



# inStock Advanced

Python Stock Tracking Project

By Miguel Mastache

10/28/22

# Introduction

---

inStock is a simple stock tracking application. It is capable of importing historical stock information, from either Yahoo! Finance, via CSV or web scraping. This app is able to save your stock purchase history, and produce reports based on your purchase history.

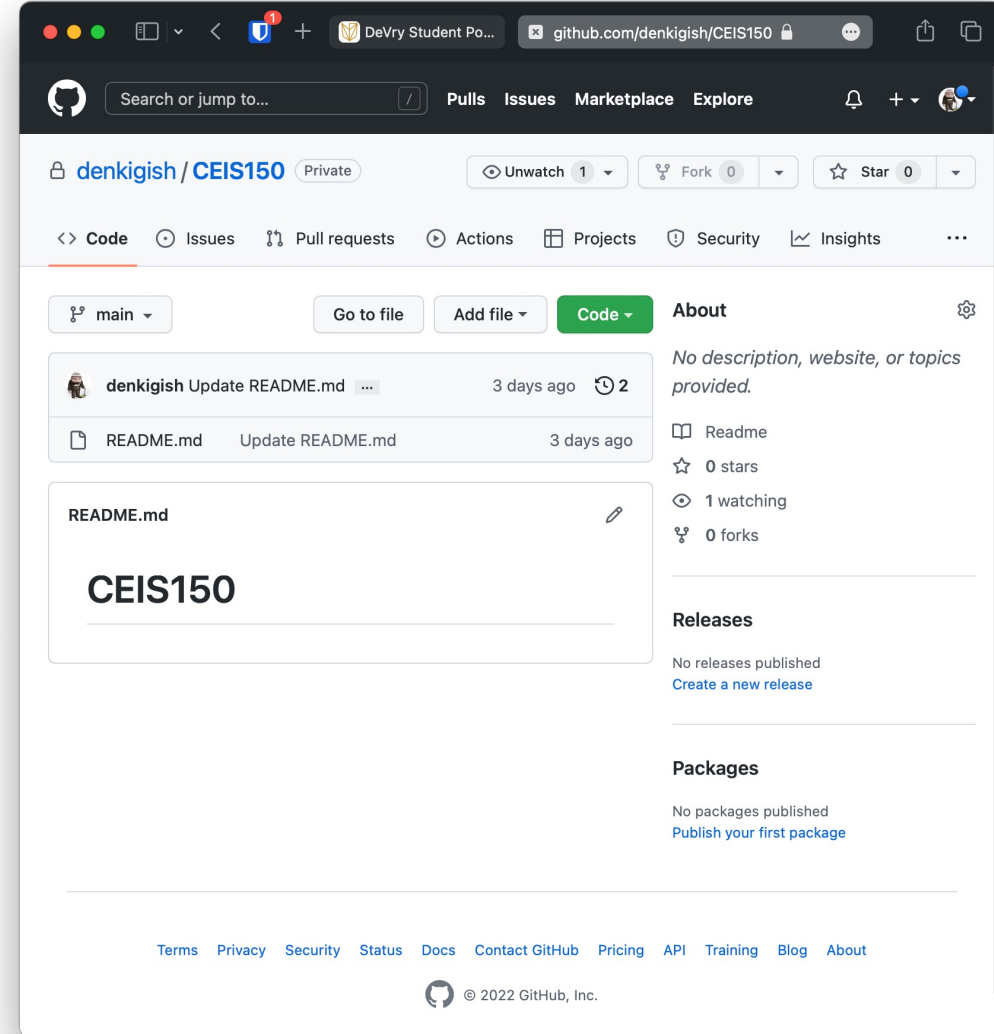
# Environment

---

This part of the project focus' on setting up the environment used to develop the stock tracking application.

# GitHub

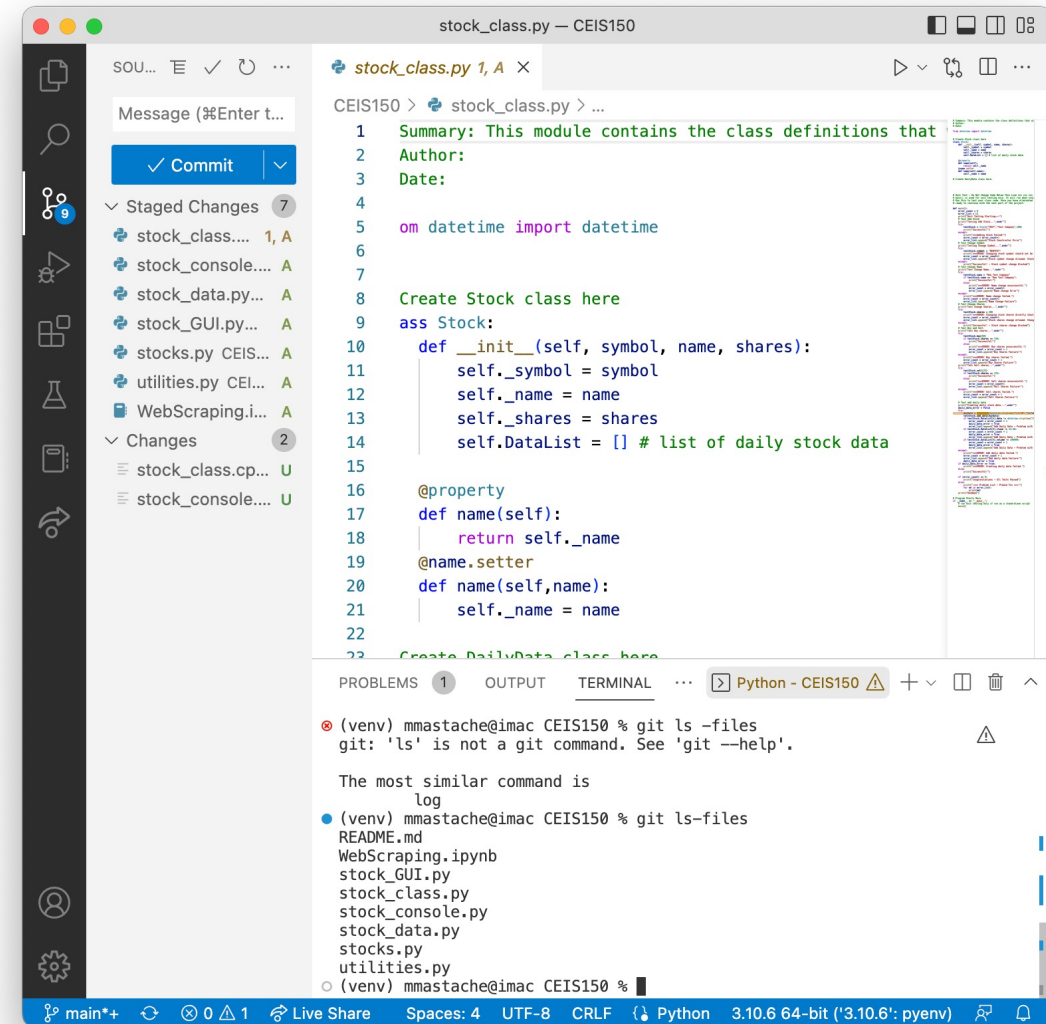
- Screen shot of GitHub project repository.
- For this class, I setup a GitHub repository. It can be found at:
- <https://github.com/denkigish>





# IDE & Starter Files

- Screen shot of your IDE (Spyder or VS Code) with the project Starter Files loaded.
- Also chose to use the VS Code IDE for development of the project. Shown is VS Code with the starter files loaded.



