NT Lab 11: Stock Market App (100 points)

Objectives

- Practice importing external modules
- Gain experience using a web-URL API to get information
- Practice with CSV text and lists
- Practice writing data to output files

Introduction

In this Lab, you will build an app that displays daily stock prices obtained through an Application Program Interface (API) from an online service that provides free stock-market information.

Step 1. Get AlphaVantage API Key

This app gets stock prices from a Web-API provided by Alpha Vantage, a company that provides free online stock info. To get stock-prices from their Web-API, you have to register at their website to get an API key. It is free and they won't spam you. Here are the steps to get your API key:

1. In your web browser, go to this website: https://www.alphavantage.co (".co", not ".com" because Alpha Vantage is in Britain).
2. Click on "GET YOUR FREE API KEY TODAY" (center of page).
3. Fill out the form:
   - "Which of the following best describes you?": Choose "Student" from the menu.
   - "Organization": enter "University of San Francisco".
   - Give your email address (don't worry; they won't spam you).
   - Click "I'm not a robot".
4. Click "GET FREE API KEY". Your API key will appear on the page below. Copy it and SAVE IT SOMEWHERE. You will paste this key into the URL your app uses to query the API for stock prices.

Step 2. Check if *requests* module is already installed, and if not, install it.

1. To check if the **requests** module is already installed, open your terminal and enter python3 to start the Python interpreter. The Python **>>>** prompt appears.
2. Enter **import requests** to see if you already have the HTTP **requests** module installed. If you have it, the Python prompt will redisplay, so you can skip to Step 2.3.

3. If you don't have the module, you will get an error saying the module is not found. Enter **exit()** to exit the Python interpreter and return to the command line, then enter **pip3 install requests** to install the **requests** module.

Step 3. Download starter code (stock_market.py**) and edit it to add your code.**

getStock() function

The starter code includes a function getStock(). It takes (as a parameter) the symbol for a stock, and calls requests.get() with a request for prices of the specified stock from the AlphaVantage API. This function is complete except it needs an API key. Copy your AlphaVantage API key into the getStock() function where indicated. (Note: Your key must be inside the quotes.) Other than adding your API key, **do not edit** the getStock() function.

The stock data returned from Alpha Vantage's Web-API includes a lot of data you don't need. The only data you need for this stock app is in the text part, so getStock() extracts that. If a stock with the specified symbol exists, the text part of the data is two lines of text like that shown below for MSFT, the symbol for Microsoft stock:

```
symbol,open,high,low,price,volume,latestDay,previousClose,change,changePercent
MSFT,107.7900,108.6600,107.7800,108.4100,10957711,2019-02-19,108.2200,0.1900,0.1756%
```

The first line is a comma-separated list of the names of all the pieces of data in the results. The second line is a comma-separated list of the data corresponding to each name. You can think of this as a table with ten columns, a row of headings, and a row of data, e.g.,

symbol	open	high	low	price	volume	latestDay	previousClose	change	changePercent
MSFT	107.7900	108.6600	107.7800	108.4100	10957711	2019-02-19	108.2200	0.1900	0.1756

In this data, "open" is today's opening price of the stock, "high" is today's highest price, "low" is today's lowest price, "price" is the current price, and "previousClose" is yesterday's closing price.

main() function

The starter code includes a main() function that is called when the app starts. Add code to make it:

1. Start by displaying this message: "Check stock prices."
2. Ask the user for a stock symbol.
3. Call getStock() with the stock symbol, and save the returned stock-price data.
4. Split the data (two text lines) into a **list** of two strings (*hint: use string method .split()*).
5. If the split data is a list of two strings (each in CSV format), the data is good and the stock was found. (*Hint: if the stock symbol was valid, the length of the split data should be 2.*) Extract the stock's previous closing price, opening price, high price, low price,

and current price. Do it in a way that will still work if AlphaVantage changes the order of items in the returned data. E.g., don't assume that the stock's opening price is in the second position (index 1) in the list, because if your program *assumes* that and tomorrow AlphaVantage moves the opening price to another position, your app won't work anymore. Write your app so it **finds** each price no matter how they are ordered, and uses the appropriate index to get each price.

6. Print the previous closing price, opening price, high price, current price, and low price.
7. If the split data **is not** a list of exactly 2 items (CSV strings), that means the stock symbol was not found. In that case, tell the user the stock symbol was not found.
8. When the app displays prices for a stock (but not when it can't find a stock), it should also write the same information to an output file **stock_prices.txt**. Each time the app successfully displays the prices for a stock, the app should append the information to the file. E.g., if the user requests prices for three stocks, when the application quits, stock_prices.txt should contain info for those three stocks, with a line like this separating them: ===========================
9. The app should not clear and replace the data in **stock_prices.txt** each time it is run. It should keep **appending** stock prices to the file.
10. The program should ask the user "Check another stock price? (Y or N)", and should perform input validation to make users enter a valid response. If the response is "Y" or "y", the program should loop back to Step 2 (not Step 1) and ask for another stock symbol. If the response is "N" or "n", the program closes the output file and quits.

Example of program's *screen* output

```
Check stock prices.

Enter stock symbol: IBM

Stock Symbol: IBM

Last Close: 133.2600

Open: 133.7600

High: 133.9300

Low: 132.2700

Current: 133.2300

Check another stock price? (Y or N): y

Enter stock symbol: aapl

Stock Symbol: AAPL

Last Close: 122.1500
```

```
Open: 123.6600
High: 124.1800
Low: 122.4900
Current: 123.0000
Check another stock price? (Y or N): x
Please enter Y or N.
Check another stock price? (Y or N): n
MacBook1E01C872:test jjohnson$
```

Example of program's *file* output

```
Stock Symbol:IBM
Last Close: 133.2600
Open: 133.7600
High: 133.9300
Low: 132.2700
Current: 133.2300
=========================
Stock Symbol:AAPL
Last Close: 122.1500
Open: 123.6600
High: 124.1800
Low: 122.4900
Current: 123.0000
=========================
```

Submitting

After testing your program, upload these files to NT Lab 11 in Canvas:

- **stock_market.py**: The source code for your web app.
- **stock_prices.txt:** The output file from your program after it has found prices for several stocks.

- **README.txt**: Explains how the program works, any resources you used, what doesn't work, and the challenges you had in creating it.