CEIS150 Programming with Objects

Developed by James Garlie

DeVry University

June 2023

Introduction

The project provided experience creating applications in Python. I used object-oriented techniques to develop a stock tracking application. The application has both console and GUI (Graphical User Interfaces). By processing the historical stock data, profit/loss reports can be generated. The system used the Python libraries to create charts and get historical stock data from web sites.

The presentation concludes with Challenges, Career Skills obtained, and a Conclusion.

CEIS150
Module 1

Software Environment Setup

The next slide is a Screen shot showing that the Python program ran successfully.

Program

This is a Screen shot showing that the Python program ran successfully.

C:\Users\16123\Documents\CEIS150\prices.py

Week_6_debugging_demo.py × | Week_6_file_processing.py × | Week_6_randomizer.py × | prices.py ×

```python
# -*- coding: utf-8 -*-
"""
Created on Fri May  5 03:39:22 2023

@author: 16123
"""

sum = 0
count = 0
full_name = input("What is your full name: ")
#You will wright the rest of the code
print("Hello ", full_name)
min_price = float(input("enter the minimum price: "))
price_list = [69.5, 95.6, 79.85, 41, 30.65, 55.5, 47.5, 45.3, 32, 97.85, 32.4]
for price in price_list:

    sum=sum+price
    if price > min_price:
        count = count+1

print("Hello",full_name,"the minimum price is ",min_price)
print("There are ",count,"prices greater than the minimum price")
print("The total price is",sum)
#I changed the prices to match Gina's prices and entered the same minimum price
# to test accuracy. Also to test # messaging.
```

Source | Console | Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

Help | Variable Explorer | Plots | Files

Console 1/A ×

```
The total price is 627.15

In [5]: runfile('C:/Users/16123/Documents/CEIS150/prices.py', wdir='C:/Users/16123/
Documents/CEIS150')

What is your full name: James Garlie
Hello  James Garlie

enter the minimum price: 77
Hello James Garlie the minimum price is  77.0
There are  3 prices greater than the minimum price
The total price is 627.15
```