The screenshot shows the Visual Studio Code IDE interface. The main editor window displays the file `stock_class.py` with the following code:

```
1 Summary: This module contains the class definitions that
2 Author:
3 Date:
4
5 from datetime import datetime
6
7
8 Create Stock class here
9 class Stock:
10     def __init__(self, symbol, name, shares):
11         self._symbol = symbol
12         self._name = name
13         self._shares = shares
14         self.DataList = [] # list of daily stock data
15
16     @property
17     def name(self):
18         return self._name
19     @name.setter
20     def name(self, name):
21         self._name = name
22
23 Create DailyData class here
```

The left sidebar shows the Explorer view with a file tree containing:

- Staged Changes (7)
- stock\_class.py (1, A)
- stock\_console.py (A)
- stock\_data.py (A)
- stock\_GUI.py (A)
- stocks.py (CEIS150, A)
- utilities.py (CEIS150, A)
- WebScraping.ipynb (A)
- Changes (2)
- stock\_class.py (U)
- stock\_console.py (U)

The bottom panel shows the Terminal window with the following output:

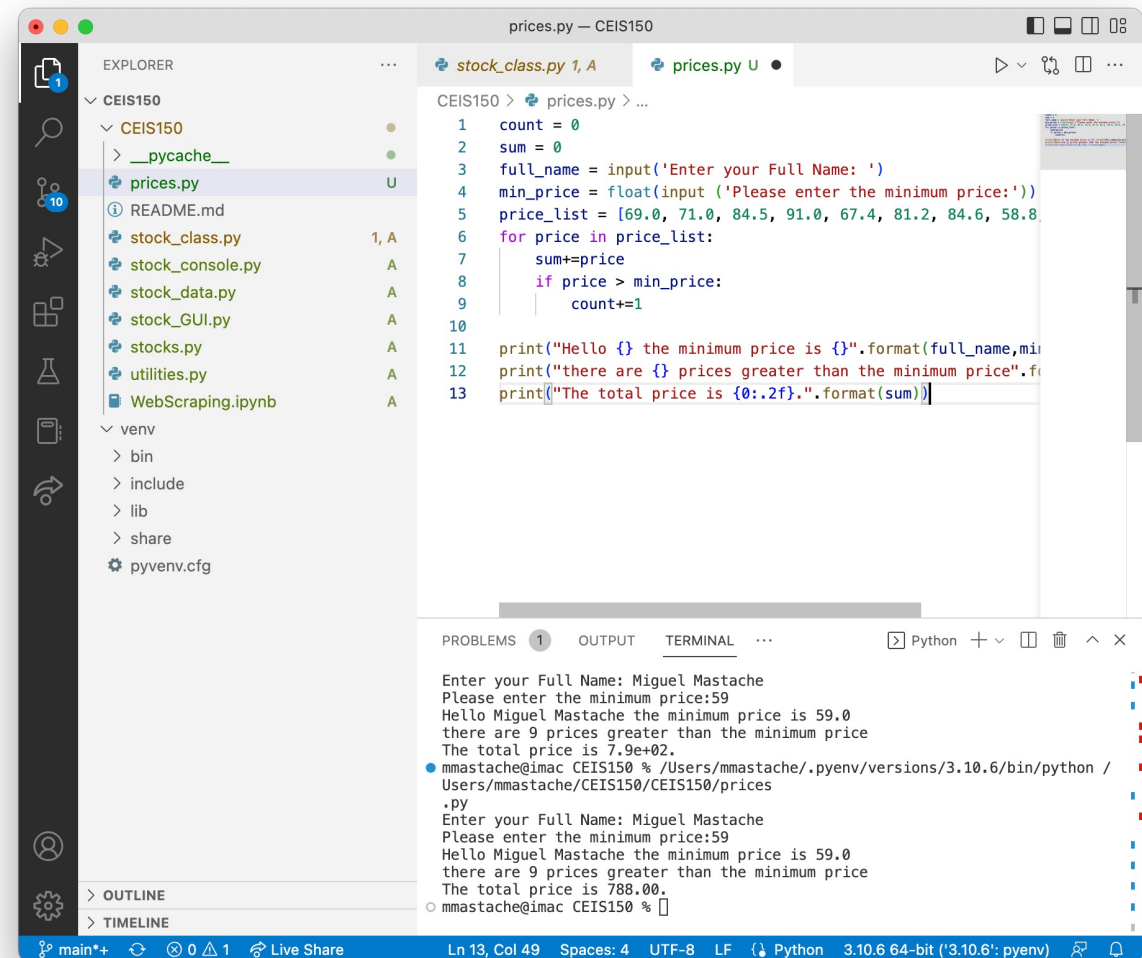
```
(venv) mmastache@imac CEIS150 % git ls -files
git: 'ls' is not a git command. See 'git --help'.

The most similar command is
  log
(venv) mmastache@imac CEIS150 % git ls-files
README.md
WebScraping.ipynb
stock_GUI.py
stock_class.py
stock_console.py
stock_data.py
stocks.py
utilities.py
(venv) mmastache@imac CEIS150 %
```

The status bar at the bottom indicates the current workspace is `main*`, using `UTF-8` encoding, `CRLF` line endings, and the `Python 3.10.6 64-bit` interpreter.

# Program

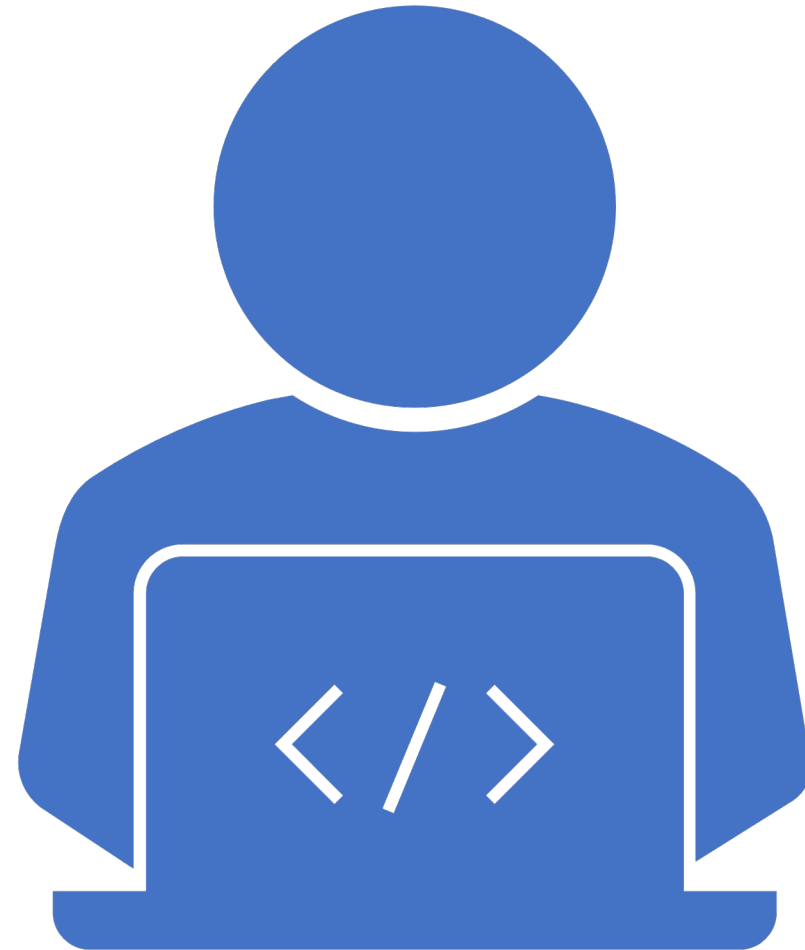
- Screen shot of Python program running successfully.
- prices.py working with price list, summing, and counting stocks above minimum price.



```
prices.py — CEIS150
EXPLORER
CEIS150
  CEIS150
    __pycache__
    prices.py
    README.md
    stock_class.py
    stock_console.py
    stock_data.py
    stock_GUI.py
    stocks.py
    utilities.py
    WebScraping.ipynb
  venv
    bin
    include
    lib
    share
    pyvenv.cfg
stock_class.py 1, A
prices.py U
CEIS150 > prices.py > ...
1 count = 0
2 sum = 0
3 full_name = input('Enter your Full Name: ')
4 min_price = float(input('Please enter the minimum price:'))
5 price_list = [69.0, 71.0, 84.5, 91.0, 67.4, 81.2, 84.6, 58.8]
6 for price in price_list:
7     sum+=price
8     if price > min_price:
9         count+=1
10
11 print("Hello {} the minimum price is {}".format(full_name,min
12 print("there are {} prices greater than the minimum price".fo
13 print("The total price is {:.2f}".format(sum))
PROBLEMS 1 OUTPUT TERMINAL ... Python + - - -
Enter your Full Name: Miguel Mastache
Please enter the minimum price:59
Hello Miguel Mastache the minimum price is 59.0
there are 9 prices greater than the minimum price
The total price is 7.9e+02.
• mmastache@imac CEIS150 % /Users/mmastache/.pyenv/versions/3.10.6/bin/python /
Users/mmastache/CEIS150/CEIS150/prices
.py
Enter your Full Name: Miguel Mastache
Please enter the minimum price:59
Hello Miguel Mastache the minimum price is 59.0
there are 9 prices greater than the minimum price
The total price is 788.00.
○ mmastache@imac CEIS150 % █
main*+ 0 1 Live Share Ln 13, Col 49 Spaces: 4 UTF-8 LF Python 3.10.6 64-bit ('3.10.6': pyenv)
```

