Piyush Kumar

**Python Programming, Spring 2019**

Programming Project
Due: March 28, 2019, 11:59 pm

- Students must work **in pairs** for this assignment.
- You must submit a link to your git hub repository on Canvas as well as a tar file.
- Please keep your code private on github and share it with us only.
- The github account need to share your repository with is: "ahanagemini".

**Problem Statement**

You will be working with stock data for this project. In order to extract the stock information for a ticker, you should use the `iex-api-python` which is described here: `https://pypi.org/project/iex-api-python/`

Your task is to prepare a Python application consisting of four separate modules (four `.py` files). We can call the four modules the **Tickers** module, the **Fetcher** module, the **Query** module and the **Predictor** module. The details of the four modules are as follows:

- **Tickers:** This module must:
  - Have a function, say `save_tickers` that fetches the first $n$ **valid** tickers from the following URL and writes the tickers in a file, say `tickers.txt`:
    `http://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NASDAQrender=download`
  - To ensure that a ticker is **valid**, you should use the `iex-api-python` to verify that the price function for the Stock corresponding to the fetched ticker works. That is, if there are some tickers for which the price() function of the iex API does not work, then that ticker should not be written to the file.
  - Write one ticker symbol per line of the file `tickers.txt`.
  - The number $n$ will be provided as a system argument to the module. There should be a main module that calls the `save_tickers` function and passes the value of $n$ to it. The value of $n$ will be $\leq 150$.
- **Fetcher:** The Fetcher module must:
  - Read all the tickers from an input file (`tickers.txt`). The tickers are listed one per line.
  - Define a function that updates the current stock information for the ticker that is passed as an argument. The information is written and updated in an information file (say `info.csv` for example).
  - The module should call this function for each input ticker in `tickers.txt` file.
  - This module should run for specified time period, say `time_lim` in seconds and update the data in the information file.
  - The information file should be a csv file, with the first row being the column headers. For each ticker in the `tickers.txt` file, the information file, should have one row for each minute.
  - The `Time` column should contain time in the HH:MM format with HH ranging from 00 to 23. There should be one and only one row corresponding a specific value of `Time` and `Ticker`.
  - In order to extract the stock information for a ticker, say "AAPL", you should use the iex-api-python which is described here: `https://pypi.org/project/iex-api-python/`. You need to fetch the current data for the following fields: `low, high, open, close, latestPrice, latestVolume`. Use the `quote()` function of the Stock corresponding to the ticker.
  - The csv must have data in the format:
    `Time, Ticker, latestPrice, latestVolume, Close, Open, low, high`

- Store the time of the query and the respective keys and values in the `info.txt` file. For each iteration, during which you save the data for a specific minute, you may wait till the start of the next minute, say, 12:37 and then save the data for all tickers during that iteration with the `Time` field set to the minute (12:37).
- Please use the information on the API page to figure out how to install iex-api-python. The page also has the information for fetching necessary data about a stock ticker.
- The arguments passed to this module are: `time_lim, ticker_filename, info_filename`

- **Query:** The Query module must:
  - Define a function that takes the information file name, time and the ticker as the input and prints the details corresponding to a specific time and ticker symbol to the terminal. The data must be printed as follows:

```
field 1: value 1
field 2: value 2
.
.
.
field n: value n
```

  - The above function also takes a `verbose` flag, which when true, the number of rows and number of columns in the information file will be printed out as well as the names of the columns.
  - Should have a main that takes the following flags and calls the function to print the values:
    verbose
    time (HH:MM format)
    file (information file that contains the information)
    ticker

- **Predictor:** The Predictor module must:
  - Define a function that takes the information file name, a ticker, a column name (say *col* which is either `latestPrice` or `latestVolume`), a time range, $t$ and a graph file name as the input.
  - The function is called by main, which accepts the arguments and passes those to the function.
  - The function must read the data from the information file, select the data for the specified ticker, and train a machine learning based model to predict the value of the specified column for the next $t$ minutes.
  - You should only use the `Time` column (you can split it into hours and minutes) to predict the value of *col* based on the historical data stored in the information file.
  - You must use the linear regression in `sklearn.linear_model` to train using the historical data for the specific ticker stored in the information file and then predict the result for the next $t$ minutes.
  - You must also plot and save the plot for the historical variation in the value of *col* as well as the predicted values. The actual and predicted data should be plotted out on the same graph in two different colours.

Please put the following five files in a folder, tar them in a tar file named **Lastname1_LastName2.tar**:

1. tickers.py: A .py file for the Tickers module. To execute this code, we will use the following command:
   python3 tickers.py number_of_tickers ticker_filename
2. fetcher.py: A .py file for the Fetcher module. To execute this code, we will use the following command:
   python3 fetcher.py time_lim ticker_filename info_filename
3. query.py: A .py file for the Query module. To execute this, the command used will be:
   python3 query.py –verbose True/False –file info_filename –ticker ticker –time time The time format should be HH:MM. Example is: 13:31.
4. predictor.py: A .py file for the Predictor module. To execute this code, we will use the following command:
   python3 predictor.py ticker info_filename graph_filename col t

5. README: Documentation and execution instructions.

**NOTE:**
You **MUST** write your codes so that the above commands can execute them exactly.
**In case your ticker code does not work you will get a zero for the entire project except the points for github. In case your fetcher code does not work you will only receive points for the github and the ticker portion. In case your query or your predictor codes do not work you will get a zero for the part corresponding to the relevant code.**

The rubric for grading is:

- Ticker module: 20 points
- Fetcher module: 25 points
- Query module: 20 points
- Predictor module: 25 points
- github: 10 points: We will check that you checked your code in regularly and that it is not that your entire project was checked in at one time. Your files also must have been checked in part by part over time. Both partners must contribute almost equally.