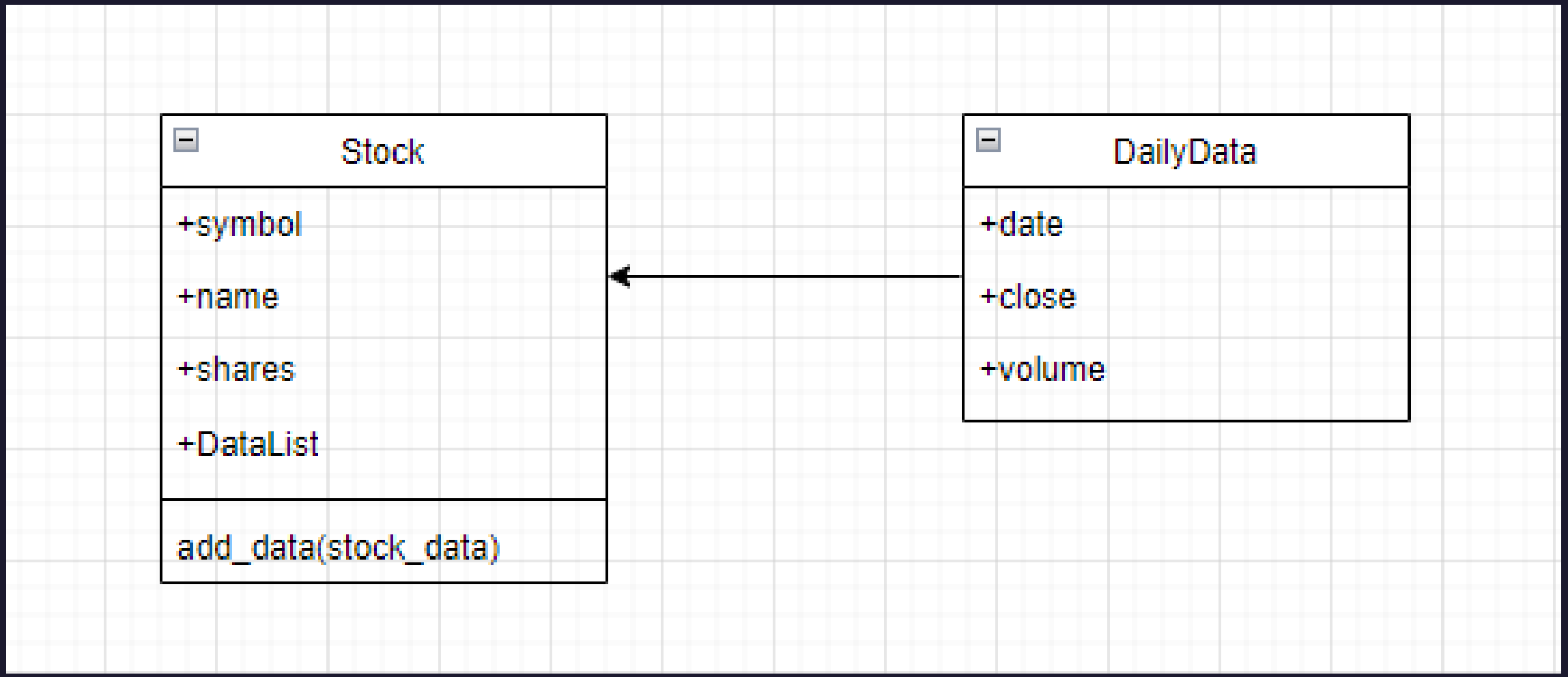
CEIS150
Module 2

Class Diagrams, Coding and Unit Testing

The next three slides show:
1) Class Diagrams,
2) Class Coding; and,
3) Unit Testing.

Class Diagram

This is a Screen shot of the Visio Class Diagram



Class Code

This is a Screen Shot of my stock_class.py file.

ek_6_randomizer.py × | welcome.py × | prices.py × | untitled1.py* × | dogcat.py × | stock_class.py ×

```python
# -*- coding: utf-8 -*-
"""
Created on Wed May 10 07:07:11 2023

@author: James Garlie
"""

class Stock:
    def __init__(self, symbol, name, shares):
        self.symbol = symbol
        self.name = name
        self.shares = shares
        self.DataList = [] # list of daily stock data

    def add_data(self, stock_data):
        self.DataList.append(stock_data)

class DailyData:
    def __init__(self, date, close, volume):
        self.date = date
        self.close = close
        self.volume = volume
```

Unit Test

This is a Screen Shot of a successful unit test.

```
Console 1/A

Meow

In [14]: runfile('C:/Users/16123/Documents/CEIS150/stock_class.py', wdir='C:/Users/16123/
Documents/CEIS150')
Unit Testing Starting---
Testing Add Stock...Successful!
Test Change Symbol...Successful!
Test Change Name...Successful!
Successful!
Creating daily stock data...Successful!
Congratulations - All Tests Passed
Goodbye
```

CEIS150
Module 3

Text-Based User Interface Summary Report

The next three slides show:
1) Adding a Stock,
2) Listing 3 Stocks; and,
3) Daily Data of my working Stock program.

Adding a Stock

This a Screen shot of a working Stock program.

```python
def add_stock(stock_list):
      option = ""
      while option != "0":
          print("Adding a stock")
          symbol = input("Enter symbol: ").upper()
          name = input("Enter company name: ")
          shares = float(input("Enter shares: "))
          new_stock = Stock(symbol, name, shares)
          stock_list.append(new_stock)
          option = input("Press enter to add another stock or 0 to quit: ")

# Remove stock and all daily data
def delete_stock(stock_list):
    print("This method is under construction")


# List stocks being tracked
def list_stocks(stock_list):
    print("Stock List ----")
    print("SYMBOL\t\tNAME\t\tSHARES")
    print("==========================================")
    for stock in stock_list:
        print(stock.symbol," " * (14-len(stock.symbol)),stock.name," " * (14-len(st
    print()
    _=input("Press enter to continue")

    # Add Daily Stock Data
def add_stock_data(stock_list):
   print("This method is under construction")
```

Help Variable Explore

Console 1/A ×

```
6 - Investor Type
7 - Load Data
0 - Exit Program

Enter Menu Option: 1
Adding a stock

Enter symbol: msft

Enter company name: microsoft

Enter shares: 200

Press enter to add another stock or 0 to quit:
Adding a stock
```

Listing 3 Stocks

This is a Screen shot of a working Stock program.

```python
def list_stocks(stock_list):
    print("Stock List ----")
    print("SYMBOL\t\tNAME\t\tSHARES")
    print("=========================================")
    for stock in stock_list:
        print(stock.symbol," " * (14-len(stock.symbol)),stock.name
    print()
    _=input("Press enter to continue")

    # Add Daily Stock Data
def add_stock_data(stock_list):
    print("Add Daily Stock Data ----")
    print("Stock List: [",end="")
    for stock in stock_list:
        print(stock.symbol," ",end="")
    print("]")
    symbol = input("Which stock do you want to use?: ").upper()
    found = False
    for stock in stock_list:
        if stock.symbol == symbol:
            found = True
            current_stock = stock
    if found == True:
        print("Ready to add data for: ",symbol)
        print("Enter Data Separated by Commas - Do Not use Spaces"
        print("Enter a Blank Line to Quit")
        print("Enter Date,Price,Volume")
        print("Example: 8/28/20,47.85,10550")
```

Built-in mutable sequence.