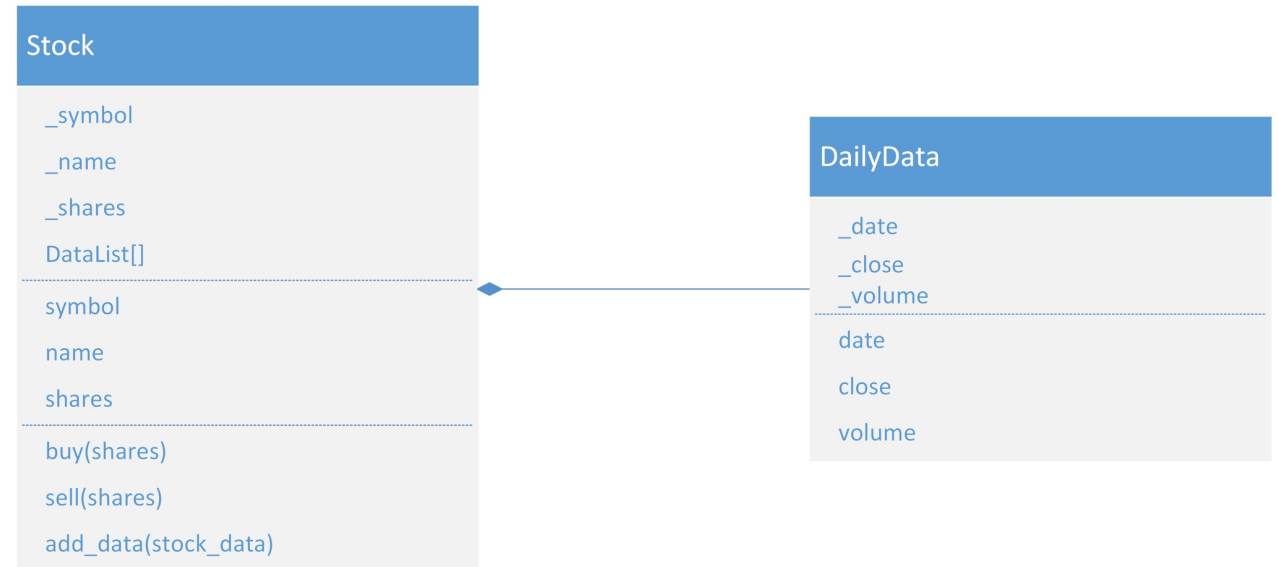
# UML and Class Definition

This section covers the development of the classes needed to work with our stock and history data. These classes are the foundation of my program.



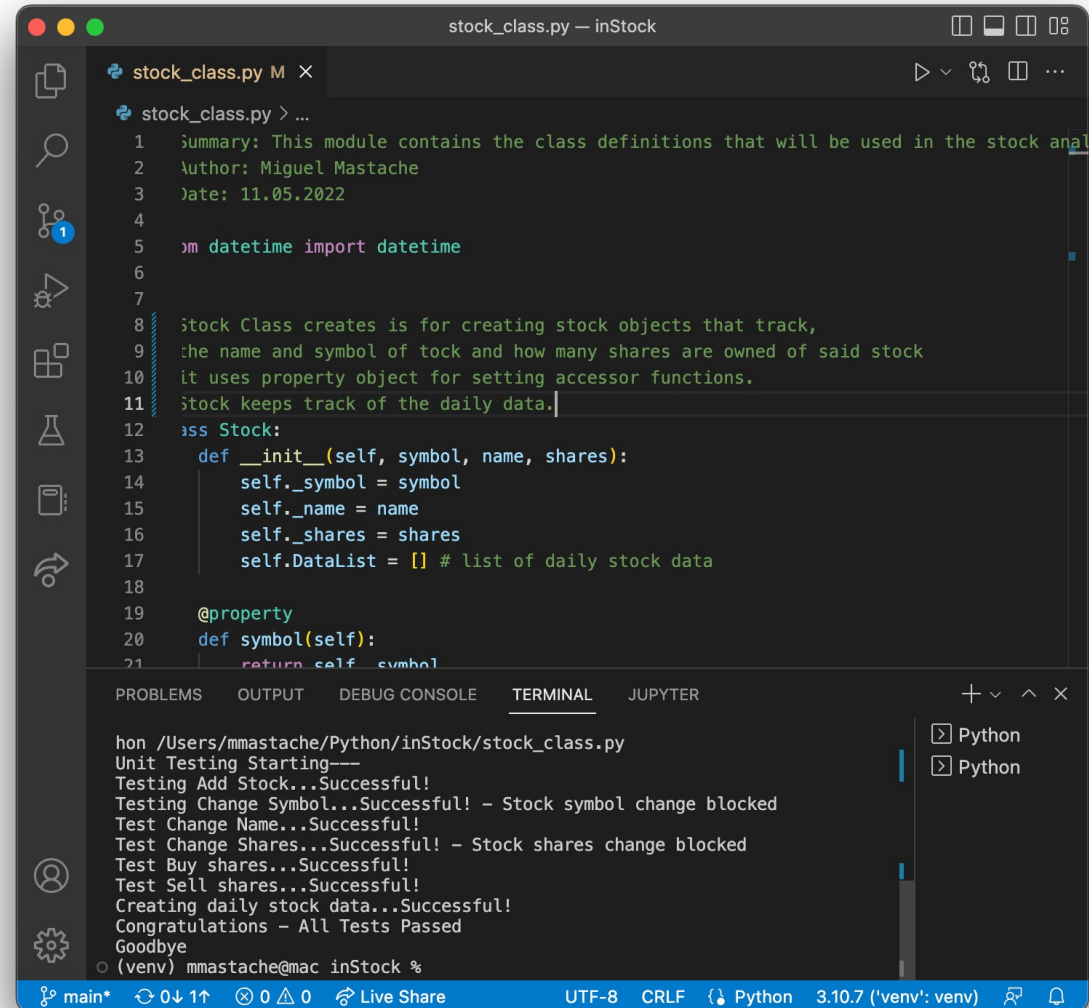
# Class Diagram

- Paste your Visio Class Diagram



# Class Code

- Screen Shot of your stock\_class.py file.



```
stock_class.py — inStock
stock_class.py M X
stock_class.py > ...
1 Summary: This module contains the class definitions that will be used in the stock anal
2 Author: Miguel Mastache
3 Date: 11.05.2022
4
5 from datetime import datetime
6
7
8 Stock Class creates is for creating stock objects that track,
9 the name and symbol of tock and how many shares are owned of said stock
10 it uses property object for setting accessor functions.
11 Stock keeps track of the daily data.
12 class Stock:
13     def __init__(self, symbol, name, shares):
14         self.symbol = symbol
15         self.name = name
16         self.shares = shares
17         self.DataList = [] # list of daily stock data
18
19     @property
20     def symbol(self):
21         return self.symbol
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

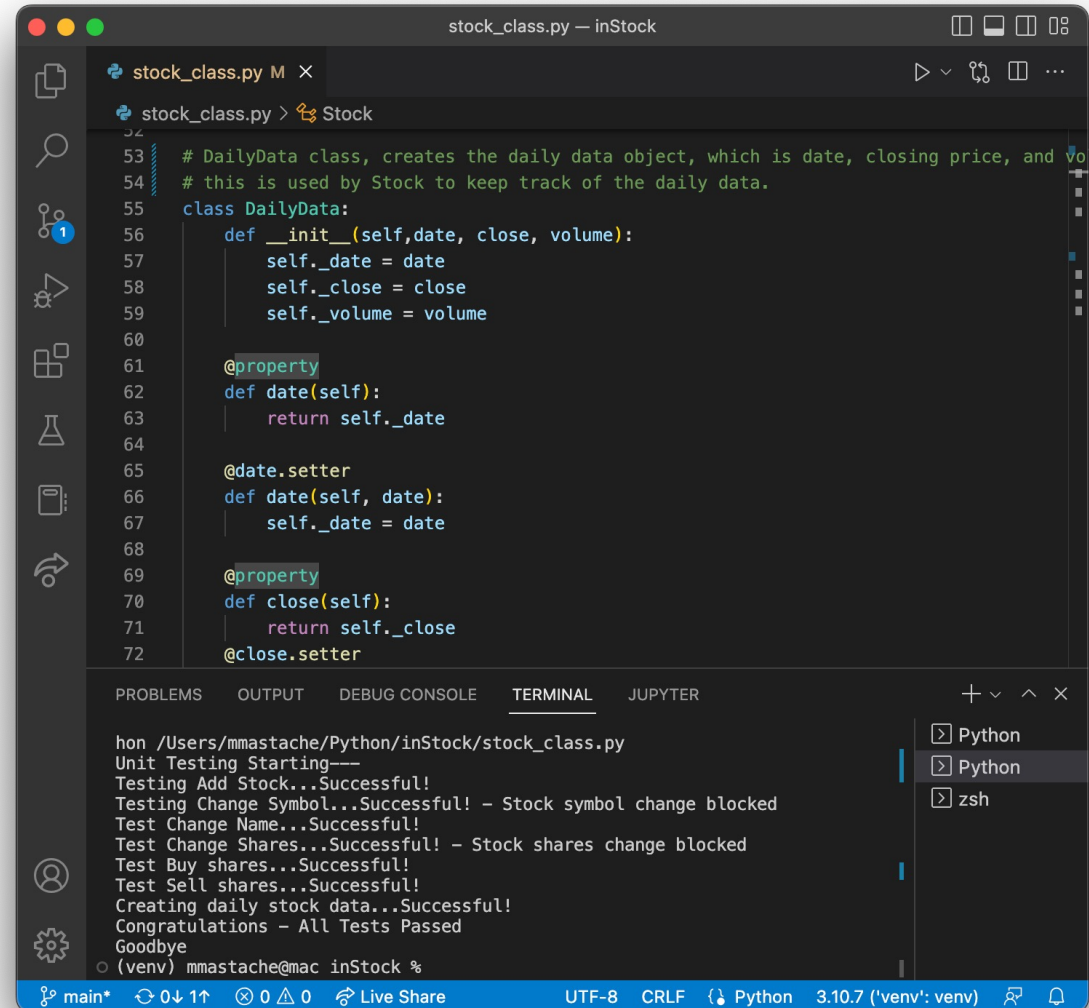
```
hon /Users/mmastache/Python/inStock/stock_class.py
Unit Testing Starting---
Testing Add Stock...Successful!
Testing Change Symbol...Successful! - Stock symbol change blocked
Test Change Name...Successful!
Test Change Shares...Successful! - Stock shares change blocked
Test Buy shares...Successful!
Test Sell shares...Successful!
Creating daily stock data...Successful!
Congratulations - All Tests Passed
Goodbye
(venv) mmastache@mac inStock %
```

main\* 0↓1↑ 0△0 Live Share UTF-8 CRLF Python 3.10.7 ('venv': venv)



# Unit Test

- Screen Shot of your successful unit test.



The screenshot shows a code editor window titled "stock\_class.py - inStock". The editor displays the following Python code:

