CSCI 141 - Fall 2019
Lab 8: Putting it all together
Due Date: Friday, December 6th, 2019 at 9:59pm

Introduction

In this lab, you'll write a program to read historical earthquake data from a file and plot each earthquake on a map using turtle graphics. An example output is shown in Figure 1.

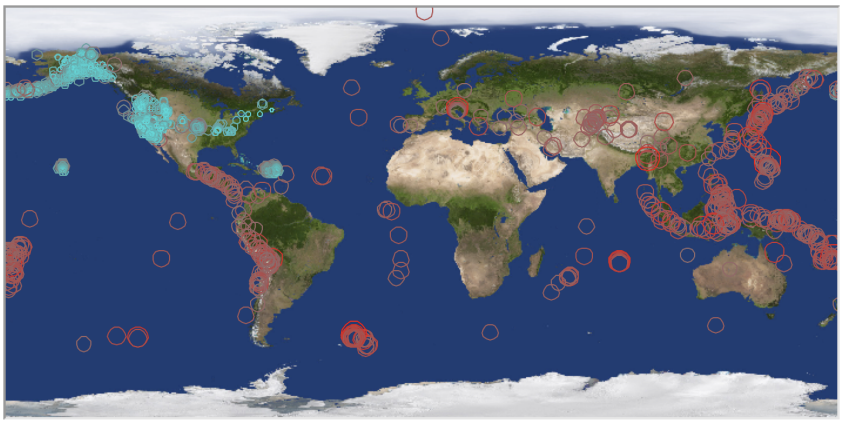

Figure 1: The output from my solution code.

1 Setup

Create a `lab8` directory in your lab environment of choice. Download the following files from the course webpage and place them in your lab8 folder:

- `plot_earthquakes.py` - this file contains skeleton code and pseudocode
- `earthquakes.csv` - this file contains the data you'll be reading
- `earth.png` - this will be set as the background image in the turtle graphics canvas (this is done for you by the `turtle_setup` function given in the skeleton code).

2 Approach and Guidelines

Implement the `parse_row` function and the remainder of the `main` function according to the pseudocode included in comments. Follow the same coding style conventions we've been using up to this point: comment at the top, good variable naming, and so on.

You'll find that you need many of the structures and concepts we've covered in this course to complete this task—ask your TA if you encounter any problems, and use this opportunity to take note of any topics you need to brush up on before the final exam.

Some hints:

- The code to read the csv file will look pretty similar to the code I provided in A5 to read the cancer data files.
- The first line of the file contains column headers, so you'll need to skip over it before starting to read data.
- Plotting earthquakes on the map is quite simple: the map image (and turtle canvas) is 720x360 pixels, with (0,0) in the center. Longitude (the x axis) goes from -180 to 180 and latitude (y axis) goes from -90 to 90, so (0,0) is in the center. To get the canvas (x, y) coordinates based on a given (lon, lat) coordinate, simply multiply each coordinate by 2.
- The skeleton includes an implementation of the `teleport` function from Lab 5.
- You can use a turtle object's `circle` method to draw a circle. See the documentation for details on how to use it.
- Coloring the circles is optional. In my color scheme, the red channel is proportional to magnitude, while the green and blue channels are inversely proportional to magnitude. For an extra challenge, try coloring the circles based on the date instead of the magnitude.

Submission

Take a screenshot of your program's output and save it as `earthquakes.png`. Submit `earthquakes.png` and `plot_earthquakes.py` to Canvas.

Rubric

You submitted `earthquakes.png` and `plot_earthquakes.py` to Canvas.	2
The top of your program has comments including your name, date, and a short description of the program's purpose.	3
The program reads the earthquake data into a list of dictionaries	15
A circle is drawn for each earthquake	5
The circle's size varies with the earthquake's magnitude	5
Total	30 points

3 Challenge - Anagrams

An *anagram* of a word is a different word spelled using the same letters. For instance, "elbow" is an anagram of "below". For purposes of this problem, we'll consider only anagrams that use exactly the same letters, without leaving any out or repeating letters more times than they appear in the original word. For example, "bow" and "bellow" are not considered anagrams of "below" for purposes of this problem.

Download the file `words.txt` from the website and write a program that finds the largest set of 6-letter words that are all anagrams of each other. As an example, among 3-letter words, there are two sets that tie for largest: `['tea', 'eat', 'eta', 'ate']` and `[['aer', 'ear', 'are', 'era']`.

My program is able to find the three-letter sets in a second or two on my laptop. Finding the 6-letter set takes a little longer - around 45 seconds. I didn't import any modules from the standard library, but you may if you'd like—`itertools` might be particularly helpful. For the sake of efficiency, you may want do some thinking and/or research into what kind of collection you store things in; lists may not always be the best choice.

Name your program `anagram.py` and submit it to Canvas for up to 2 points of extra credit. One point is awarded for a correct solution; the second point is awarded for solving it without importing any modules.