If no argument is given, the constructor creates a new empt

Help Variable E

Console 1/A

```
7 - Load Data
0 - Exit Program

Enter Menu Option: 3
Stock List ----
SYMBOL      NAME        SHARES
=========================================
MSFT            microsoft       100.0
MSFT            Microsoft       100.0
TSLA            Tesla           200.0
WMT             Walmart         300.0


Press enter to continue
```

Daily Data

This is a Screen shot of a working Stock program.

```python
def add_stock_data(stock_list):
    print("Add Daily Stock Data ----")
    print("Stock List: [",end="")
    for stock in stock_list:
        print(stock.symbol," ",end="")
    print("]")
    symbol = input("Which stock do you want to use?: ").upper()
    found = False
    for stock in stock_list:
        if stock.symbol == symbol:
            found = True
            current_stock = stock
    if found == True:
        print("Ready to add data for: ",symbol)
        print("Enter Data Separated by Commas - Do Not use Spaces")
        print("Enter a Blank Line to Quit")
        print("Enter Date,Price,Volume")
        print("Example: 8/28/20,47.85,10550")
        data = input("Enter Date,Price,Volume: ")
        while data != "":
            date, price, volume = data.split(",")
            daily_data = DailyData(date,float(price),float(volume)

            current_stock.add_data(daily_data)
            data = input("Enter Date,Price,Volume: ")
        print("Date Entry Complete")
    else:
        print("Symbol Not Found ***")
```

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The

Help | Variable Explorer | P

Console 1/A

```
Ready to add data for:  MSFT
Enter Data Separated by Commas - Do Not use Spaces
Enter a Blank Line to Quit
Enter Date,Price,Volume
Example: 8/28/20,47.85,10550

Enter Date,Price,Volume: 8/28/21,44,20000

Enter Date,Price,Volume: 8/29/21,46,21000

Enter Date,Price,Volume:
Date Entry Complete

Press Enter to Continue ***
```

CEIS150
Module 4

Inheritance Summary Report

The next three slides show:
1) Inherited classes,
2) Unit Tests; and,
3) A Stock Menu program.

Inherited classes

This is a Screen shot of the classes

```
@author: James Garlie
"""

from stock_class import Stock
#create parent class
class Retirement_Account:
    def __init__(self, balance, number):
        self.balance = balance
        self.number = number

#create Traditional account
class Traditional(Retirement_Account):
    def __init__(self, balance, number):
        Retirement_Account.__init__(self, balance, number)
        self.Stock_List =[]

    def add_stock(self, stock_data):
        self.Stock_List.append(stock_data)

#create derived class Robo account
class Robo(Retirement_Account):
    def __init__(self, balance, number, years):
        Retirement_Account.__init__(self, balance, number)
        self.years = years

    def investment_return(self):
        return (self.years*self.balance*1.05)

# Unit Test - Do Not Change Code Below This Line *** *** *** *** *** *** *** ***
# main() is used for unit testing only. It will run when stock_class.py is run.
```

Unit Tests

This is a Screen shot of a unit tests successfully completed

```python
# -*- coding: utf-8 -*-
"""
Created on Thu May 25 07:28:01 2023

@author: James Garlie
"""

from stock_class import Stock
#create parent class
class Retirement_Account:
    def __init__(self, balance, number):
        self.balance = balance
        self.number = number

#create Traditional account
class Traditional(Retirement_Account):
    def __init__(self, balance, number):
        Retirement_Account.__init__(self, balance, number)
        self.Stock_List =[]

    def add_stock(self, stock_data):
        self.Stock_List.append(stock_data)

#create derived class Robo account
class Robo(Retirement_Account):
    def __init__(self, balance, number, years):
        Retirement_Account.__init__(self, balance, number)
        self.years = years

    def investment_return(self):
        return (self.years*self.balance*1.05)

# Unit Test - Do Not Change Code Below This Line *** *** *** *** *** *** *** *** ***
# main() is used for unit testing only. It will run when stock_class.py is run.
# Run this to test your class code. Once you have eliminated all errors, you are
```

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument mus
specified.