```
53 # DailyData class, creates the daily data object, which is date, closing price, and vo
54 # this is used by Stock to keep track of the daily data.
55 class DailyData:
56     def __init__(self, date, close, volume):
57         self._date = date
58         self._close = close
59         self._volume = volume
60
61     @property
62     def date(self):
63         return self._date
64
65     @date.setter
66     def date(self, date):
67         self._date = date
68
69     @property
70     def close(self):
71         return self._close
72     @close.setter
```

Below the code editor, the terminal output shows the results of a unit test:

```
hon /Users/mmastache/Python/inStock/stock_class.py
Unit Testing Starting---
Testing Add Stock...Successful!
Testing Change Symbol...Successful! - Stock symbol change blocked
Test Change Name...Successful!
Test Change Shares...Successful! - Stock shares change blocked
Test Buy shares...Successful!
Test Sell shares...Successful!
Creating daily stock data...Successful!
Congratulations - All Tests Passed
Goodbye
(venv) mmastache@mac inStock %
```

The terminal output indicates that all tests passed successfully. The status bar at the bottom of the editor shows "main\*", "0 ↓ 1 ↑", "0 △ 0", "Live Share", "UTF-8 CRLF", "Python 3.10.7 ('venv': venv)", and other system icons.

# Console Interface

---

This section covers the creation of console-based interface for working with stocks and the stock price history. It develops the code for adding stock, and its daily data.

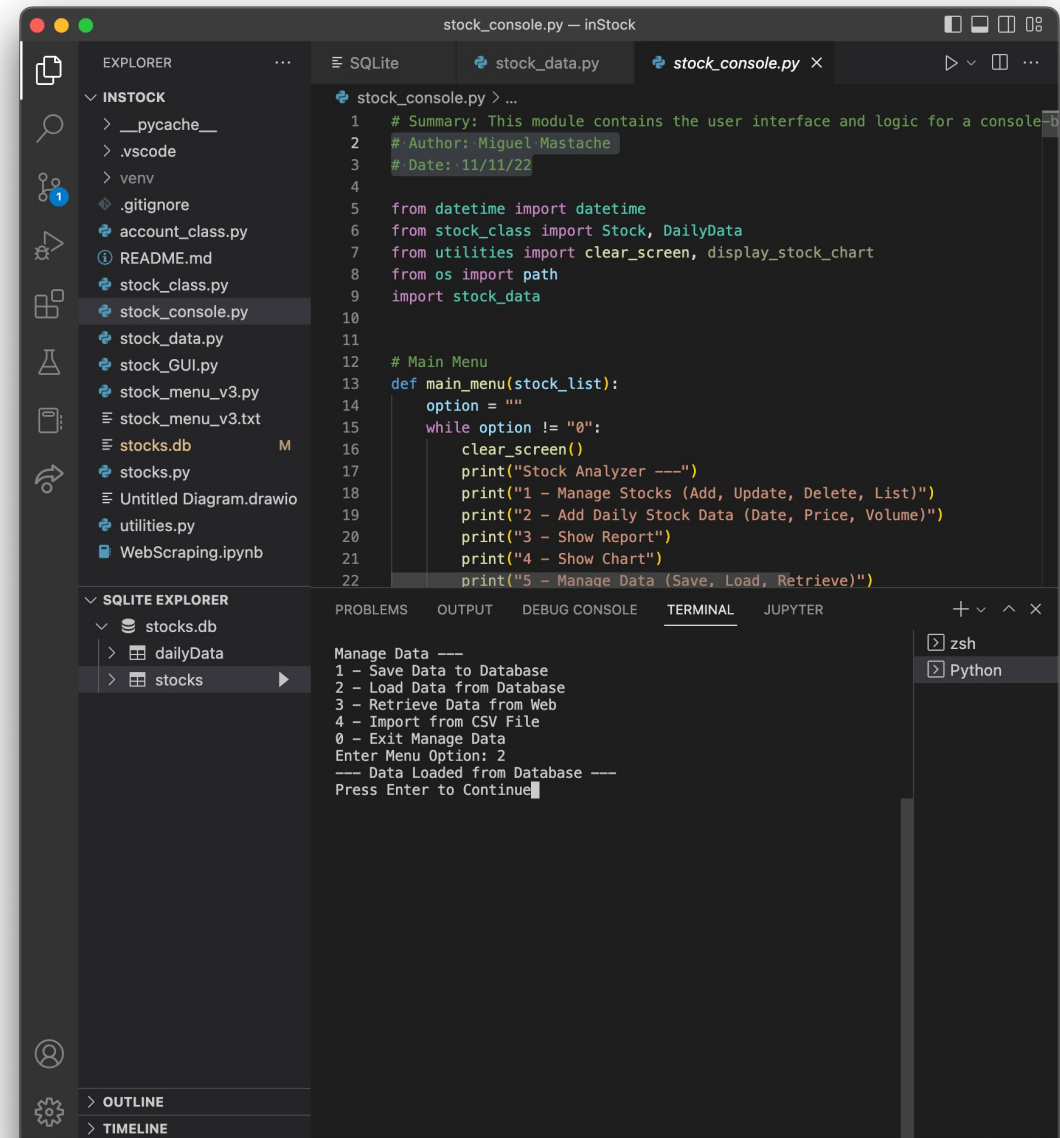
# Report

- Paste a screen shot of your working Stock Report.

