Geography 378: Introduction to Geocomputing
Lab 3: Python File I/O and System Access

Assigned: 10/11
Due: 10/25
25 points

Hand-in

- Please collect your answers in separate .py files (e.g. task1.py). Zip these files into a single compressed file called **lab3_yourname.zip.**
- Submit the file to the assignment folder called "Lab 3".
- Include appropriate comments to explain what each line or block of code accomplishes. **You must comment your code for full credit**.

Notes

In this lab you need to deal with bad user inputs. Implement and build on what you know about using while loops, try/except statements, function calls, and testing with print statements to make sure things are working like you think they are. Under no circumstances should your program bomb (i.e. it either restarts or ends gracefully).

Getting Started

The file CityPop.csv is a CSV file (comma-separated values) that contains data for 21 cities with their coordinates and populations (in millions) over several years. If you open this file in Notepad or Notepad++, you will see 22 lines of text with values separated by commas. The first line of the file contains the column names, and the other lines each pertain to a different city. All of the column names begin with a letter; this isn't a big deal here, but many programs run into problems when identifying a value using a number, so this is generally good practice. If you open a CSV file in Excel, you will see a table with each line shown as a row and each comma-separated value in its own cell. Excel detects the CSV format and knows to automatically convert newline characters into row separations and commas into column separations.

For a primer on importing CSV data from a file into a Python script, check out this handy tutorial: www.geospatialtraining.com/blog/index.php/file-handling-with-python-for-gis-programmers.

Lab Tasks

1. (5 pts) Create a python script that reads in the contents of CityPop.csv and stores the data in a container. Task 2 will expect that your program can find cities by name within the container, so think about that as you set up your data container (see **Strategies**). Briefly describe how you store the data in the comments.
2. (5 pts) Extend Task 1 so that the user can query the population of a city in a particular year.

- Prompt the user for a city and a year (note that the format of year is 'yr1970'). Test their input to make sure it is valid and corresponds to values within the file.

3. (5 pts) Extend Task 1 so that the user specifies two cities (from the CityPop.csv file) and the program then calculates the distance between them.
    - Prompt the user for the city names, find the corresponding latitude and longitude values in the file contents (if they exist), and utilize your great circle distance function to calculate the distance.
4. (10 pts) Extend Task 1 so that your program calculates the population change over two years specified by the user for all the cities.
    - Prompt the user for two years. Test their input to make sure it is valid and corresponds to values within the file. Find the population change between those two years for all cities in the database. Output the results to a new CSV file *CityPopChg.csv*. It should **only** contain 3 fields: id, city, population_change. Include the field names in the first line of your CSV file

Strategies

- Start by opening and reading in the contents of the CSV file. Test to see whether the file exists, and exit gracefully if it does not.
- Consider how to store the contents of the CSV for easy manipulation. Get field names from the CSV header (the first line of input) and make sure you can easily retrieve the attribute value in the table. Don't do any programming until you have thought this through.