DeVry University

College of Engineering and Information Sciences

1. Python Stock Tracking Project

Module 5

# Background

1. The project will provide students with experience creating applications in Python. Students will use object-oriented techniques to develop a stock tracking application. By processing the historical stock data, profit/loss reports can be generated. Data storage will allow users to save and retrieve stock data. The system will use the Python libraries to create charts.

**In this part of the project, we use the matplotlib to create a chart showing stock prices over time.**

# Objectives – Module 5

1. Use the pyplot class in the matplotlib library to create a simple stock chart.

# Steps

1. **Always test as you go!**

Investors often use charts to visualize stock price history. Python has an extensive library that can be used to enhance the base Python language. You can use any of these modules in your own programs.

To create our stock chart we will be using the matplotlib library. This library makes it easy to create many types of visualizations from simple line charts to more complex and interactive graphics.

Consider exploring the many examples at <https://matplotlib.org/>. You can also find more information there about using matplotlib.

## 1. Implement display\_stock\_chart()

* 1. Open the **stock\_menu.py** program you have been working on.

Make sure that there is an **import matplotlib.pyplot as plt** line near at the top of the file. The starter file already has this but it’s a good idea to check. This line will import the pyplot class located in matplotlib and create an instance called **plt** which will be our plot object.

### 1.1 The following pseudocode can be used to create a stock chart.

The function display\_stock\_chart(stock\_list, symbol) accepts two parameters. The first is our list of stocks. The second is the stock symbol for the stock we want to see on our chart.

**display\_stock\_chart(stock\_list, symbol)**

create a list called **date**

create a list called **price**

create a list called **volume**

set **company** to “”

for each **stock** in **stock\_list**

if **stock**.**symbol** = **symbol**

set **company** to **stock**.**name**

for each **dailyData** in **stock**.**DataList**

**date**.append(**dailyData.date**)

**price**.append(**dailyData.close**)

**volume**.append(**dailyData.volume**)

call **plt.plot()** passing **date**, **price**

call **plt.xlabel()** passing ‘Date’

call **plt.ylabel()** passing ‘Price’

call **plt.title()** passing **company**

call **plt.show()**

## 2. Implement display\_ chart()

* 1. We will be working in the **stock\_menu.py** file for this part of the project so load this file into your IDE.

### 2.1 The following pseudocode can be used to create a stock chart that will call the display\_stock\_chart code.

The function display\_ chart(stock\_list) will call the display\_stock\_chart() function based on the symbols chosen by the user.

**display\_ chart(stock\_list)**

output “Stock List: [“ (do not start a new line at the end)

for each **stock** in **stock\_list**

output **stock.symbol** + “ “ (do not start a new line at the end)

output “]” (end the stock list)

input **symbol** (convert to upper case)

set **found** to False

for each stock in stock\_list

if **stock.symbol** = **symbol** then

set **found** to true

current\_stock = stock

if **found** = True

call **display\_stock\_chart()** and pass in **stock\_list, symbol**

else

output error message for symbol not found

pause and prompt the user to press Enter to continue

### 1.3 Test the Code

Let’s create a chart!

Run the **stock\_menu.py** file.

Enter in at least one stock with 4 Daily Data points.

Return to the main menu and select Show Chart.