Help | Variable Explorer | Plots | Files

Console 1/A

```
Users/16123/Documents/CEIS150')
Reloaded modules: stock_class
Unit Testing Starting---
Testing Add Retirement Account...Successful!
Testing Add Traditional Account...Successful!
Test Change Balance...Successful!
Test Change Number...Successful!
Testing Add Robo Account...Successful!
Test Change Balance...Successful!
Test investment return...Successful!
Congratulations - All Tests Passed
Goodbye

In [34]:
```

Stock Menu Program

This is a screen shot of the classes in the main program showing the Traditional account.

```
Do you want a Traditional (t) or Robo (r) account: t
Choose stocks from the list below:
Stock List: [MSFT  WMT  ]

Which stock do you want to purchase, 0 to quit: msft

How many shares do you want to buy?: 800
Bought  800.0 of MSFT
Stock List: [MSFT  WMT  ]

Which stock do you want to purchase, 0 to quit: 0
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: 3
Stock List ----
SYMBOL      NAME        SHARES
=========================================
MSFT            microsoft       900.0
WMT             walmart         200.0
```

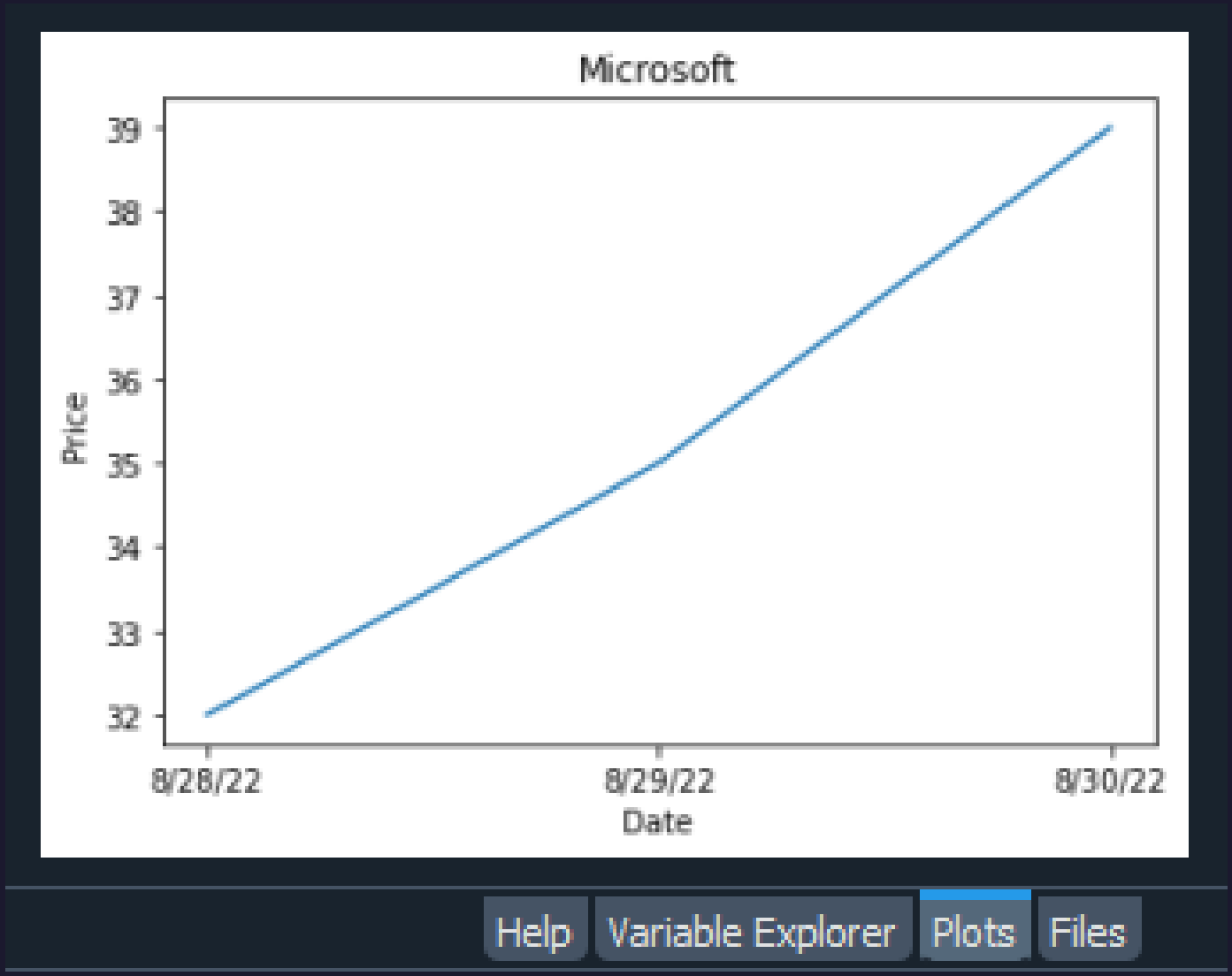
CEIS150
Module 5

Creating a Chart

The next slide shows a Screen shot of a stock chart.

Chart

This is a Screen shot of the stock chart.



CEIS150
Module 6

Loading Data

The next two slides show:
1) Loading Data; and,
2) Importing Data

File

This is a Screenshot of the TSLA.csv file downloaded from Yahoo finance

This PC › Documents › CEIS150

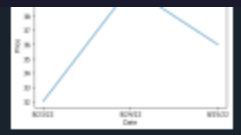
Search CEIS150

Name	Date modified	Type	Size
__pycache__	5/25/2023 8:44 AM	File folder	
account_class.py	5/25/2023 8:14 AM	PY File	5 KB
CEIS150 Project Template Module Deliver...	5/5/2023 5:00 AM	Microsoft PowerP...	119 KB
CEIS150_W2_Project Template Module De...	5/10/2023 7:36 AM	Microsoft PowerP...	103 KB
CEIS150_W3_Project Template Module De...	5/20/2023 3:43 PM	Microsoft PowerP...	235 KB
CEIS150_W4_Project Template Module De...	5/25/2023 8:56 AM	Microsoft PowerP...	168 KB
CEIS150_W5_Project Template Module De...	5/31/2023 1:25 PM	Microsoft PowerP...	137 KB
CEIS150_W6_Project_Template_Module_D...	6/5/2023 6:09 PM	Microsoft PowerP...	45 KB
dogcat.py	5/9/2023 9:22 PM	PY File	1 KB
prices.py	5/5/2023 4:53 AM	PY File	1 KB
stock_class.py	5/10/2023 7:25 AM	PY File	5 KB
stock_menu.py	6/6/2023 7:41 AM	PY File	10 KB
