Fall 2022
Homework 7

Released: October 28, 2022
Due: November 4th, 2022 @ 9pm ET
Late Due Date (1-48 hours late): November 6th, 2022 @ 9pm ET

Homework 7: BLUEbikes!

Introduction
The city of Boston and Blue Bikes provide remarkably complete data (https://www.bluebikes.com/system-data) about how Blue Bikes are used from month to month. We've downloaded the data for the month of September and filtered and cleaned it for you. (We made it a much smaller data set and kept only Sept 23 - 30th).

Blue Bikes are a vital piece of public transit infrastructure and it's important that Boston's amateur data science team (that's you!) evaluates how they are being used in order to make data-informed decisions about how to expand and upgrade the network.

Expectations
We expect you to write your code in a *modular style* using a well-defined collection of functions. The functions you implement are totally at your discretion. As always, you will be graded on a) correctness, b) code design and efficiency, and c) documentation and style. You may not use Pandas, numpy, or any machine learning library such as Scikit-Learn that would trivialize some parts of the implementation.

What you're practicing in this assignment:
- function/program design
- dictionaries (yes, you must use dictionaries everywhere we tell you that you must)
- lists, etc

Requirements:
1. **Read the data**. Load the `trips.csv` file, which contains all blue bikes trips in the last week of September, into a **list of dictionaries** where, for each dictionary, the column headers are the keys (strings) and the values are the values for the corresponding row. (You may use the code we wrote in class on 10/28 to create this list of dictionaries.) Read `stations.csv` file into a separate **dictionary** that maps station names to the GPS coordinates (latitude and longitude) for that station. Do not use the `csv` library. Write your own functions to read in both csv files.

   Warning! These are ***large*** files with many lines of data! To check if you've properly loaded them, test the length and take a look at the first few rows. You should have 445 stations and 141609 trips (yes, more than one hundred thousand).

   Here are the first two rows of the list of dictionaries from `trips.csv`:
   ```
   [{'duration': 602, 'start_day': 23, 'start_day_name': 'Friday', 'start_station':
   'MIT Vassar St', 'end_station': "Packard's Corner - Commonwealth Ave at Brighton
   Ave", 'bike_id': 8250}, {'duration': 466, 'start_day': 23, 'start_day_name':
   ```

```
'Friday', 'start_station': 'Ruggles T Stop - Columbus Ave at Melnea Cass Blvd',
'end_station': 'St. Alphonsus St at Tremont St', 'bike_id': 8335}]
```

Here are two examples of station entries you would have in your stations dictionary:

```
{'1200 Beacon St': [42.34414899, -71.11467361],
 '160 Arsenal': [42.36466403, -71.17569387],
… }
```

2. **Calculate the distance and speed of each trip**. Using the provided haversine distance function to calculate the distance between two stations. With the distance and trip time you can determine the trip speed in miles per hour. Note that the given trip durations are in seconds.

   Warning! Some trips are associated with stations that are not in your `stations.csv` file. Make sure that these trips do not cause your program to stop (for instance, by throwing an error). We recommend filling in a special value (e.g., None or "N/A") for these trips.

```
import math

# Earth's radius varies because the earth is not a perfect sphere.
# Use this number as it is considered the global average.
EARTH_RADIUS = 3959

def haversine_distance(start, end):
        """
        Calculates the distance in miles between two points on the earth's surface
        described by latitude and longitude.
        Parameters:
                start: list
                        list of two floats—latitude and longitude
                end: list
                        list of two floats—latitude and longitude
        Return:
                float - distance in miles between the two points
        """
        lat1 = start[0]
        long1 = start[1]
        lat2 = end[0]
        long2 = end[1]

        lat1 = math.radians(lat1)
        long1 = math.radians(long1)
        lat2 = math.radians(lat2)
        long2 = math.radians(long2)

        delta_lat = lat2 - lat1
        delta_long = long2 - long1

        # the earth's radius is a constant value
        a = math.sin(delta_lat / 2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(delta_long / 2)**2
        haversine = EARTH_RADIUS * 2 * math.asin(math.sqrt(a))
        return haversine
```

Hint: you may find it useful to add distance and speed information to each row in your data by inserting new keys into each dictionary with this information.

