CSCI 1100 — Computer Science 1 Homework 4
Loops and Lists

Overview

This homework is worth **90 points** total toward your overall homework grade (Part 1 is worth 30 points and Part 2 is worth 60 points). It is due Thursday, March 1, 2018 at 11:59:59 pm. As is usual, there will be a mix of autograded points, hidden points and TA graded style points. There are two parts to the homework, each to be submitted separately. You will need the utilities and data we provide in `hw4Files.zip`, so be sure to download that and unzip it into your directory for HW 4. All parts should be submitted by the deadline or your program will be considered late.

The homework submission server URL is below for your convenience:

`https://submitty.cs.rpi.edu/index.php?semester=s18&course=csci1100`

Your programs for each part for this homework should be named:

```
hw4Part1.py
hw4Part2.py
```

respectively. Each should be submitted separately.

See the handout for Homework 3 for a discussion on grading and on what is considered excess collaboration. These rules will be in force for the rest of the semester.

Final note, you will need to use loops in this assignment. We will leave the choice of loop type to you. Please feel free to use `while` loops or `for` loops depending on the task and your personal preference.

Part 1: Words with alternating vowels and consonants

Here's a short, easy one to get started! Write a program that reads in words entered by the user and checks whether:

- the word has at least 8 characters,
- has alternating consonants and vowels, and
- the vowels are in non-decreasing alphabetical order.

The program should loop reading additional words until the user enters an empty string.

Note that you can check letters for alphabetical order using `<`.

For example:

```
>>> 'a' > 'b'
False
>>> 'z' > 'b'
True
```

Your program should work for words entered upper or lower case, and must use a function `is_alternating(word)` that returns `True` if the word has the above pattern, and `False` otherwise. You are welcome to use additional functions if you would like. **Hint.** Remember to write a loop that goes through each letter and checks for the necessary conditions. Using indexing is important here as you need to compare letters in different positions. Here is an example run of this program:

```
Enter a word: afegiritiw
afegiritiw
The word 'afegiritiw' is alternating

Enter a word: afegiretiw
afegiretiw
The word 'afegiretiw' is not alternating

Enter a word: zafegiritiw
zafegiritiw
The word 'zafegiritiw' is alternating

Enter a word: tesur
tesur
The word 'tesur' is not alternating

Enter a word: Tesurrux
Tesurrux
The word 'Tesurrux' is not alternating

Enter a word:
```

And using real words,

```
[evaluate hw4Part1.py]
Enter a word: acAdEmic
acAdEmic
The word 'acAdEmic' is alternating

Enter a word: acari
acari
The word 'acari' is not alternating

Enter a word: aCAriDan
aCAriDan
The word 'aCAriDan' is not alternating

Enter a word: bejEweleD
bejEweleD
The word 'bejEweleD' is alternating

Enter a word: bejeweled1
bejeweled1
The word 'bejeweled1' is not alternating

Enter a word:
```

Here is a little hint that will help: Given a letter stored in the variable x, the boolean expression:

```
x in 'aeiou'
```

is `True` if and only if `x` is a lower case vowel.

When you have tested your code, please submit it as `hw4Part1.py`. Be sure you use the correct filename, otherwise, Submitty will not be able to grade your submission.

Part 2: ZIP code look up and distance calculation

As part of an Open Data Policy, the U.S. Government released a large number of datasets "to conduct research, develop web and mobile applications, design data visualizations" [1]. Among these is a dataset that provides geographic coordinates for U.S. 5 digit ZIP codes. For this assignment, we will be using a slightly improved version of the ZIP dataset available at [2]. For each ZIP code, this dataset lists geographic coordinates (latitude and longitude), city, state, and county. Our goal is to use available data to develop a new data product that would allow users not only to query and search the existing data source but also to access additional functionality such as computing the distance between two locations.

Write a program that would allow users to lookup locations by ZIP code, lookup ZIP codes by city and state, and determine the distance between two locations designated by their ZIP codes. The program interacts with the user by printing a prompt and allowing them to enter commands until they enter `'end'` at which point the program prints `Done` and finishes. If an invalid command is entered, the program prints `Invalid command, ignoring` and is ready to take the next command.