TSLA.csv	6/5/2023 3:09 PM	Microsoft Excel C...	19 KB
welcome.py	5/6/2023 4:21 AM	PY File	1 KB

Importing data

This is a Screenshot showing the code and the historical data import from the TSLA.csv file.

```python
    for stock in stock_list:
        if stock.symbol == symbol:
            with open(filename, newline="") as stockdata:
                datareader = csv.reader(stockdata, delimiter=",")
                next(datareader)
                for row in datareader:
                    daily_data = DailyData(str(row[0]), float(row[4]), float(row[6]))
                    stock.add_data(daily_data)
    display_report(stock_list)

    # Display Report
def display_report(stock_list):
    print("Stock Report ====")
    for stock in stock_list:
        print("Repoert for: ", stock.symbol, stock.name)
        print("Shares: ", stock.shares)
        #variable initialization
        count = 0
        price_total = 0
        volume_total = 0
        lowPrice = 9999999.99
        highPrice = 0.0
        lowVolume = 99999999999
        highVolume = 0

        for daily_data in stock.DataList:
            count = count + 1
            price_total = price_total + daily_data.close
            volume_total = volume_total + daily_data.volume
            if daily_data.close < lowPrice:
                lowPrice = daily_data.close
            if daily_data.close > highPrice:
                highPrice = daily_data.close
            if daily_data.volume < lowVolume:
                lowVolume = daily_data.volume
```



Help | Variable Explorer | Plo

Console 1/A

```
Enter stock symbol: tsla

Enter the file name: TSLA.csv
Stock Report ====
Repoert for:  TSLA Tesla
Shares:  200.0
2022-06-06 238.279999 84204600.0
2022-06-07 238.886673 72808500.0
2022-06-08 241.866669 76210500.0
2022-06-09 239.706665 96491400.0
2022-06-10 232.229996 97536600.0
2022-06-13 215.736664 102767400.0
2022-06-14 220.889999 97988700.0
2022-06-15 233.0 119131800.0
2022-06-16 213.100006 107390700.0
2022-06-17 216.759995 92641800.0
2022-06-21 237.036667 122793000.0
2022-06-22 236.08667 101107500.0
2022-06-23 235.070007 104202600.0
2022-06-24 245.706665 95770800.0
2022-06-27 244.919998 89178300.0
2022-06-28 232.66333 90391200.0
2022-06-29 228.490005 82897200.0
2022-06-30 224.473328 94600500.0
2022-07-01 227.263336 74460300.0
2022-07-05 233.066666 84581100.0
```

