CODE GUIDE:
/shaiwal/Uber/ByTimezone

1. /code_scheduling
schedulecode.py
It is for collecting uber data at a specified time
schedulecode_merge.py
It is merging the json files at a specified time
schedulecode_merge.py
It is for collecting information about the 50 Random text.txt locations

2. /code_old
collectdataandvisualize.py
very initial code has functions to collect the uber data for fare and estimated time and also has keys inside the code.
Also, we can visulaize the data by making a table and making an html file.
..outtable.html
..estimatepricesjson.txt
..estimatestimejson.txt
collectdataver1.py --->
collectdataver2.py ---> **.... collectdata_v3.py** -----> **collectdata_v4.py**----> **collectdata_v5.py**
This final code version 5 will be used for further processing.

file2sqlite_v2.py ---> creates an sqlite with table
create loclist with id and its value so that we can query and get the list between two row numbers.
key2json.py---> converts the server token (keys for uber)txt file to json file with key_id:server_token

bulkrunningScripy.py ,---> for selecting a part of rows in the dataset and running the code in the background
locationNameIndexing.py,---> making a sqlite file
reading_from_sqlite.py--- > reading from sqlite file(making a query using sqlite lib in python)

3./code

CODE NAME	DESCRIPTION
bulkrunnignScript_6.py	Run the collectdata_v5_6.py for different inputs. (HardCoded according to Number of locations in Timezone 6)
bulkrunnignScript_10.py	Run the collectdata_v5_10.py for different inputs.(Timezone 10)
bulkrunnignScript_16.py	Run the collectdata_v5_16.py for different inputs.(Timezone 16)
bulkrunnignScript_20.py	Run the collectdata_v5_20.py for different inputs.(Timezone 20)
bulkrunnignScript_any.py timezone total_count	**Run the collectdata_v5_any.py for different inputs. (For all other timezone) total_count number of locations in that timezone**

checksurcharge.py	**Finds the % of timezone data which has surcharge.Timezon 6,10,16,20(Checked)**
collectdata_v5_6.py keyid idlow idhigh	Collect the UBER fare data for OD pairs which lie in timezone 6. Select OD pairs between idlow and idhigh using the keyid.
collectdata_v5_10.py keyid idlow idhigh	Collect the UBER fare data for OD pairs which lie in timezone 10.Select OD pairs between idlow and idhigh using the keyid.
collectdata_v5_16.py keyid idlow idhigh	Collect the UBER fare data for OD pairs which lie in timezone 16.Select OD pairs between idlow and idhigh using the keyid.
collectdata_v5_20.py keyid idlow idhigh	Collect the UBER fare data for OD pairs which lie in timezone 20.Select OD pairs between idlow and idhigh using the keyid.
collectdata_v5_any.py keyid idlow idhigh timezone	**Collect the UBER fare data for OD pairs which lie in the specified timezone**
diff_distribution.py	**Makes a json file with OD pair as key and value as the difference in fare bwteen uber and nyc. Will be used for plotting the CDF.**
diffmin_actualfare.py	**When calulated fare is less than minimum but have to pay the minimum fare , finding out how much profit. For Timezone 6,10,16,20 into a same plot.**
estimator.py timezone cab_service	Estimating the surcharge given a pickup location
estimator_v2.py timezone cab_service	Improved popularity estmator
estimator_v3.py timezone cab_service	Improved surcharge estmator
estimator_v4.py timezone cab_service	**Final Model with no cross validation**
estimator_v5.py timezone cab_service	**Final Model with 5 fold cross validation**
external_test.py keyid cab_service timezone	**For the random set of 50 locations comapare the UBER API Real surcharge and our estimated Average Surcharge for the hour. Give the keyid to use the UBER API**
file2sqlite_v2.py	**Simple create table locind (location text,id integer primary key) for each line in the txt file. Outputs a sqlite.**
findcandidates.py	**Using the candidat set, common locations between the four timezones(6,10,16,20) and using these candidated we can plot the difference fare distributon. PLOT THE CDF.**
heatmap_basemap.py	**Trying to make a static heatmap using basemap library python**

heatmap_google.py	**Csv to dynamic heatmap using Google API**
Heatmap_plot.R	**CSV to heatmap. For plotting the static heatmap in R.**
key2json.py	**Key file of Uber to a JSON file with keyid(integer) and value(server_token)**
lwr.py	**Locally Weighted Regression for Input of Dimension > 1 D**
lwr_1D.py	**Locally Weighted Regression for Input of Dimension = 1 D**
lyft_bulkrunnignScript.py timezone	**Run the lyft_generatekeyfile.py first and then lyft_collectdata_v5.py for only the candidate set out of Timezone 6,10,16,20**
lyft_collectdata_v5.py keyid idlow idhigh timezone	**Collects the data for the specified range of OD pairs for the specified timezone(LYFT)**
lyft_generatekeyfile.py	**Used for Outh2.0 Authentication, first the tokens are generated and then used for collecting data.**
lyft_mergejson.py	**Merges the json output generated by lyft_collectdata_v5.py**
lyft_zdiff_distribution.py	**Lyft fare minus NYC Fare(Make a json file with key as OD pair and values are differcne in between many lyft services and NYC fare)**
lyft_zfindcandidates.py	**Using the Differecne fare distribution , make a CDF and plot it.**
makepi_drop_profile.py	**Using the union of pickup locations and dropoff locations , genearte a profile such that key is lat_long and two more keys pickup and dropoff each is a list of size 24 containing the populairty for that timezone.**
makepi_drop_profile_v2.py start end	**Improved by sending queries in bulk. select dropoff,timezone,count(dropoff) from nyctaxi where dropoff in {loc} group by dropoff,timezone;**
mergejson_6.py	Just merging the Timezone 6 results
mergejson_10.py	Just merging the Timezone 10 results
mergejson_16.py	Just merging the Timezone 16 results
mergejson_20.py	Just merging the Timezone 20 results
mergejson_any.py timezone total_count	**total_count is number of locations in that timezone**
pi_dropbulk.py	**Running makepi_drop_profile_v2.py for diff . Inputs.**

pi_dropmergejson.py	**Merges all the json files for pi_drop profile into one**
svm_surcharge.py	**Assuming features = lat_long , target = 0,1(Yes/No) for surcharge ,used SVM for this classification.**
txttojson.py	**Reads the file line by line adn makes a json with key as line number as value as data on the line**
Uberminfare_check.py	**Validated whether the surcharge*base min_fare = current minimum fare**
Validate_faredata.py	**fare= (base fare + (price/distance)*distacnce + (price/minute)*(minutes))*surcharge** **if fare < minimum:** **fare = minimum** **This final fare** **check** **lies between the Low Estimate and High estimate.** **If yes, keep count of it and print the percentage**