Practicum 3: Boston Neighborhood Demographics

Due Date: September 24th at 11:59pm EST

This week, we'll be using data from Boston's Analyze Boston data portal to look at demographic data of the city's different neighborhoods. On Canvas, I've provided data files for 9 neighborhoods: Allston, Back Bay, Dorchester, Fenway, the Harbor Islands, Jamaica Plain, Longwood, Rosindale, and Roxbury. Each file contains the following information on three separate lines:

- The area of the neighborhood (in square miles)
- The population of the neighborhood
- The median age of the residents in the neighborhood

Download the files and *make sure they are in the same folder* as your `practicum03.py` file.

We'll use the demographic data to practice the following:

1. Reading data from files
2. Plotting data
3. Working with conditionals

1 Reading Demographic Data

Write Python code that does the following:

1. Prompts the user for which neighborhood that they would like to analyze
2. Reads the neighborhood's data from its corresponding file. For simplicity, assume that if the users specifies the Harbor Islands or Jamaica Plain then the input is the same as the neighborhood's filename, like `harborislands` and `jamaicaplains`.
3. Prints the information about the neighborhood

The program output should look like this (user input in **bold underlined text**):

```
What neighborhood would you like to know about? Fenway
Fenway is 0.88 square miles large, has 32598 residents, and the median
age is 23 years old.
```

In your code, make a comment and answer the following question: What are two things that a user might do to accidentally cause your program to encounter an error? Users usually aren't trying to break your program, but it's good practice to try and anticipate what might go wrong if someone else used your code.

2 Comparing Demographics

Let's visualize differences between the neighborhoods using Matplotlib. In DS2000, you saw how you can use the `plot()` function to plot points. Today, we're going to use the `bar()` function instead to make bar graphs. The bar function takes (at least) two things: the coordinate where we want to place the bar (horizontally), and how high we want the bar to be. So if we want to use it, it looks something like this:

```
# Remember to import matplotlib at the top of your file
import matplotlib.pyplot as plt

# Plotting a single black bar at coordinate 1 with a height of 5
plt.bar(x=1, height=5, color="black")
```

Write Python code that does the following:

1. Prompts the user for at least 3 neighborhoods
2. Prompts the user for what they would like to compare (size, population, or age)
3. Reads the size, population, and median age date for each of the three neighborhoods
4. Plots a bar chart based on what the user specified in Step 2

The program output should look like this (user input in **bold underlined text**):

```
What is the first neighborhood you want to compare? Fenway
What is the second neighborhood? Roxbury
What is the third neighborhood? Dorchester

What demographic do you want to compare? Population

# The final output would then be a plot of the populations of Fenway,
# Roxbury, and Dorchester
```

Make sure that you label your axes!

Did you notice how a lot of your code is the same for each neighborhood? Soon in DS2000 you'll learn how to write this kind of program without having to copy-paste the same code over and over. This will let us plot more data much more easily.

3 Ideal Neighborhoods

Websites and apps like Craigslist, Zillow, and Airbnb allow users to specify criteria for what kind of neighborhood they would like to live in or visit. Write Python code that does the following:

1. Prompts the user for their ideal geographic size of a neighborhood (in square miles), the ideal population size, and the ideal median age.

2. Checks if the neighborhood from Part 1 (Reading Demographic Data) of this assignment matches their criteria. A neighborhood is close enough to the ideal criteria if:
    a. The geographic size is **within 0.25 miles** of the value they enter.
    So if a user says the size of their ideal neighborhood is 2 square miles, then we'll say the neighborhood matches the criteria if it is bigger than 1.75 square miles and less than 2.25 square miles.
    b. The population size is **within 3,000 people** of the value they enter
    c. The median age is **within 4 years** of the value they enter

3. Prints out one of three messages:
    a. If *all 3 criteria are met* for the ideal neighborhood, then print out a congratulatory message saying the neighborhood is a great match
    b. If *only 1 or 2 of the criteria are met*, then print out a message telling the user that the neighborhood is an OK match
    c. If *none of the criteria are met*, then print out a message telling the user that the neighborhood is a poor match

The program input should look like this (user input in **bold underlined text**):

```
What is the geographic size of your ideal neighborhood? 2.5
What is the population size of your ideal neighborhood? 35000
What is the median age in your ideal neighborhood? 24
```

The program output should look something like the following, depending on the user input:

```
Fenway fits your size criteria.
Fenway does not fit your population criteria.
Fenway does not fit your age criteria.

Fenway is an OK match.
```

Finish Early?

Try making the following changes to your code:

1. In Part 1, we assumed that the user would input `harborislands` or `jamaicaplain` (without spaces) if they wanted information about the Harbor Islands or Jamaica Plain. Use conditionals to allow your code to work when the user places a

space in the neighborhood's names, like `harbor island` or `jamaica plain`.

2. In Part 1, we also assumed that the user would type a lowercase version of the neighborhood as input. Edit your code so that it runs even if the user passes a mixed case input like `fEnWAy`.

3. In Part 2, we plotted bar charts of demographic data for different cities. Even though the user knows what cities they input, your final plot doesn't actually label which bar corresponds with which city. Find a way to edit your plot so that it is labeled properly. **Hint:** Look up the function `plt.legend()` or `plt.xticks()`.

4 Submit Your Work

When you are done, submit the following to Canvas in a zip folder:

1. The Python file (.py) containing your code answers for this practicum.
2. A text file (.txt, .doc, .pdf, etc.) with any comments about anything you were not able to get working and what you tried to solve it.

Please name your zip folder `LastName_Practicum03.zip`.

This assignment was originally created by Felix Muzny. It has been updated here by Ryan Gallagher.