IPython console | Histo

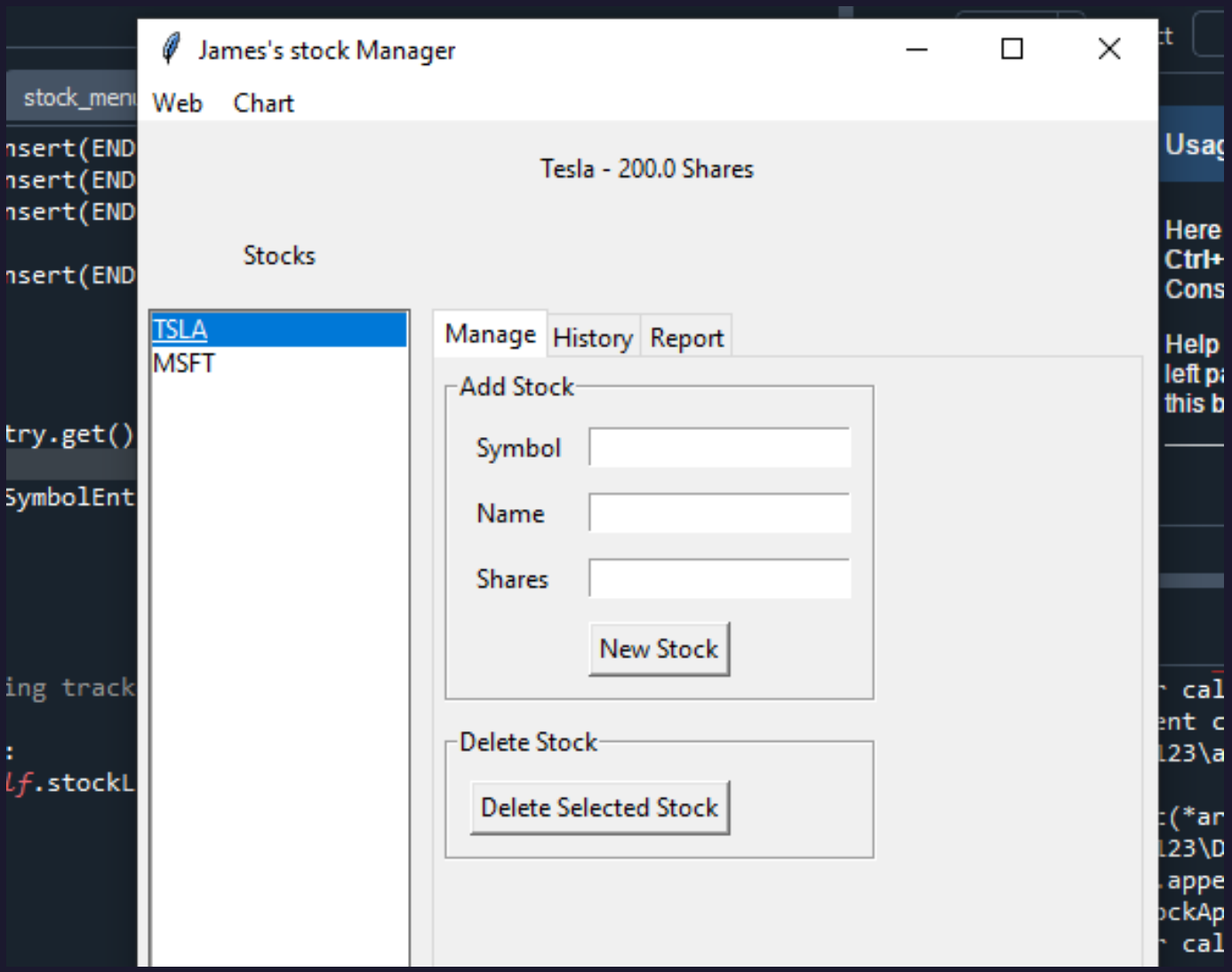
CEIS150
Module 7

GUI (Graphical User Interfaces)

The next four slides show:
1) Stocks in GUI,
2) History Tab
3) Chart with Graph; and,
4) Report Complete.