```
python
0 - Exit Program
Enter Menu Option: 3
Stock Report ---
Report for: F Ford
Shares: 100.0
11/11/22 50.0 50.0
11/12/22 60.0 60.0
Summary --- 11/11/22 to 11/12/22
Low Price: $50.00
High Price: $60.00
Average Price: $55.00
Low Volume: 50.0
High Volume: 60.0
Average Volume: 55.0
Starting Price: $50.00
Ending Price: $60.00
Change in Price: $10.00
Profit/Loss:$1,000.00
Report for: GM General Motors
Shares: 100.0
11/11/22 70.0 70.0
11/12/22 70.7 80.0
Summary --- 11/11/22 to 11/12/22
Low Price: $70.00
High Price: $70.70
Average Price: $70.35
Low Volume: 70.0
High Volume: 80.0
Average Volume: 75.0
Starting Price: $70.00
Ending Price: $70.70
Change in Price: $0.70
Profit/Loss:$70.00
Report for: TSLA Tesla
Shares: 100.0
11/11/22 99.99 100.0
11/12/22 700.0 1.0
Summary --- 11/11/22 to 11/12/22
Low Price: $99.99
High Price: $700.00
Average Price: $400.00
Low Volume: 1.0
High Volume: 100.0
Average Volume: 50.5
Starting Price: $99.99
Ending Price: $700.00
Change in Price: $600.01
Profit/Loss:$60,001.00
Report Complete
*** Press Enter to Continue ***
```

# Load Database

- Paste a screen shot of your Data Loaded from Database message.



The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** A file tree for a project named 'INSTOCK'. The file 'stock\_console.py' is selected.
- SQLITE EXPLORER:** A view showing a SQLite database named 'stocks.db' with two tables: 'dailyData' and 'stocks'.
- Code Editor:** The 'stock\_console.py' file is open, showing Python code. The code includes a main menu function that handles various options, including 'Load Data from Database' (option 2).
- TERMINAL:** The terminal output shows the execution of the program. The user has entered '2' at the menu prompt, and the program has successfully loaded data from the database. The output is: 

```
Manage Data ---
1 - Save Data to Database
2 - Load Data from Database
3 - Retrieve Data from Web
4 - Import from CSV File
0 - Exit Manage Data
Enter Menu Option: 2
--- Data Loaded from Database ---
Press Enter to Continue
```

# Report

- Paste a screen shot of your working Stock Report.

The screenshot shows a VS Code editor window titled "stock\_console.py - inStock". The Explorer sidebar on the left shows a project structure with files like "stock\_console.py", "stock\_data.py", "stock\_GUI.py", "stock\_menu\_v3.py", "stock\_menu\_v3.txt", "stocks.db", "stocks.py", "utilities.py", and "WebScraping.ipynb". The SQL Explorer sidebar shows a database named "stocks.db" with tables "dailyData" and "stocks". The main editor displays the code for "stock\_console.py", which includes a main menu function. The terminal at the bottom shows the output of the program, displaying a summary of stock data for TSLA Tesla from 11/17/22 to 11/19/22, including metrics like Low Price, High Price, Average Price, Low Volume, High Volume, Average Volume, Starting Price, Ending Price, Change in Price, and Profit/Loss. The output also shows a table of daily stock data.

```
stock_console.py > ...
1 # Summary: This module contains the user interface and logic for a console-
2 # Author: Miguel Mastache
3 # Date: 11/11/22
4
5 from datetime import datetime
6 from stock_class import Stock, DailyData
7 from utilities import clear_screen, display_stock_chart
8 from os import path
9 import stock_data
10
11
12 # Main Menu
13 def main_menu(stock_list):
14     option = ""
15     while option != "0":
16         clear_screen()
17         print("Stock Analyzer ----")
18         print("1 - Manage Stocks (Add, Update, Delete, List)")
19         print("2 - Add Daily Stock Data (Date, Price, Volume)")
20         print("3 - Show Report")
21         print("4 - Show Chart")
22         print("5 - Manage Data (Save, Load, Retrieve)")
```

```
Summary --- 11/17/22 to 11/19/22
Low Price: $4.00
High Price: $6.00
Average Price: $5.00
Low Volume: 200.0
High Volume: 400.0
Average Volume: 300.0
Starting Price: $4.00
Ending Price: $6.00
Change in Price: $2.00
Profit/Loss:$2.00
Report for: TSLA Tesla
Shares: 1.0
11/17/22 6.0 600.0
11/18/22 7.0 200.0
11/19/22 8.0 300.0
Summary --- 11/17/22 to 11/19/22
Low Price: $6.00
High Price: $8.00
Average Price: $7.00
Low Volume: 200.0
High Volume: 600.0
Average Volume: 366.6666666666667
Starting Price: $6.00
Ending Price: $8.00
Change in Price: $2.00
Profit/Loss:$2.00
Report Complete
*** Press Enter to Continue ***
```



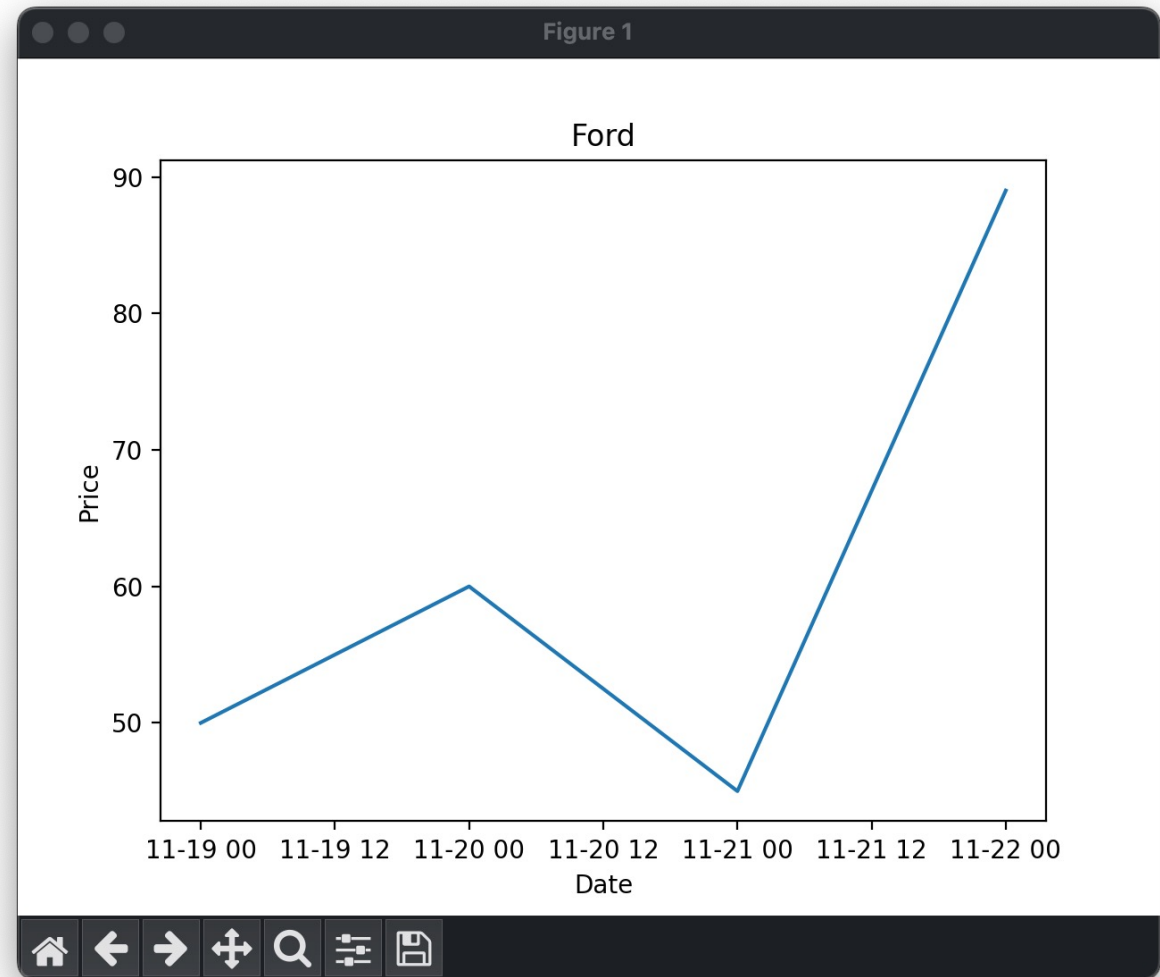
# Charts

In this section, we use the matplotlib library to display charts for the stock data.



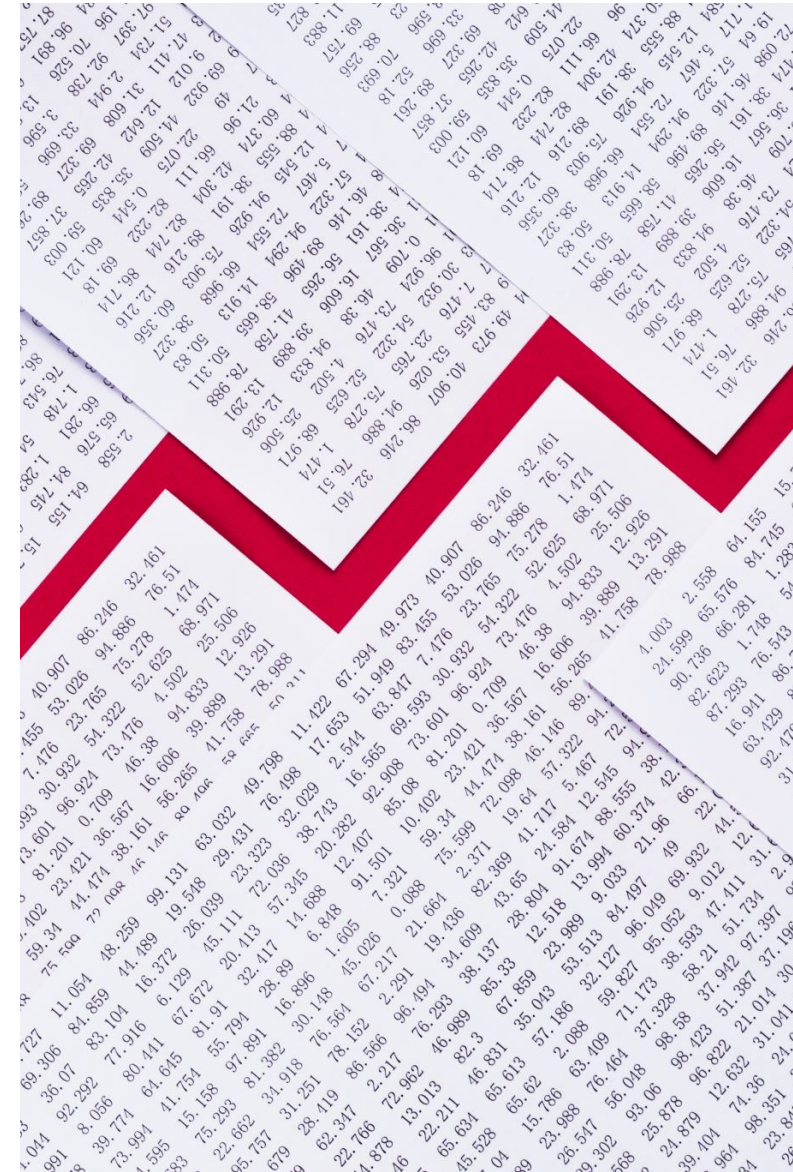
# Optional Chart

- Paste a screen shot of your stock chart.



# File Processing

This section covers file processing, by developing file saving code, and importing of saved data in csv format.



# Code

The image shows a screenshot of a Visual Studio Code editor window. The editor is displaying a Python file named `stock_menu_v3.py` with the following code:

