Project 1 Reflection

C.S. John Lee

10/27/2019

<u>What I completed</u>

I completed a program that will input a few user preferences of cities and return a happiness index of a city. There are choices to either select a city and return the happiness index and several characteristics of the city or to return a list of ALL cities ranked by happiness index. Although I am content with the results and proud of applying classes in this program, there are definitely things I'd like to improve upon in this program. For instance, it only takes into account 15 cities which could be expanded. Maybe some data science methods can help to find info for all cities in the USA or even the World. This program also takes into account only 6 attributes, equally weighted, to determine if a city meets your needs, we definitely know there's more than 6 attributes and that each attribute weight differently to each person.

<u>How to Test and Use Project Location Happiness Score Program</u>

- ➔ Run the program and you should be prompted to input your name
- ➔ Afterwards, there will be (3) questions to determine your characteristic
  - Daylight throughout a year -> enter either steady or fluctuating, others will raise error
  - Weather -> enter either hot or cold, others will raise error
  - Population Density -> enter high or low, others will raise error
- ➔ User characteristics will be printed, next would be for you to enter a command
  - Enter m/move, f/forever home, h/help, q/quit, others will raise an error
  - If m/move:
    - Enter s/select/select location or r/return, others will raise an error
    - If s/select/select location:
      - List of locations appears, must type one of these locations
      - Input the exact characters (not case dependent), others will raise an error and send you back to first command prompt
      - If correct location inputted, location characteristics and the happiness index will be returned
    - If r/return:
      - Returns the places where you've moved
      - Goes back to first command prompt
  - If f/forever home:
    - Shows the list of all locations (x15) ranked by highest to lowest happiness score
    - Goes back to first command prompt
  - If h/help:
    - Shows an explanation of each command from first command prompt
    - Goes back to first command prompt
  - If q/quit:
    - Quits the program

<u>Challenges Faced when Creating Project Location Happiness Score Program</u>

1) When errors returned in the command prompt, it can take an hour or two to fix the error. Luckily, these errors are due to my lack of understanding of classes meaning that as I learn, I will have less errors. I didn't have many errors where it was due to a mistake (i.e. missing semi colon like C++).
2) When changing or fixing part of the code, I was always worried that it would affect another part of my code. Classes help separate different issues like this but in this program, if I wanted to add a new attribute to the locations, that would force me to update the general equation for the happiness score, which also updates the command prompt to user when you want to show the user the new attribute, and finally, should the person also have an input?
3) When close to completing the project, I would run the program over and over to see where the user might run into problems. I'd have to predict different ways the front-end can be ineffective or lead to errors so this was challenging. Additionally, I added time.sleep() in several areas which slowed this process. I learned to comment the time.sleep() out of the program until I debug the rest of the program.
4) Trying to make this program pythonic was hard. I attempted to do so by organizing print statements into a class so that the prompt is much cleaner. I tried to comment confusing methods so that it will help. I received feedback to avoid commenting too much because PEP 257 standard says to only comment what the function does not how it works. Lastly, when creating the method "statslocation" under class "happiness", I noticed that I was repeating a LOT by creating the list one-by-one for each city. I added a for-loop and changed the code from 15 x 3 lines to only 6 lines of code!
5) After running this program, I realized how simple/trivial it is. Behind the scenes is a 400 line code of messiness and some headaches. I definitely learned how challenging certain small details were, especially when accounting for all the different ways a user could input into the program.