# Practicum 3 - Measuring Boston Neighborhoods

**DS2001 - Computer Science Practicum**
**Fall 2021**
**September 22-23, 2021**
**Deadline:** September 24th, 2021 at 12:00pm noon Boston time

**What we're practicing today:**
- writing programs with a main
- conditionals (branching)
- reading data from files (file i/o)
- graphing (with matplotlib)

**Handouts:**
- File Processing (from DS 2000)
- Data visualization with matplotlib (from DS 2000)
- Conditionals (from DS 2000)

**What to do if you miss practicum this week:**
- Fill out the "I'm missing class" form ASAP
- Follow up in office hours with Felix (find office hours links on the course website)
- Come to practicum next week if you are well again. **Do not come to class if you are not feeling well.**

Create **one** file for the whole practicum called **p3.py**, and we'll add to it as we go. We have a few problems to tackle today; get through as many as you can, and submit whatever you have done at the end of your section. Each question in this write-up will describe a problem to work on, then will ask you a question that you'll answer with your group.

At the end, everyone should make sure to turn in your code to gradescope **with your group's answers to the questions written as comments in your file!**

**Task 0: Write your file comment, your `main(), and import the matplotlib module`**
First, write down your name in a comment at the top of your file and a small description of what the file is, like we did last time. This is a good habit to get into for all your programs!

Write down your group members' names in comments below this.

Then, write down a skeleton for your main() function. All the code that we write today will go inside this function.

```
import matplotlib.pyplot as plt # so that we can graph things

def main():
    # we'll start off with a print so that we can run our code
    print("Practicum 3!")

# don't forget to call your main function!
main()
```

**Task 1: Read in the population data**

This week, we'll be looking at tracking some demographics data. Specifically, we'll be looking at data regarding the neighborhoods in Boston. We got this data from one of Felix's favorite data sources: Analyze Boston.

We've provided you with a number of data files that are named like NEIGHBORHOOD.txt. There is one data file for each of a selection of neighborhoods in Boston. Download 3 or more of these files and make sure they are in the same folder as your p3.py file.

Each data file contains the following information, each on three separate lines:
area (in square miles)
population
median age of residents

First, ask the user which neighborhood they would like to analyze, then load in the data from that file. You'll want to save each of the pieces of information into its own variable, then display (use the print function) this information to the user.

Example (user input is in **bold underlined**):

```
What neighborhood? roslindale

information about roslindale
size: 2.51
population: 29206
median age: 39
```

**Group Question**: What are two things that a user might do to accidentally cause your program to encounter an error? (The user isn't trying to break your program—we all know that humans aren't always perfect at following instructions though!) Write down your answer in comments in your program!

**Task 2: What is the user's ideal neighborhood?**

Ask the user what their ideal value for each metric is in their neighborhood. Then, tell them whether or not the neighborhood that they picked from Task 1 fits their criteria.

- For size, if the size is within 0.25 square miles of the number that they enter, this counts as "fitting".
- For population, if the population is within 3000 people of the number that they enter, this counts as "fitting".
- For age, if the median age is within 4 years of the number that they enter, this counts as "fitting".

Next, print out a message either congratulating the user for finding a good fit if 3 out of 3 of their ideal metrics matched, a message telling them that the fit is so-so if 1 or 2 matched, or a message telling the user that the neighborhood isn't a great fit for them if 0 metrics matched.

Example (user input is in **bold underlined**):

```
...
Ideal size? 2.5
Ideal population? 35000
Ideal age? 24
```

```
roslindale fits your size criteria.
roslindale does not fit your population criteria.
roslindale does not fit your age criteria.
roslindale is a so-so match.
```

**Group Question**: Test your program on files for a few different neighborhoods. Did it work? Did you need to update any code? What code did you have to update?

**Task 3: Load in information about a second neighborhood**

Now, if a user did not find their ideal neighborhood (0 out of 3 matches), we'd like to let them compare the neighborhood that they chose in Task 1 to a second neighborhood. Prompt the user for the name of the neighborhood that they'd like to compare against, then load in the corresponding data file.

For the three metrics: size, population, and median age, print out a statement saying whether they are higher or lower than the user's original neighborhood.

Example (user input is in **bold underlined**):

```
...
Too bad! Let's check out another neighborhood.

Compare against? jamaicaplain
jamaicaplain's size is higher
jamaicaplain's population is higher
jamaicaplain's age is lower
```

**Group Question**: What might a better comparison than size & population reported separately be for seeing if a neighborhood is a "good fit" for a user based on their size (both geographic & population) preferences?