```
4
5 @author: D99003734
6 """
7 # Author: Miguel Mastache
8 # Date: 12/3/22
9 from datetime import datetime
10 from stock_class import Stock, DailyData
11 from account_class import Traditional, Robo
12 import matplotlib.pyplot as plt
13 import csv
14
15
16 def add_stock(stock_list):
17     option = ""
18     while option != "0":
19         print("Add stock ---")
20         symbol = input("Enter Ticker Symbol: ").upper()
21         name = input("Enter Company Name: ")
22         shares = float(input("Enter Number of Shares: "))
23         new_stock = Stock(symbol, name, shares)
24         stock_list.append(new_stock)
25         option = input("Stock Added - Enter to Add Another Stock or 0 to Stop: ")
26
27 # Remove stock and all daily data
```

The terminal window at the bottom shows the output of the script, which includes the following text:

```
Profit/Loss:$115.87
2022-12-01 14.41 14360500.0
Low Price: $12.10
High Price: $127.97
Average Price: $37.77
Low Volume: 999999999999
High Volume: 86940400.0
Average Volume: 14323483.266932271
Change in Price: $115.87
Profit/Loss:$115.87
2022-12-02 14.62 10800800.0
Low Price: $12.10
High Price: $127.97
Average Price: $37.68
Low Volume: 999999999999
High Volume: 86940400.0
Average Volume: 14309504.365079366
Change in Price: $115.87
Profit/Loss:$115.87
**** No daily history

Report for: TSLA TESLA
Shares: 1.0
**** No daily history

---Report complete ---
Press Enter to Continue ***
```

The status bar at the bottom of the editor shows the current file is `main*`, the cursor is at `Ln 18, Col 25`, and the environment is `Python 3.10.7 ('venv': venv)`.

# File

- Paste a screen shot of the file downloaded from Yahoo finance

```
-zsh
-rw-r--r--@ 1 mmastache staff 9568 Dec 3 18:55 stock_menu_v3.py
-rw-r--r--@ 1 mmastache staff 2570 Nov 9 20:36 stock_menu_v3.txt
-rw-r--r-- 1 mmastache staff 20480 Nov 22 13:16 stocks.db
-rw-r--r-- 1 mmastache staff 422 Nov 5 17:34 stocks.py
-rw-r--r-- 1 mmastache staff 1162 Nov 22 13:05 utilities.py
drwxr-xr-x 7 mmastache staff 224 Oct 29 09:50 venv
mmastache@mac inStock % ls -l
total 248
-rw-r--r--@ 1 mmastache staff 17683 Dec 3 08:08 AFRM.csv
-rw-r--r-- 1 mmastache staff 10 Oct 29 09:07 README.md
-rw-r--r-- 1 mmastache staff 2339 Nov 5 17:34 Untitled Diagram.drawio
-rw-r--r-- 1 mmastache staff 5910 Nov 5 17:34 WebScraping.ipynb
drwxr-xr-x 6 mmastache staff 192 Nov 22 13:13 __pycache__
-rw-r--r-- 1 mmastache staff 4740 Nov 15 20:21 account_class.py
-rw-r--r-- 1 mmastache staff 7746 Nov 5 17:34 stock_GUI.py
-rw-r--r-- 1 mmastache staff 6806 Nov 22 11:43 stock_class.py
-rw-r--r-- 1 mmastache staff 13183 Nov 22 13:18 stock_console.py
-rw-r--r-- 1 mmastache staff 4029 Nov 19 19:17 stock_data.py
-rw-r--r--@ 1 mmastache staff 9568 Dec 3 18:55 stock_menu_v3.py
-rw-r--r--@ 1 mmastache staff 2570 Nov 9 20:36 stock_menu_v3.txt
-rw-r--r-- 1 mmastache staff 20480 Nov 22 13:16 stocks.db
-rw-r--r-- 1 mmastache staff 422 Nov 5 17:34 stocks.py
-rw-r--r-- 1 mmastache staff 1162 Nov 22 13:05 utilities.py
drwxr-xr-x 7 mmastache staff 224 Oct 29 09:50 venv
mmastache@mac inStock %
```



# File

- Paste a screen shot of the file downloaded from Yahoo finance

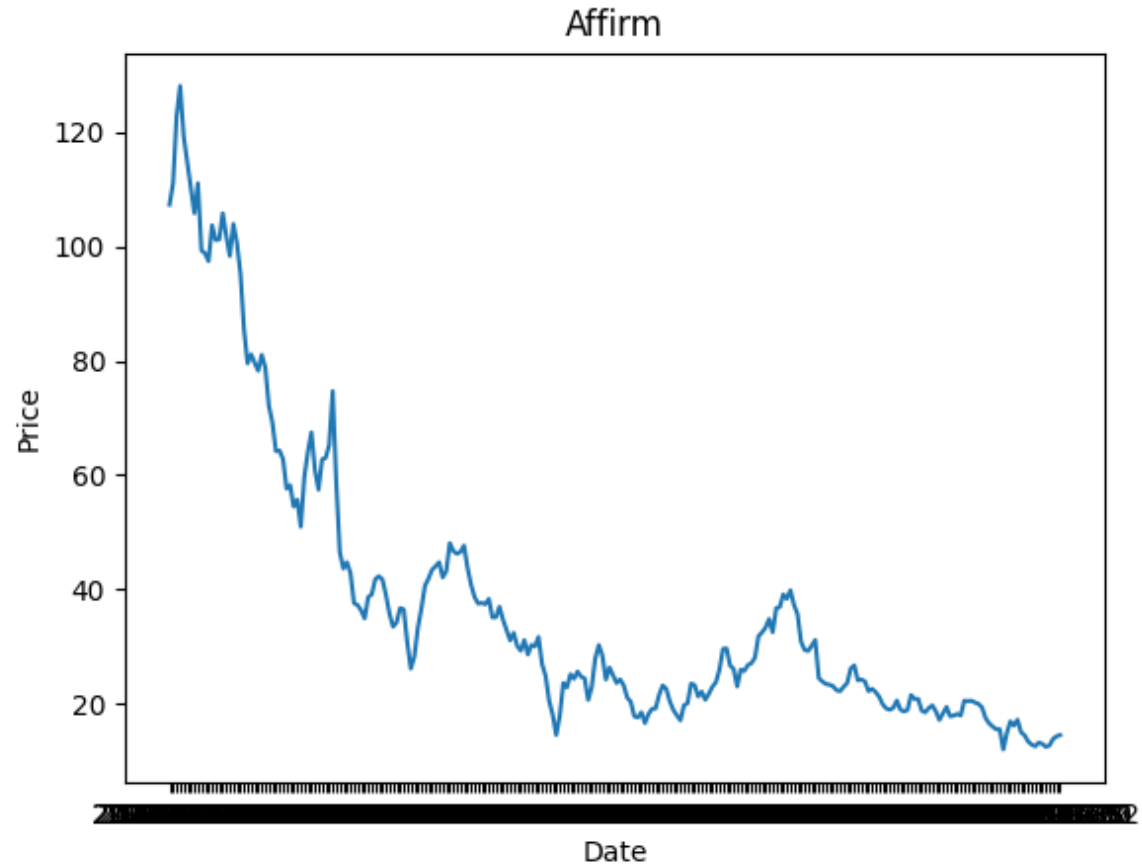
The screenshot shows a code editor window titled "AFRM.csv - inStock". The main editor area displays a CSV file with the following content:

```
1 Date,Open,High,Low,Close,Adj Close,Volume
2 2021-12-03,114.870003,114.870003,103.839996,107.239998,107.239998,12534
3 2021-12-06,103.010002,114.959999,96.440002,110.949997,110.949997,947690
4 2021-12-07,116.555000,124.110001,116.250000,122.730003,122.730003,89686
5 2021-12-08,122.555000,128.860001,117.820000,127.970001,127.970001,78684
6 2021-12-09,126.489998,127.889999,115.849998,119.129997,119.129997,81711
7 2021-12-10,119.129997,122.000000,112.230003,114.680000,114.680000,63516
8 2021-12-13,113.980003,116.889999,107.669998,109.959999,109.959999,97364
9 2021-12-14,104.558998,108.849998,101.349998,105.769997,105.769997,11541
10 2021-12-15,103.000000,113.190002,101.669998,110.980003,110.980003,11070
11 2021-12-16,113.000000,113.320000,92.059998,99.239998,99.239998,19807300
12 2021-12-17,95.690002,102.389999,92.334999,98.769997,98.769997,16414200
13 2021-12-20,94.595001,99.610001,93.510002,97.370003,97.370003,7297800
14 2021-12-21,98.839996,104.129997,95.620003,103.629997,103.629997,7591700
15 2021-12-22,100.943001,105.400002,99.160004,101.070000,101.070000,436970
16 2021-12-23,100.809998,102.070000,96.330002,101.160004,101.160004,451980
17 2021-12-27,102.260002,108.199997,102.260002,105.730003,105.730003,81181
18 2021-12-28,104.820000,107.000000,99.924004,101.769997,101.769997,534360
19 2021-12-29,101.239998,102.099998,96.519997,98.279999,98.279999,5499600
20 2021-12-30,98.419998,107.570000,97.561996,103.870003,103.870003,6357800
21 2021-12-31,102.900002,105.739998,100.500000,100.559998,100.559998,37289
22 2022-01-03,102.000000,102.209999,94.750000,95.209999,95.209999,8442700
23 2022-01-04,94.688004,95.514999,80.089996,85.410004,85.410004,21958300
24 2022-01-05,82.970001,85.680000,78.910004,79.529999,79.529999,10118700
25 2022-01-06,78.889999,83.680000,74.360001,81.099998,81.099998,10471300
26 2022-01-07,79.419998,83.390999,77.564003,79.620003,79.620003,6723600
27 2022-01-10,76.750000,78.750000,71.703003,78.239998,78.239998,10784400
28 2022-01-11,77.430000,83.599998,76.699997,81.029999,81.029999,9699800
29 2022-01-12,83.029999,84.680000,78.639999,78.790001,78.790001,6972100
```

The editor interface includes a file explorer on the left showing the project structure, a terminal at the bottom, and a status bar at the very bottom. A notification bubble is visible in the bottom right corner of the editor area.

# Importing data

- Screenshot of the historical data import



# Importing data

- Screenshot of the historical data import

The screenshot displays a Jupyter Notebook environment with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'INSTOCK' with various files including 'AFRM.csv', 'stock\_class.py', 'stock\_console.py', 'stock\_data.py', 'stock\_GUI.py', 'stock\_menu\_v3.py', 'stock\_menu\_v3.txt', 'stocks.db', 'stocks.py', 'utilities.py', and 'WebScraping.ipynb'. The code editor shows the following Python code:

```
4
5 @author: D99003734
6 """
7 # Author: Miguel Mastache
8 # Date: 12/3/22
9 from datetime import datetime
10 from stock_class import Stock, DailyData
11 from account_class import Traditional, Robo
12 import matplotlib.pyplot as plt
13 import csv
14
15
16 def add_stock(stock_list):
17     option = ""
18     while option != "0":
19         print("Add stock ----")
20         symbol = input("Enter Ticker Symbol: ").upper()
21         name = input("Enter Company Name: ")
22         shares = float(input("Enter Number of Shares: "))
23         new_stock = Stock(symbol, name, shares)
24         stock_list.append(new_stock)
25         option = input("Stock Added - Enter to Add Another Stock or 0 to Stop: ")
26
27 # Remove stock and all daily data
```

The terminal output shows the following text:

```
Average Price: $37.68
Low Volume: 999999999999
High Volume: 86940400.0
Average Volume: 143809504.365079366
Change in Price: $115.87
Profit/Loss:$115.87
**** No daily history