Hint 2: Here are examples of what your first two rows in your list of dictionaries might look like after this operation. (We've highlighted the added information for you).

```
[{'duration': 602, 'start_day': 23, 'start_day_name': 'Friday', 'start_station':
'MIT Vassar St', 'end_station': "Packard's Corner - Commonwealth Ave at Brighton
Ave", 'bike_id': 8250, 'dist': 1.0413475448513339, 'mph': 6.227327510738874},
{'duration': 466, 'start_day': 23, 'start_day_name': 'Friday', 'start_station':
'Ruggles T Stop - Columbus Ave at Melnea Cass Blvd', 'end_station': 'St. Alphonsus
St at Tremont St', 'bike_id': 8335, 'dist': 0.7073886058314272, 'mph':
5.464804680242785}]
```

And here's an example of what a row that we were unable to calculate distance and mph for might look like:

```
{'duration': 1231, 'start_day': 23, 'start_day_name': 'Friday', 'start_station':
'Graham and Parks School – Linnaean St at Walker St', 'end_station': 'Harvard
University Radcliffe Quadrangle at Shepard St / Garden St', 'bike_id': 6442,
'dist': None, 'mph': None}
```

3. **Visualize the distribution of distances and speeds for the trips (distances.pdf and speeds.pdf)**. Create two well-labeled histograms using matplotlib's `plt.hist()` function. This function takes a single list of values, then calculates the frequencies of these values for you. Feel free to use additional optional parameters to make the graph visually appealing to your eye.

   ```
   plt.hist(values, bins = 100)
   ```

   matplotlib hist documentation

   Hint: think about how you might be able to use a `get_column` or `extract_column` function to help you here. (Both professors have written these functions in `data_utils.py` and `dataproc.py`.)

   Hint 2: you don't want to include data from trips that we couldn't calculate a valid distance or speed for, so make sure to do something to filter these out! (But you won't be filtering out trips with a calculated distance of 0, that's a different case!)

4. **Create a dictionary that maps day of the week to the count of trips associated with that day of the week. Using only trips that end at the station "Forsyth St at Huntington Ave".** Print a report displaying these counts. This is the only required output of your program.

   For example, your output might look like:

   ```
   Number of trips ending at Forsyth St at Huntington Ave:
   Friday: ???
   Saturday: ???
   ...
   ```

   Note that we do not expect these to be in "standard week day" order.

5. **Extra credit (up to +9).** Create a visualization (singlebike.pdf) depicting the voyage of a single bike throughout the course of late September 2022. Your visualization should show the physical locations of

each start/end station that this bike came from/went to for each trip. Make sure to choose a bike that went on at least 50 trips in your trip data. Plot Northeastern and any other (0 to 3 other) geographic points in Boston/Cambridge/Somerville that will help your user understand where the stations are that the bike has traveled between.

The data is in order, top-to-bottom, first trip on September 23rd to last trip on September 30th.

All appearance choices are up to you! Make sure that your visualization is well-labeled and flex your creative muscles!

Program Structure

Reminders (we copy + pasted much of this from the style guide!) about what your program should look like:

```
""" Your Name
    course
    assignment
    date

    description of the program (1 - 2 sentences)

    your copy+pasted program output here
"""

import matplotlib.pyplot as plt
# other imports here

# constants go here
# NO other variables should be defined outside of functions

def example_function1(example_parameter):
    """
    Your function comment here. Must include all three of:
    what does it do, what parameters does it take, what value does it return
    """
    # your function code here

def example_function2(example_parameter1, example_parameter2):
    """
    Your function comment here
    """
    # your function code here

def main():
    # Your code goes here!
    # Your code goes here!

if __name__ == "__main__":
    main()
```

What to Submit

Submit your program (`bluebikes.py`), your visualizations (`distances.pdf`, `speeds.pdf`, and, optionally, `singlebike.pdf`). **In your program header, include the program text output produced when you run your program.**

Please Read: It is your responsibility to verify that your program runs (by examining the autograder output), and that ALL required files have been submitted to gradescope. If you forget to submit your code, your visualization, or your program doesn't run, your grade will be affected. If you *resubmit* your assignment for any reason, your resubmission must include ALL required files. Files from previous submissions are deleted from gradescope and not accessible to the graders! Resubmissions after the due date (but before the late due date) will receive the full late penalty.

No homework files will be accepted for grading or regrading after the late due date!