The following commands are recognized:

1. **loc** allows the user to enter a ZIP code, then looks up city, state, county, and geographic coordinates that correspond to the ZIP code and prints this data; If a ZIP code is invalid or not found in the dataset, the program prints a error message instead. Look at the following sample output:

   ```
   Command ('loc', 'zip', 'dist', 'end') => loc
   loc
   Enter a ZIP code to lookup => 12180
   12180
   ZIP code 12180 is in Troy, NY, Rensselaer county,
           coordinates: (042°40'25.32"N,073°36'31.65"W)
   ```

   Note that coordinates are enclosed in parentheses and are separated by a comma; each coordinate is printed in integer degrees (three digits), followed by the ° symbol, integer minutes (two digits), followed by the ' character, integer and fractional seconds (two integer and two decimal digits), followed by the " character, and a cardinal direction letter (N, S, W, or E) with no spaces anywhere in the coordinate representation.

2. **zip** allows the user to enter city and state, then looks up the ZIP code or codes (some cities have more than one) which correspond to this location and prints them; if city and/or state are invalid or not found in the dataset, it prints an error message instead.

   ```
   Command ('loc', 'zip', 'dist', 'end') => zip
   zip
   ```

```
Enter a city name to lookup => troY
troY
Enter the state name to lookup => ny
ny
The following ZIP code(s) found for Troy, NY: 12179, 12180, 12181, 12182, 12183
```

3. **dist** allows the user to enter two ZIP codes and computes the geodesic distance between the location coordinates ; if any of the ZIP codes entered is invalid or not found in the dataset, it prints a corresponding error message instead.

```
Command ('loc', 'zip', 'dist', 'end') => dist
dist
Enter the first ZIP code => 19465
19465
Enter the second ZIP code => 12180
12180
The distance between 19465 and 12180 is 201.88 miles
```

4. **end** stops fetching new commands from the user and ends the program

The utility module provided for this part of homework will give you some help. It provides a function `read_zip_all()` that returns a list each element of which is, in turn, a list that contains data about one zip code. Try the following:

```
import hw4_util
zip_codes = hw4_util.read_zip_all()
print(zip_codes[0])
print(zip_codes[4108])
```

would give:

```
['00501', 40.922326, -72.637078, 'Holtsville', 'NY', 'Suffolk']
['12180', 42.673701, -73.608792, 'Troy', 'NY', 'Rensselaer']
```

Note that the data on a particular ZIP code has the following fields in order: zip code (string), latitude (float degrees), longitude (float degrees), city (string), state (string), and county (strings).

Implementation Details

You will need to define two functions that should strictly follow specifications outlined below:

1. `zip_by_location(zip_codes, location)` which finds the ZIP code for a given location.
   Parameters:
   `zip_codes` a list of ZIP codes data in the format, returned by `read_zip_all()`

   `location` a two-element tuple where the first element is the city name and the second element is the state abbreviation, e.g. `('trOy', 'nY')`. Both elements are string values. City names

and state abbreviations can be written using any case or a mixture of lower and upper case.

Return value:
A list which contains ZIP code or codes for the specified location. Each ZIP code is a string value. If the location is invalid, an empty list is returned.
E.g., `['12179', '12180', '12181', '12182', '12183']`

2. `location_by_zip(zip_codes, code)` which finds location information corresponding to the specified ZIP code.

   Parameters:

   `zip_codes` a list of ZIP codes and associated data in the format, returned by `read_zip_all()`

   `code` ZIP code as a string value, e.g. `'12180'`.

   Return value:
   A five-element tuple `()latitude, longitude, city, state, county)`. Latitude and longitude are in fractional degrees (floating point values). All other elements are string values.
   E.g., `(42.673701, -73.608792, 'Troy', 'NY', 'Rensselaer')`

It is required that you implement functions according to specifications given above. You are also expected to define other functions, as necessary. You need to determine which functions to define and to design their specifications yourself, following the example we gave above for the two required functions. Do not forget to include function specifications as comments right above the function's definition in your code. Avoid copying and pasting repeated pieces of code. Those should be factored out as functions. You may lose points for excessive duplication of code.

Hints and Helps

Formatting your output is a big part of this program. Here are a few hints.

1. The ° symbol can be generated using another escape character similar to `'\n` or `'\t'`. To generate the °, use `\xb0`. Just like the tab and the newline, this is a single character.
2. When printing out the location, there is a tab character at the start of the second line before `coordinates`.
3. if you want to print leading `0`s before an integer, use the `{:0Nd}` where `N` is the total number of spaces you want the integer to occupy. So a format of `:07d` will use 7 spaces to print your integer, padding the front of the integer with `0`s. I.e.

