Packaging Greengraph
Assessment 1
MPHYG001

Leo Carlos-Sandberg
16101556

28/12/2016

Design Choices and Implementation

This project has been written in python 3 due to the fact that this is the version installed on my computer and it seemed right to use the newest version.
I have chosen to split my code up into three primary files: map, graph and greengraph. Each file only deals with one major aspect of the code, allowing there to be a structure. This split is enough to make the code more readable and manageable but not so much that other files have to be constantly referred to in understanding the code.
I removed the definition for show_green as it was not called any where in the code so did nothing more then confuse the code.
The entry point was designed to have a default setting, this was to allow quick calling and to act as a alternative to simply throwing an error if a field was not filled in.
I also decided to display the plot with or without the addition of the --out call so that it can be seen even if you do not wish to save it as an image.
The commit statements were given a version number x.y.z, where x represented large changes to the code such as a new interface or a change in the codes end operation. Where y represented a smaller change such as new files being added such as tests and z represented much smaller changes, such as bug fixes or adding new elements to preexisting files.

Entry Point

The command line entry point was made to resemble the one shown in the assignment and can be called on the command line after installation as follows:
greengraph --from “start” --to “end” --steps “steps” --out “file name”
There is also another argument that is -h, which will display the help for this call and can be called as:
greengraph -h
An example call is:
greengraph --from London --to Oxford --steps 10 --out graph.png
This will calculate the number of green pixels at 10 evenly spaced steps between London and Oxford and then save this plot as graph.png as well as displaying it.

Problems

The majority of the problems that I dealt with during this project were down to the fact I had to convert from python 2 into python 3. This lead to some error messages which were very difficult to understand.
I also had some difficulty setting my files up to be pip installed, as I had not done this before, especially when it came to the extra files that were needed within the folders.
In my tests I had some trouble calling my fixtures, unless they were in the same file, this probably was intermittent.
Also it was my first time using mocks, which I found difficult and confusing sometimes.

Preparing Work for Release

Preparing work for release involves packaging it, creating extra files, tests and uploading it to somewhere that it can be installed from. This work, especially the tests, can lead to extra development time for the project and extra work for the developer. This can seem like a larger disadvantage to any developer, however the processes can be very beneficial, making code more robust to future changes by using tests, making it properly documented with supplementary files and saving it to a more secure external location in case of corruption.
Also once code is packaged like this, it can be used by others, helping to increase your renown and citations as well as simply adding to the scientific community. Packaging for pip installation goes a ways towards this goal allowing the code to be easily installed and then again uninstalled, making it more appealing for other users. Using package indexes like PyPi is good, as you know the code is robust and has been tested, however parts maybe more outdated and new packages may take a while to appear due to the submission processes. However newer packages can be easily downloaded from github using pip install, there is less quality guaranties but there is much faster accesses to new packages.

Building a Community

The first step of building a community would be uploading the project to github, which makes it available everyone and allows contributions to it in the form of additional features, bug reporting and fixing etc. When sufficient testing has been done the project could be submitted to PyPi, which would increase the trust in the quality of the project. Finally it could be added to a package management system say homebrew. During this time this work could be publicized through use in papers, talking to fellow scientists and seeing if they would like to use it and posting on forums.