Report for: TSLA TESLA
Shares: 1.0
**** No daily history

---Report complete ---
Press Enter to Continue ***
Stock Analyzer ---
1 - Add Stock
2 - Delete Stock
3 - List stocks
4 - Add Daily Stock Data (Date, Price, Volume)
5 - Show Chart
6 - Investor Type
7 - Load Data
0 - Exit Program
Enter Menu Option: 5
Stock List: [AFRM TESLA ]
Which stock do you want to use?: AFRM
Press Enter to Continue ***]
```

The status bar at the bottom indicates the current position is at line 18, column 25, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.10.7, and the environment is 'venv:venv'.

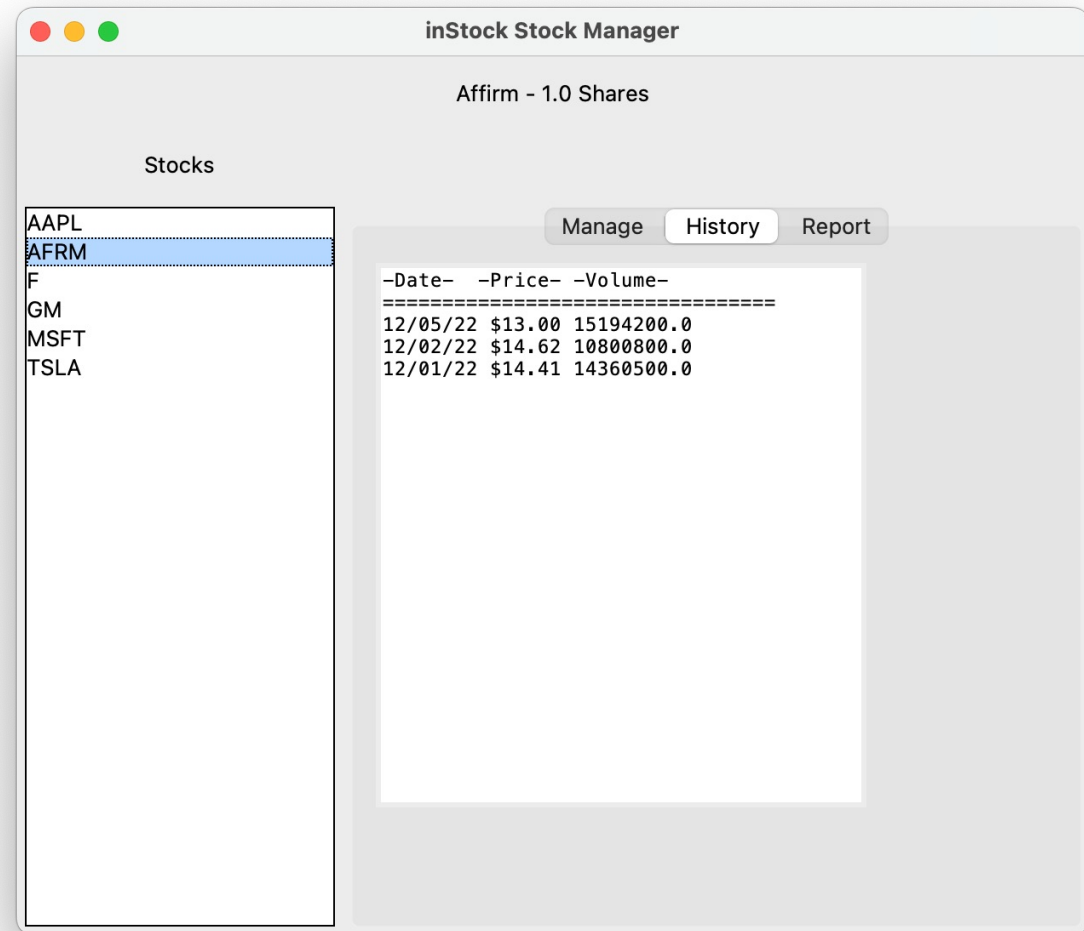


# GUI

This section covers the development of the graphical user interface for the stock tracking application.

# History Tab

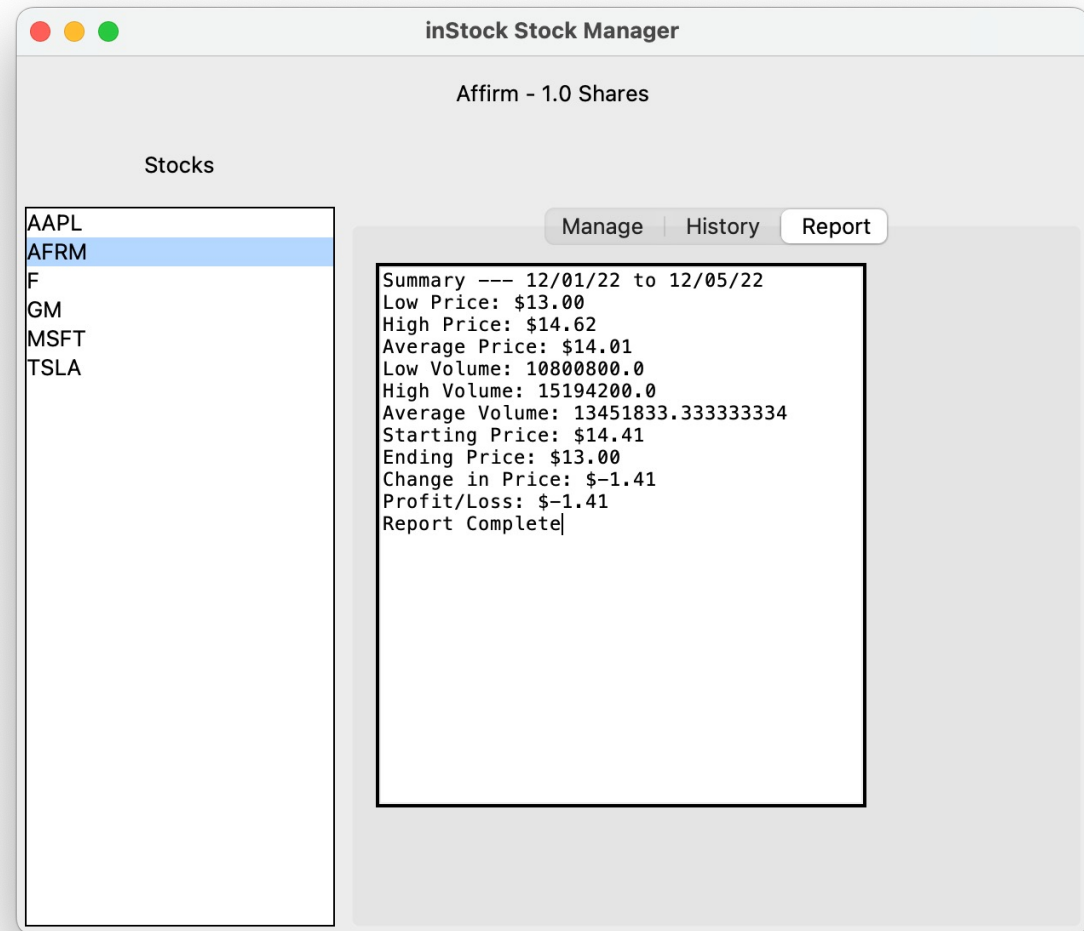
- Paste a screen shot of your history tab.





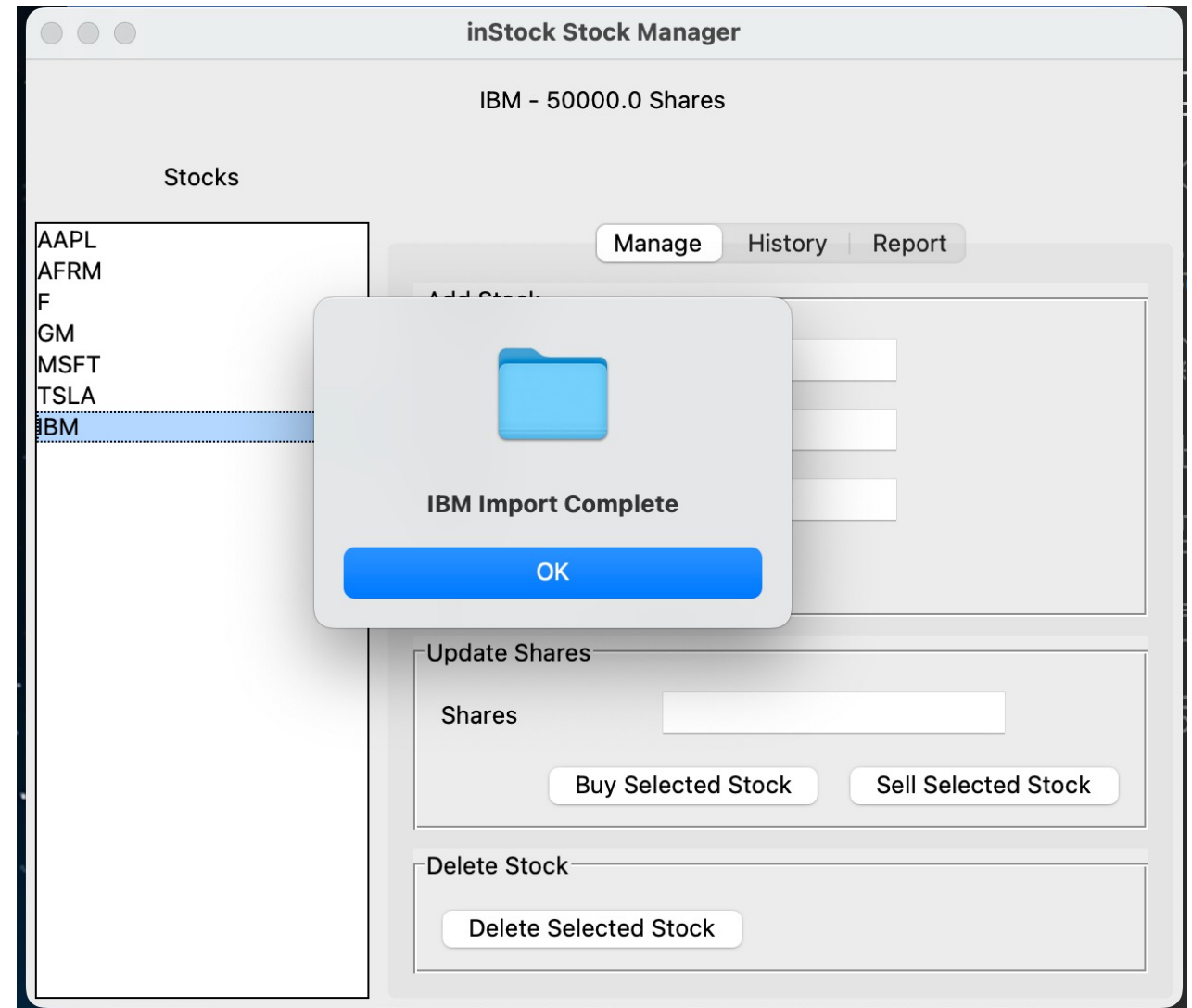
# Report Tab

- Paste a screen shot of your report tab.



# Import Complete

- Paste a screen shot of your Import Complete confirmation message.



# Challenges

---

A few challenges were faced in the development of this project. First and foremost, I am not a fan of the limitations imposed by Anaconda. I prefer to use a clean version of python without the conda package. Therefore, I had to learn how to use Visual Studio Code, venv, and pip. Additionally, keeping track of all the installation packages proved to be challenging, but using virtual environments helped address this.

# Career skills developed

---

I became familiar with the use of pyenv on macOS.

Used virtual environments using venv.

Used pip to manage python packages.

Became familiar with Visual Studio Code.

Used object oriented programming.

Became familiar with gui development using tkinter.

# Conclusion

---

Development of this app, allowed me to see how different pieces of software come together and work together to develop a modern gui based application. Additionally, introduced me to further areas of pursuit, in particular web scraping, database processing, and gui development.