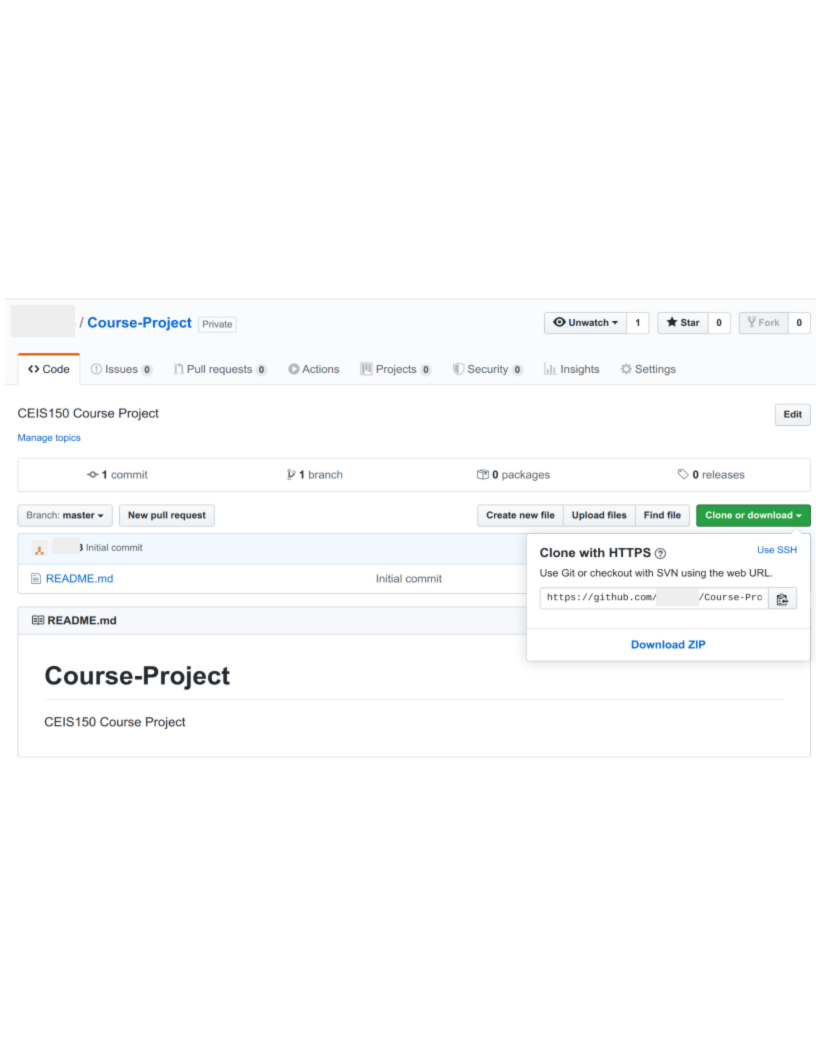
Enter a stock symbol from the stock list.

You should see a chart. In VS Code the chart may open in another window. If it’s hidden by the VS Code window, you may have to click on the other window to see the chart. Spyder users may see the chart appear in the plots tab or in the main output window. Click on the tab to see the chart.

# Deliverables

1. **To submit this part of the project:**
2. 1. Run the program (stock\_menu.py).
3. 2. Create a chart.
4. 3. Take a screen shot of the chart.
5. 4. Paste the screen shot into the appropriate slide in the PowerPoint template file provided in the Project area for this module.
6. 5. Submit the file in the Assignments area.

# Optional

1. Push your updated project files to GitHub. This will provide a backup for your project should anything happen to your local files. It will also allow you to go back through your files to see a change history. If you accidentally modify or overwrite code, you can always find an earlier version of the file.
2. ****
3. Find your repository url on GitHub. Then navigate to your project folder and use the following commands. Replace X with the week/module number you are submitting. Replace {your url}with the address of your GitHub repository.

The following commands will: stage the changes, commit the changes, and push the updates to GitHub.

1. **git add --all**
2. **git commit -m "Module X"** ←Change X to the Module you are submitting.
3. **git push {your url}** ←Change {your url} to the url for your repository on GitHub.

**Note: VS Code users can use the Source Control tab on the left to stage, commit, and push updates to GitHub.**

1. **Note: Use the Snipping Tool built into Windows to take screen shots (click Start and search for Snipping Tool)**
2. **To submit this part of the project:**
3. 1. Run the program (stock\_console.py).
4. 2. Create a chart.
5. 3. Take a screen shot of the chart.
6. 4. Paste the screen shot into the appropriate slide in the PowerPoint template file provided in the Project area for this module.
7. 5. Submit the file in the Assignments area.