   ```
   print("{:07d}".format(7))
   ```

   will give

   ```
   0000007
   ```

4. You will need to manage the latitude and longitude to convert among representations. In particular, you need to convert the fractional degrees in the zip code list to degrees, minutes, seconds. There are `60` minutes in a degree and `60` seconds in a minute. When you convert, all of your latitudes and longitudes should be positive. Instead, use east (`E`) and west (`W`) designators of longitude; and north (`N`) and south (`S`) designators for latitude. Negative longitudes are west, positive are east. Negative latitudes are south, positives are north. In bath cases, a latitude or longitude of `0` (the equator or the prime meridian, respectively) do not have a designator.

5. The distance between two points on the surface of Earth uses a simplified haversine formula for arc length on a sphere. You should implement Eq. 1 which assumes the Earth to be spherical. This formula gives the shortest surface distance d "as-the-crow-flies", ignoring Earth's relief.

$$\begin{aligned} \Delta latitude &= latitude_2 - latitude_1 \\ \Delta longitude &= longitude_2 - longitude_1 \\ a &= sin^2\left(\frac{\Delta latitude}{2}\right) + cos(latitude_1) \cdot cos(latitude_2) \cdot sin^2\left(\frac{\Delta longitude}{2}\right) \\ d &= 2R \cdot arcsin\left(\sqrt{a}\right) \end{aligned} \tag{1}$$

   Where $R = 3959.191$ miles is the radius of the Earth and all angle measurements are in **radians** (the zip code data gives latitude and longitude in degrees).

Sample Output

Below is an example output that demonstrates program features:

```
Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP code to lookup => 12180
12180
ZIP code 12180 is in Troy, NY, Rensselaer county,
        coordinates: (042°40'25.32"N,073°36'31.65"W)

Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP code to lookup => 1946
1946
Invalid or unknown ZIP code

Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP code to lookup => 19465
19465
ZIP code 19465 is in Pottstown, PA, Chester county,
        coordinates: (040°11'30.87"N,075°39'55.12"W)

Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP code to lookup => 00000
00000
```

```
Invalid or unknown ZIP code

Command ('loc', 'zip', 'dist', 'end') => loc
loc
Enter a ZIP code to lookup => ksahdkja
ksahdkja
Invalid or unknown ZIP code

Command ('loc', 'zip', 'dist', 'end') => dist
dist
Enter the first ZIP code => 19465
19465
Enter the second ZIP code => 12180
12180
The distance between 19465 and 12180 is 201.88 miles

Command ('loc', 'zip', 'dist', 'end') => dist
dist
Enter the first ZIP code => 12180
12180
Enter the second ZIP code => 12180
12180
The distance between 12180 and 12180 is 0.00 miles

Command ('loc', 'zip', 'dist', 'end') => dist
dist
Enter the first ZIP code => 12180
12180
Enter the second ZIP code => 55499
55499
The distance between 12180 and 55499 cannot be determined

Command ('loc', 'zip', 'dist', 'end') => zip
zip
Enter a city name to lookup => troY
troY
Enter the state name to lookup => ny
ny
The following ZIP code(s) found for Troy, NY: 12179, 12180, 12181, 12182, 12183

Command ('loc', 'zip', 'dist', 'end') => zip
zip
Enter a city name to lookup => Amsterdam
Amsterdam
Enter the state name to lookup => NL
NL
No ZIP code found for Amsterdam, NL

Command ('loc', 'zip', 'dist', 'end') => help
help
Invalid command, ignoring

Command ('loc', 'zip', 'dist', 'end') => END
END
```

```
Done
```

When you have tested your code, please submit it as `hw4Part2.py`. Be sure to use the correct filename or Submitty will not be able to grade your submission.

References

[1] "Data.gov: The home of the U.S. Government's open data," 2018, Accessed on: Feb. 23, 2018. [Online]. Available: https://www.data.gov/

[2] "Download: Zip code latitude longitude city state county csv," San Diego, CA, USA, 2018, Accessed on: Feb. 23, 2018. [Online]. Available: https://www.gaslampmedia.com/download-zip-code-latitude-longitude-city-state-county-csv/