MPHYG001 First Continuous Assessment: Packaging Greengraph

Summary

In an appendix, taken from http://development.rc.ucl.ac.uk/training/ engineering/ch01data/110Capstone.html, are classes which generate a graph of the proportion of green pixels in a series of satellite images between two points.

Your task is to take this code, and do the work needed to make it into a proper package which could be released, meeting minimum software engineering standards

- **package** this code, into a git repository, suitable to be installed with `pip`
- create an appropriate command line entry point, so that the code can be invoked with `greengraph --from London --to Oxford --steps 10 --out graph.png` or a similar interface
- Implement automated tests for each element of the code
- Add appropriate standard supplementary files to the code, describing license, citation and typical usage

Assignment Presentation

For this coursework assignment, you are expected to submit a short report and your code. The purpose of the report is to answer the non-coding questions below, to present your results and provide a brief description of your design choices and implementation. The report need not be very long or overly detailed, but should provide a succinct record of your coursework. The report must have a cover sheet stating your name, your student number, and the code of the module (MPHYG001).

Submission

You should submit your report and all of your source code so that an independent person can run the code. The code and report must be submitted as a single zip or tgz archive of a folder which contains **git** version control information for your project. Your report should be included as a PDF file, report.pdf, in the root folder of your archive.

All coursework should be submitted electronically through the Moodle for the course. There is no need to include your source code in your report, but you can refer to it and if necessary reproduce lines if it helps to explain your solution. The deadline for submissions is 5pm on 11th January 2016. Marks will be available by 1st February.

Marks Scheme

- Code broken up into appropriate files, and arranged into an appropriate folder structure [2 marks]
- Git version control used, with a series of sensible commit messages [2 marks]
- Command line entry point, using appropriate library to parse arguments [3 marks]
- Packaging for `pip` installation, with `setup.py` file with appropriate content [5 marks]
- Automated tests for each method and class (2 marks), with appropriate fixtures defined (1 mark) and mocks used to avoid tests interacting with internet (2 marks). [5 marks total]
- Supplementary files to define license, usage, and citation. [3 marks]
- A text report which:
- Documents the usage of your entry point [1 mark]
- Discusses problems encountered in completing your work [1 mark]
- Discusses in your own words the advantages and costs involved in preparing work for release, the use of package managers like pip and package indexes like PyPI [2 marks]
- Discusses further steps you would need to take to build a community of users for a project [1 mark]

[25 marks total]

Appendix

```
import numpy as np
import geopy
from StringIO import StringIO
from matplotlib import image as img


class Greengraph(object):
    def __init__(self, start, end):
        self.start=start
        self.end=end
        self.geocoder=geopy.geocoders.GoogleV3(
          domain="maps.google.co.uk")

    def geolocate(self, place):
        return self.geocoder.geocode(place,
            exactly_one=False)[0][1]
```

```
    def location_sequence(self, start,end,steps):
      lats = np.linspace(start[0], end[0], steps)
      longs = np.linspace(start[1],end[1], steps)
      return np.vstack([lats, longs]).transpose()

    def green_between(self, steps):
        return [Map(*location).count_green()
                for location in self.location_sequence(
                    self.geolocate(self.start),
                    self.geolocate(self.end),
                    steps)]

class Map(object):
    def __init__(self, lat, long, satellite=True,
           zoom=10, size=(400,400), sensor=False):
        base="http://maps.googleapis.com/maps/api/staticmap?"

        params=dict(
            sensor= str(sensor).lower(),
            zoom= zoom,
            size= "x".join(map(str, size)),
            center= ",".join(map(str, (lat, long) )),
            style="feature:all|element:labels|visibility:off"
          )

        if satellite:
            params["maptype"]="satellite"

        self.image = requests.get(base, params=params).content
        # Fetch our PNG image data
        self.pixels= img.imread(StringIO(self.image))
        # Parse our PNG image as a numpy array

    def green(self, threshold):
        # Use NumPy to build an element-by-element logical array
        greener_than_red = self.pixels[:,:,1] > threshold* self.pixels[:,:,0]
        greener_than_blue = self.pixels[:,:,1] > threshold*self.pixels[:,:,2]
        green = np.logical_and(greener_than_red, greener_than_blue)
        return green

    def count_green(self, threshold = 1.1):
        return np.sum(self.green(threshold))

    def show_green(data, threshold = 1.1):
        green = self.green(threshold)
        out = green[:,:,np.newaxis]*array([0,1,0])[np.newaxis,np.newaxis,:]
```

```
        buffer = StringIO()
        result = img.imsave(buffer, out, format='png')
        return buffer.getvalue()

mygraph=Greengraph('New York','Chicago')
data = mygraph.green_between(20)
plt.plot(data)
```