**Task 4: Graph the data**
Next, we'll visualize the differences between the neighborhoods with matplotlib. (Make sure to consult the matplotlib handout!

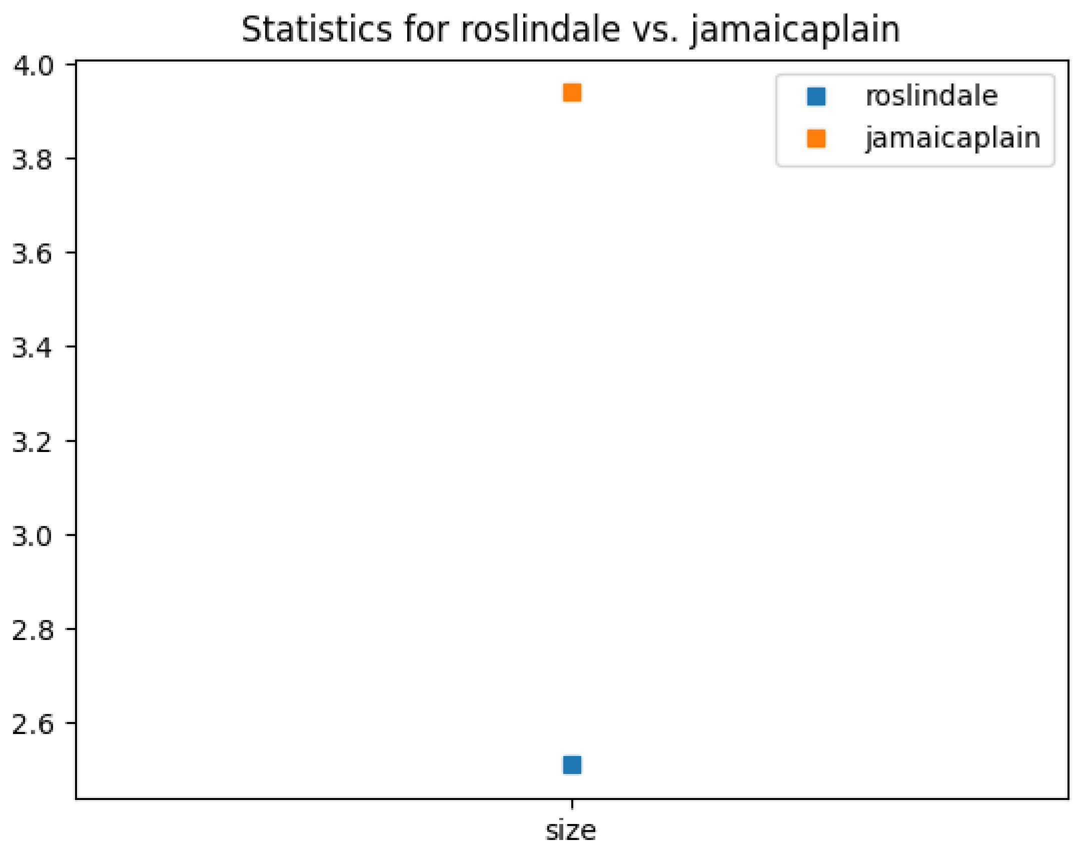
For each of the three metrics: size, population, and median age, we'll make a graph their values for the neighborhoods. We'll use a name (e.g. "size") for the x-value and the actual value of the metric as the y-value. Then, we'll label each data point with the neighborhood that it's from.

Here's an example of some code that you might use to plot a data point:

```
# x-value is the string "size", y value is the size value,
# plot this point as an x, and label it with the neighborhood
# note that size and neighborhood are variables!
plt.plot("size", size, "x", label = neighborhood)
# plot your other point
```

```
plt.legend()
plt.title("your title here")
plt.xlabel("label your y axis")
plt.xlabel("label your x axis")
plt.show()
```

You might produce a graph like:



**Group Question**: Notice that size, population, and median age are all different orders of magnitude. What might be one good way to graph these all on the same graph instead of three separate graphs. Try this out!

**Improving your program (do these if you finish early):**

- personalize your chart by changing the color of the markers
- make it so that your program still works even if the user enters their neighborhood name in MiXeD CAse or CAPITALS
- difficult: make it so that if the user puts spaces between words like "jamaica plain", your program still reads the correct file
    - hint: think about using the str.split() and str.join(list) functions.