Northeastern University
DS2001 – Programming with Data — Social Science Practicum
Fall 2019, Mattsson & McCabe

Practicum 4: Big Digital Data

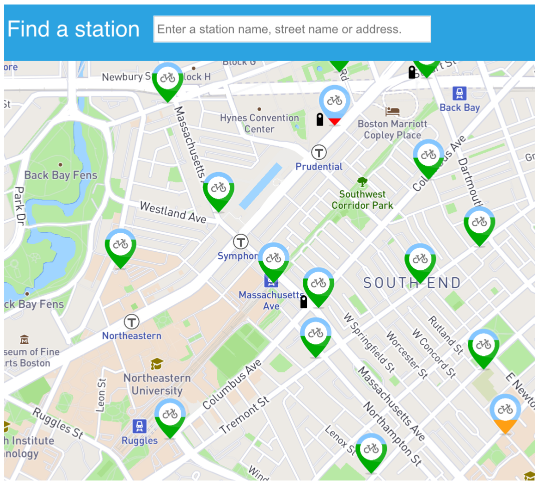
In this week's practicum we will be asking Python to process digital trace data for us. Specifically, we will be calculating the distance and speed of trips taken with Boston Blue Bikes. We will be working with a manageable-sized subset of data, but keep an eye out for pieces of code that would be extensible.

1 Commuting and Sightseeing on the Boston Blue Bikes

Modern digital infrastructure collects data about many aspects of our lives, every day. Analyzing, understanding, and putting such digital trace data to use has come to define Data Science as we know it today. Most often, digital trace datasets are collected under the course of standard operation for some system. Extracting useful information from data that wasn't collected for that purpose is tricky, but also an important component of modern data analysis.

The Blue Bikes system is owned by the municipalities of Boston, Brookline, Cambridge and Somerville, and operated by Motivate International, Inc. This company keeps track of trip durations so that it can charge customers the appropriate rate, and they keep track of where their bikes are taken in and out so that they can rebalance them as needed. The company also collects data to run an application that shows that availability of Blue Bikes and docks in real time: `https://member.bluebikes.com/map/`

We will use Boston Blue Bikes data from bike #5561 during August, 2019 to study urban biking patterns.



1.1 Download

Please download `pr04.zip` from the practicum website (or Blackboard), unzip it, and move the directory/folder to where you want it in your file system. Use Atom's `File > "Add Project Folder..."` to open this folder and view `bluebikes.py`.

1.2 View data for Boston Blue Bike #5561

First, we must load the dataset. You should see `from data import data_header, data_as_list_of_lists` at the top of your file. This tells `bluebikes.py` to load the custom functions from `data.py` named `data_header` and `data_as_list_of_lists`. These functions take as an argument the name of the file that you want to load. Please use `‘bluebike-5561.csv’`.

You should print out the dataset neatly and confirm that it looks like what you expect. Use the table below to note anything you might need to recall about the dataset, such as the types in each column.

Table 1: Boston Blue Bike #5561, August 2019

bikeid	**start/end_name**	**start/end_location**	**trip_duration**	**user_type**	**birth_year**	**gender**
5561	Kendall T	(42.362, -71.084)	1007	Subscriber	1999	1

Tuple data type A `tuple` is a data type in Python. A `tuple` is very similar to a `list`, with one important difference. You can index into and loop over a `tuple` the same way you would a `list`. What you cannot do is *change just one element* of a `tuple`. They are an immutable data type, which is a fancy way of saying that they are just like `strings` and `integers`: you can’t change just one letter or one decimal place. To change a `tuple`, you have to reassign the whole thing.

1.3 Distance of trips with bike #5561

The dataset gives us the coordinates, in latitude and longitude, of the station where each bike trip starts and ends. This is useful for the app developers who need to plot the stations on a map, and the employees replenishing bicycles who need to find the full/empty stations in person. But we are interested in urban biking patterns, and so we would like to know instead the distance of each trip.