Stocks in GUI

This is a Screen shot of the GUI working.



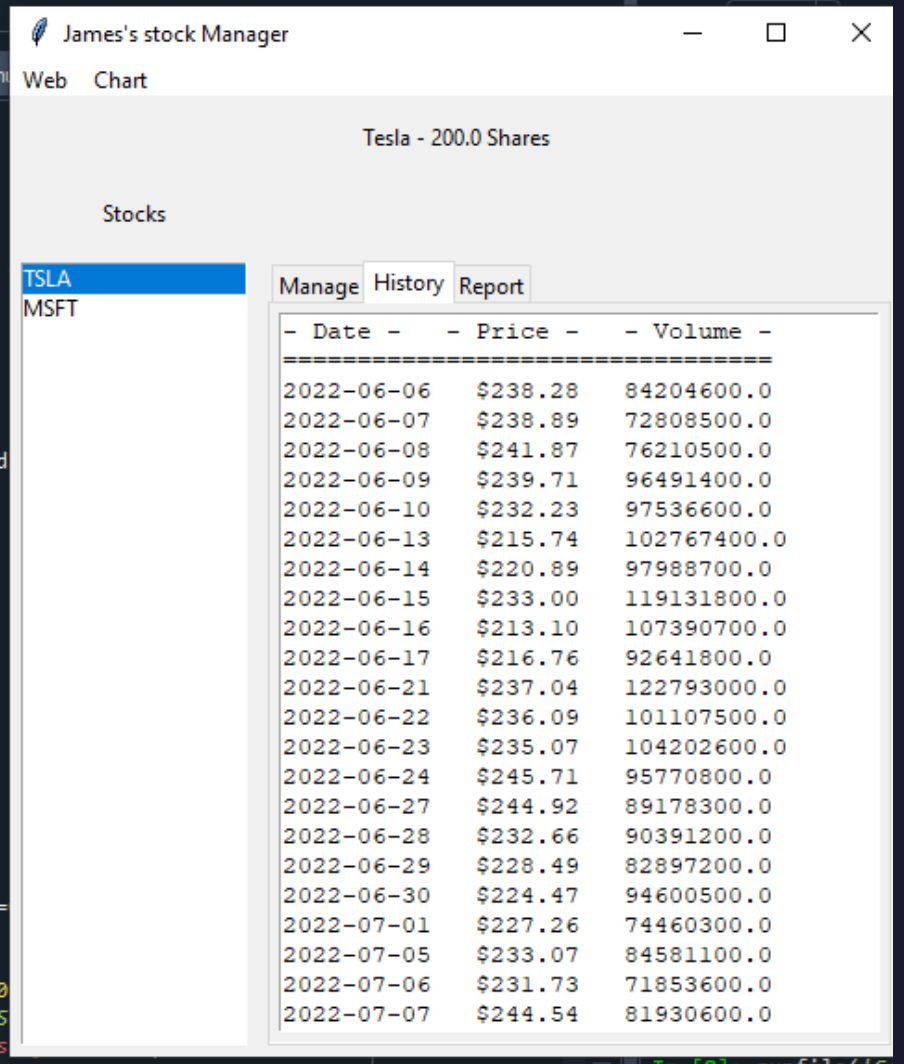
History Tab

This is a Screen shot of the History tab with import working.

5123\Documents\CEIS150\stock_GUI.py

py × prices.py × dogcat.py × stock_class.py × stock_menu

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jun 17 19:48:43 2023
@author: James Garlie
"""
# Summary: This module contains the user interface

from datetime import datetime
from stock_class import Stock, DailyData
from os import path
from tkinter import *
from tkinter import ttk
from tkinter import messagebox, simpledialog, filed
import csv
import matplotlib.pyplot as plt
import json

class StockApp:
    def __init__(self):
        self.stock_list = []

