2022 Spring -- CSCI 1300L
Lab 09: Dictionary operations: processing time zone data

Introduction

In this lab, you will use the dictionary structure in Python. You will create a small application that deals with searching and comparing world time zone data. The dictionary structure, which stores key-value pairs, is better than a single-element structure of data collection (e.g. a list) in this assignment. Specifically, keys in the dictionary store the location information and their values store the time zone information.

Lab objectives

After completing this lab, you will be able to:
(1) understand how key-value pairs apply to a real-world object,
(2) construct expressions involving dictionaries, and
(3) select and use dictionary operations and methods to process data.

Assignment

Travelers often look up a destination's time zone information before their plan of travel. They also need to know the time difference (i.e. offset) between the origin and the destination. In this assignment, you will create a Python program that helps travelers search a location's time zone information and compare difference (in hours) between two locations.

For simplicity, we assume that the program only deals with the "standard" time zone information. That is, no daylight-saving time handling or any special requirement set by the local government during a certain period of time or at a specific area. All time zone data are provided for your convenience, represented by the Coordinated Universal Time (UTC).

Use the provided document *time.txt*, which has been well formatted with the dictionary's structure, as your program's dictionary data. You will only need to (1) copy the entire plain text data, (2) add { and } before and after the data to make it as the dictionary data, and (3) use a dictionary name for the enclosed data.

Example: my_time_zone_data = { xxxxx……. xxxxx }

Red texts are what you need to add into the Python code. Included in the curly braces, " xxxxx……. xxxxx" are the copied data from *time.txt*. Note that a technically preferred way of processing a large data set is to read from a file or a database rather than to copy the content into the program code. At this time, we do not focus on file processing.

Next, it will be the time to write the Python code using the loaded dictionary data to perform the following tasks:

1. User enters the location data. The program tells its time zone information.
If such a location does not exist, the program reports that there is no such a location.

2. User enters two location data. The program displays the time zone information for both.
The program also calculates the time zone difference (offset) and displays the result.
If any one location does not exist, the program reports that the invalid location data.
No comparison of time zone difference if any one location input is invalid.

3. User enters a valid number (a value from the time zone's values).
The program provides the list of locations that have the same time zone value.
Hint: using a loop, and/or a branching statement.
The program reports the user's invalid input, e.g. an out-of-range number or a string.

4. Allow the user to enter a new location and its time zone information.
Then, the program displays such a location name and its time zone information.
Such information MUST be first stored and then retrieved from the dictionary.

Here is an example of the program's possible input and output. Your program's input and output may be slightly different from this example. Red texts are user inputs.

Please enter a location:
Greenwich
Its time zone is 0.

Please enter two locations, one for origin and the other for destination:
America/New_York
Europe/Zurich
Origin: -5; Destination: 1. Time zone offset: -6 hours

Please enter a time zone value:
0
Here are the locations in this time zone:
Africa/Abidjan
Africa/Accra
…
Zulu

Add a new location and its time zone information:
America/Athens,GA
5
Here is the data you have added into the time zone database:
{'Amercia/Athens,GA' : 5}

For this assignment, you can customize the text input and output if you want. We mainly check the functioning of the program. For the user's location input, we assume that the user will enter exact location data that would have a match in the dictionary's keys, for example, they will enter "America/New_York" rather than "New York". This is to ensure that such a key can be found. However, if the user enters "ny" or "new", the program will report an invalid location name.

Additionally, you do not have to write the program to repeatedly interact with the user by accepting the user's input. The program will finish once all four tasks are performed.

You are allowed to use any dictionary operators and methods that had been introduced in the class ONLY. You cannot use any other Python library or "exotic" methods for this programming challenge. The entire assignment must be completed with the creation and use of a dictionary. If no dictionary structure is used in this assignment, no credit will be given.

Submission instruction

After you have completed the assignment, upload and submit the Python source code file *Lab09.py* to eLC. You do not need to submit the *time.txt* file along with your Python code. Always double check that your submission was successful on eLC.

Grading

A score between 0 and 5 will be assigned.

1. Part-(1) works: successful handling of the user input (0.5) and displaying of data (0.5). (1 point)
2. Part-(2) works: successful handling of two user inputs (0.5) and correct processing and displaying of time zone difference (0.5). (1 point)
3. Part-(3) works: successful handling of the user input (0.5) and displaying of data (0.5). (1 point)
4. Part-(4) works: successful handling of the user input (0.5) and the program's ability to add the user input (a key-value pair) into the dictionary (0.5). (1 point)
5. Only a single source code file is submitted and no other file is submitted (0.5) and the entire Python program can be executed without any additional error (0.5). (1 point)

Special notice regarding the submission:

Late submission penalty. Points will be deducted from the original grade. If your submission is after the posted deadline…

(1) within 24 hours: -2
(2) between 24 hours and 48 hours: -3
(3) between 48 hours and 72 hours: -4
(4) after 72 hours: assignment will not be accepted.