In `bluebikes.py` you will find the function `distance(start_location,end_location)` that calculates the distance between two locations. The `arguments` to this function are two location coordinates, which are each a `tuple` consisting of a latitude and a longitude value. The distance is computed using the Haversine formula, where R is the radius of the earth:

$$d_{Haversine}(i,j) = 2R * \arcsin\sqrt{\sin\left(\frac{\Delta_{lat}}{2}\right)^2 + \cos\left(i_{lat}\right) * \cos\left(j_{lat}\right) * \sin\left(\frac{\Delta_{lon}}{2}\right)^2}$$

Please loop through the dataset and use this function to compute the distance of each trip. Add this number to the end of each `record`, as a new column. Please also update the `header` to include this new `"trip_distance"` column.

Do these numbers make sense? What are you expecting the Haversine distance function to give? Is this what you are getting? Please provide a brief comment on this question in your code.

1.4 Fastest and slowest trips with bike #5561

We are interested in how fast people are biking around Boston. Please define your own function `speed(trip_distance,trip_duration)` that calculates the average speed of the trip. You function should have two `arguments`, and should `return` the result in miles per hour. Use this formula:

$$speed = \frac{distance}{time}$$

Test your function. Once you have it working, loop through the dataset and use this function to find the fastest and slowest trips in the dataset. Print out the range of speeds that people have maintained while biking with Boston Blue Bike #5561.

User-defined functions One of the most useful tools of programming is the *user-defined function*. They take a piece of code and package it up neatly. This means that you can use this piece of code again and again without re-writing it, just like you do with `print(ARGUMENT)`. In Python, it is quite simple to define your own function. The syntax you would use goes like this:

```
def function_name(ARGUMENT_1,ARGUMENT_2):
    some code that turns ARGUMENT_1 and ARGUMENT_2 into VALUE
    return VALUE
```

1.5 Overall average speed with bike #5561

We would like to know, overall, how fast people are biking with this bike. Please find the average speed of trips taken with bike #5561.

Should you be including every trip? Please explain why you might (or might not) want to exclude trips in the dataset from this particular calculation. Write your explanation as a comment in your code.

1.5.1 ... and for any Boston Blue Bikes trip dataset

You just wrote code that calculated the average speed for a particular subset of a larger dataset. In the future, we may want to calculate this same value for a different subset of the Blue Bikes data, or even for the full dataset. Please turn your previous code into a function that `returns` the average speed. The `argument` that you give your function should be a dataset in the form of a list of trips.

1.6 Average biking speed of commuters vs. casual users

The future is now, actually. The Boston Blue Bikes offer several ways to use the system. Motivate International are building a model of bike-replenishment and they suspect that there could be important differences in how fast commuters and casual users get where they're going. Please create new datasets (ie. lists of trips) for these two types of riders, and calculate the average biking speed for each.

- `Subscribers` – Users with an annual pass to Boston Blue Bikes
- `Customers` – Users who purchased a single-ride or single-day pass

To what extent is the title of this section the question we are answering? Please write a short comment on this question as a comment in your code.

1.7 Submit Your Work!

Please submit your work to Blackboard.
If you used one file, you can upload it as-is.
If you used more, please first make a `zip` file of the folder within which you did your work.

- On MacOS, select your work folder, right-click and choose `Compress pr03`
- On Windows, select your work folder, right-click, `Send To`, `Compressed (zipped) folder`

1.8 Citations

`https://www.bluebikes.com/system-data`

2 Done Early?

2.1 Column indexing

Using a for loop each time you want to access the values of a column is very tedious. Write a function (using a for loop internally) that returns a list of all the values for a given column, then use that, in combination with the `median` function, to get the median birth year of riders of this bike.

2.2 Optional function parameters

Read up quickly on optional function parameters. See if you can edit your average speed function to accept an optional boolean argument that tells it whether or not you want to include zeros in the average.

2.3 Boston Blue Bike #????

Within `data.py` there are code comments that should let you produce the corresponding dataset for any of the Boston Blue Bikes. See if you can edit this code to do this exercise for a larger subset of the data.

This handout was originally created by Carolina Mattsson and Stefan McCabe, Fall 2019.