had was in programming the `test_green_between` method in `test_graph.py`. I wanted to keep track of how the different `Map` instances were constructed in the green_between method. This would allow me to test that the `Map` instances had been constructed correctly. However I found it difficult to envisage doing this. Eventually I realised that I needed to mock the `__init__` method of the `Map` class. But this still produced an error until I made the mock of the `__init__` method return `None` as an `__init__` method is meant to. After taking these steps I was able to implement the test as I wanted to.

Advantages and disadvantages of packaging Python projects

There are advantages and disadvantages of packaging Python projects. The main advantage of packaging a Python project is that this allows other people to install and use it. This is useful for the Python community if the project provides a useful function. But this is also useful for the author of the package as more people using it will enable the author to become more well known. This advantage will hopefully be enhanced by the proper citation of software as dictated in CITATION files. Another advantage of packaging a Python project is that if you make it open source then it is

possible that a community will build up around the project. The community may spot bugs and further develop the project which alleviates work for the original developer. It is also likely that a project on something like github will have a longer lifetime and higher relevancy.

The disadvantage of packaging a project is the additional work this entails. To package a project additional files must be created and more testing carried out. However I think that the benefits far outweigh the extra work required initially.

The use of package managers like pip is to be recommended in my opinion. When installing a package using the `python setup.py install` it can be difficult to keep track of where the files are installed to. This can make uninstalling a package very difficult unless the user understands how their Python installation works in detail. However to uninstall a package installed with pip one only needs to run `pip uninstall <package>`. However there is a disadvantage associated with using pip in conjunction with a package index like PyPI. Package indices do not always have the most up to date versions of their listed packages which can sometimes be a problem. Also newer packages may take time to appear on a package index. But the advantage of a package index is that the user can be confident that the packages on it work and are well tested. The ability to use pip to install a github Python package is highly useful in this context as the user can then install a newer more experimental package and uninstall it easily if it doesn't work or causes problems.

Building a community

To build a community of active users for this project a number of steps would need to be taken. Firstly I would talk to other academics at UCL, people in my research area and in the UCL research software hub. I would ask their advice about building a community and see if they wanted to be part of the project. Outside of UCL I would also try to talk to other academics at conferences or on forums.

Outside of the academic community I would try and publicise the project through github and build an active community of users. Github is the ideal place to do this as the users can immediately begin contributing to the project, giving them a sense of ownership and an interest in seeing the project succeed.