DS2001 - CS Practicum

Spring 2023

Practicum 4 –Loops and Lists and Functions

Practicum is DUE at the end of your scheduled practicum.

We'll be working in pairs, and today you can choose your own partner instead of being assigned. You may not finish every problem every week, but we expect your best effort.

We will let you know when there are 15 minutes left in class – at that point, it is time to wrap up your code and move on to the self-reflection portion of the assignment! Read the self-reflection questions below, write your answers, and save them in a PDF.

Submit both your code and your self-reflection PDF on Gradescope.

Grading Policy
You will receive full credit if you submit your self-reflection (with thoughtful answers) and any code by the deadline. All questions in the self-reflection must be answered for full credit.

Feedback
We will provide feedback on your code if requested in your self-reflection. Otherwise, we will not look at your code. Our feedback will mirror the expectations of your DS200 homeworks, and we will use the same criteria for grading your projects in DS2001. So, make sure you ask for feedback when helpful, and read our notes!

Today's Goals

- Open and read data from a file using a while loop
- Store data in lists
- Conduct an analysis of the data

Working in Pairs

Each pair will work at one computer, and everyone contributes. Here's how we split up the work:

- Navigator: Dictates the code to be written. Explains the *why* as we go. Checks for syntax errors.
- Driver: Writes the code. Listens closely to the navigator. Asks questions whenever there is a lack of clarity.
- Both driver and navigator are responsible for contributing to the work, and for making space for the other person to make contributions.

Programming Assignment

For today's practicum, you'll be re-visiting practicum 3 and improving your code with the new knowledge you have from recent lectures! Re-working code to use a new technique or to be more efficient is a *very common* thing in computer science. So common in fact that there's a name for it – refactoring.

1. Gather Data

Download bike_counts.txt from the course website and save it in the same folder as the python file you will write today. This datafile contains counts of bicycles on Broadway St. in Cambridge in 15 minutes increments for one week of 2023 (01/15 – 01/21). The data is derived from this source: https://data.cambridgema.gov/Transportation-Planning/Eco-Totem-Broadway-Bicycle-Count/q8v9-mcfg/data.

The format of bike_counts.txt is:

```
Day (day of the week)
Date
Time
Total (total bicycles)
Westbound (bicycles heading west)
Eastbound (bicycles heading east)
```

Start a python file for today's work. Above `main`, save the data filename as a constant variable.

Outside of `main`, declare a function with the name `read_data`. This function should have a parameter for the name of a data file. In the body of read_data, create empty lists that will each hold one type of data we have to work with: day, date, time, total, westbound, and eastbound. Open the file passed to `read_data` and use a while loop to read in the data, appending each line to the appropriate list. Have `read_data` return a list of lists, where each list is one of the lists you filled in reading in the data file (ex. [days_lst, dates_lst, …]).

Inside of `main`, call `read_data` to read in bike_counts.txt. Save the result as a variable.

2. Computations

Outside of `main`, declare a function with the name `total_bikes`. This function should have a parameter for a list of bike counts. In the body of `total_bikes`, calculate the total number of bikes. Have `total_bikes` return the string: "The total number of bikes was <total number>". To check if your function is working, you will need to call it in `main`.

Outside of `main`, declare a function with the name `west_vs_east`. This function should have parameters for a list of bike counts westbound and a list of bike counts eastbound. In the body of `west_vs_east`, calculate total bikes observed going eastbound and westbound and determine which is larger. Have `west_vs_east` return either the string: “More bikes traveled westbound.” or the string “More bikes traveled eastbound.”, as appropriate. To check if your function is working, you will need to call it in `main`.

Outside of `main`, declare a function with the name `max_15mins`. This function should have a parameter for a list of bike counts. In the body of `max_15mins`, calculate maximum number of bikes observed in one 15 minute segment. Have `max_15mins` return the string: “The maximum number of bikes in one 15 minute segments was <max number for a 15 minute segment>.”. To check if your function is working, you will need to call it in `main`.

Outside of `main`, declare a function with the name `daily_totals`. This function should have parameters for a list of days and a list of bike counts. In the body of `daily_totals`, use a for loop that iterates by position to calculate the total number of bikes observed each day of the week. **Note:** This is similar to what you did originally in practicum 3, but this time asks you to *use a for loop that iterates by position, not value*. Have `daily_totals` return a list containing the total number of bikes for each day of the week (ex. [mon_total, tue_total, …]). To check if your function is working, you will need to call it in `main`.

3. Communication

In `main`, call each of the functions you declared in the computations section above. Print the results of each to the user.

Outside of `main`, declare a function with the name `daily_bar_plot`. This function should have a parameter for a list of total bikes per day of the week. In the body of `daily_bar_plot`, Use matplotlib to create a bar chart that shows how many total bikes were observed each day on Broadway St.. Call `daily_bar_plot` in `main` to produce a bar chart of number of bikes per day of the week.

4. Test Those Functions!

Change the filename saved as a constant above main to bike_counts_v2.txt. This file is formatted identically to bike_counts.txt but has data for a different week. Download the file from the course website and save it in the same folder as your practicum 4 code. Run your code to check that it still works with this new file. If it doesn’t run, debug!

At this point, please complete the self-reflection! If you still have time afterwards, scroll down for a little extra/bonus work you can do on this assignment.

Self-Reflection

Create a PDF with answers to the following questions (each should be answered collaboratively with your partner unless instructed otherwise):

- Does this work reflect the best effort of both partners?
- In what way(s) did each partner (1) make contributions to the assignment, and (2) give the other person space to make contributions? (Each partner should respond to this question separately.)
- Which problem(s) would you like feedback on and why?

Keep Going (if time!)

Here are some ways you can improve and build on what we've done so far. Try any/all of them in the time we have left.

- Declare functions that…
    - Calculate the total bikes traveling westbound and eastbound on each day of the week. Create a grouped bar chart that shows the comparison of these two numbers.
    - Print out the first date and time where 0 bikes are observed on Broadway.
    - Count and print how many times the maximum number of bikes were observed on Broadway.