        # Create Window
        self.root = Tk()
        self.root.title("James's stock Manager")

        # Add Menu
        self.menubar = Menu(self.root)

        self.filemenu = Menu(self.menubar, tearoff=


        self.webmenu = Menu(self.menubar, tearoff=0
        self.webmenu.add_command(label = "Import CS
        self.menubar.add_cascade(label="Web",menu=s
```

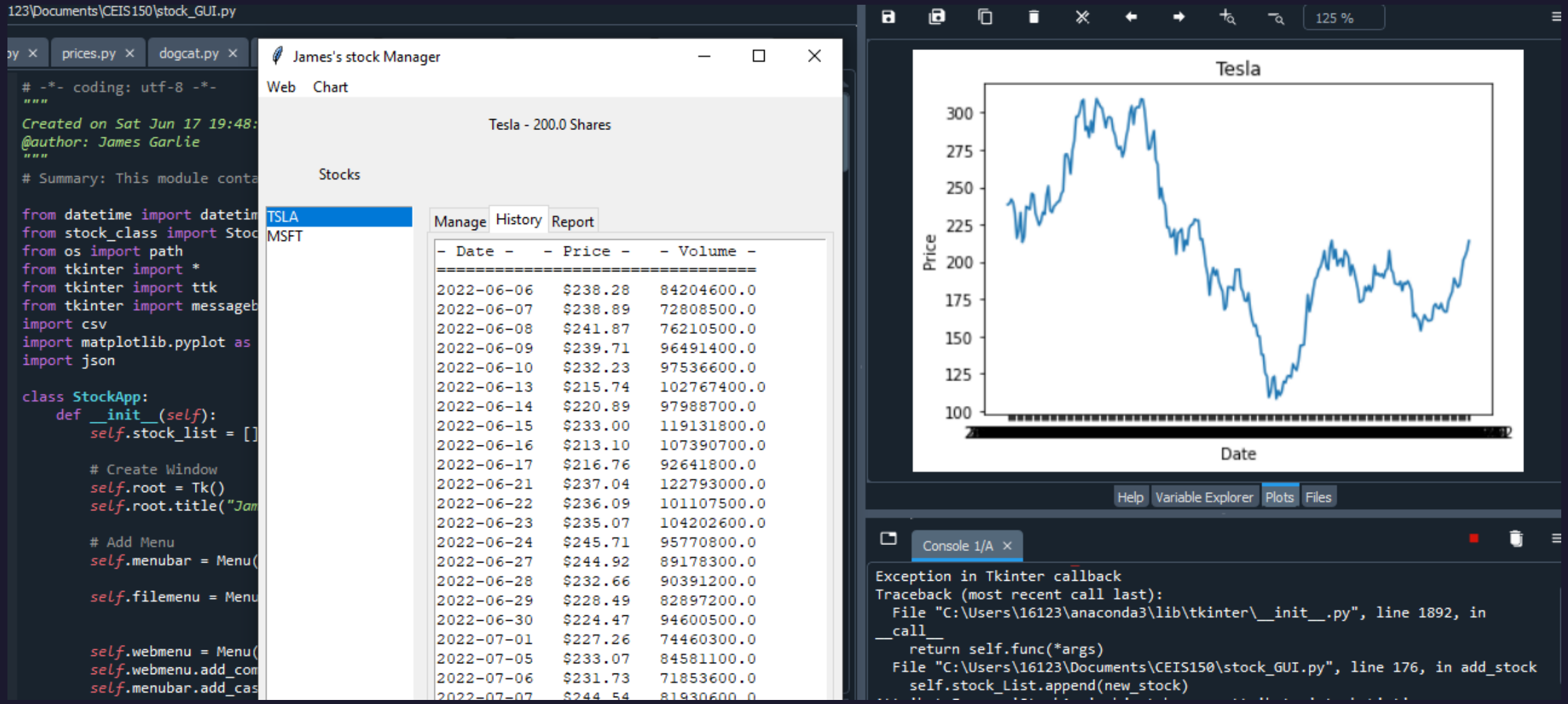


Date	Price	Volume
2022-06-06	$238.28	84204600.0
2022-06-07	$238.89	72808500.0
2022-06-08	$241.87	76210500.0
2022-06-09	$239.71	96491400.0
2022-06-10	$232.23	97536600.0
2022-06-13	$215.74	102767400.0
2022-06-14	$220.89	97988700.0
2022-06-15	$233.00	119131800.0
2022-06-16	$213.10	107390700.0
2022-06-17	$216.76	92641800.0
2022-06-21	$237.04	122793000.0
2022-06-22	$236.09	101107500.0
2022-06-23	$235.07	104202600.0
2022-06-24	$245.71	95770800.0
2022-06-27	$244.92	89178300.0
2022-06-28	$232.66	90391200.0
2022-06-29	$228.49	82897200.0
2022-06-30	$224.47	94600500.0
2022-07-01	$227.26	74460300.0
2022-07-05	$233.07	84581100.0
2022-07-06	$231.73	71853600.0
2022-07-07	$244.54	81930600.0

In [8]: runfile('C:

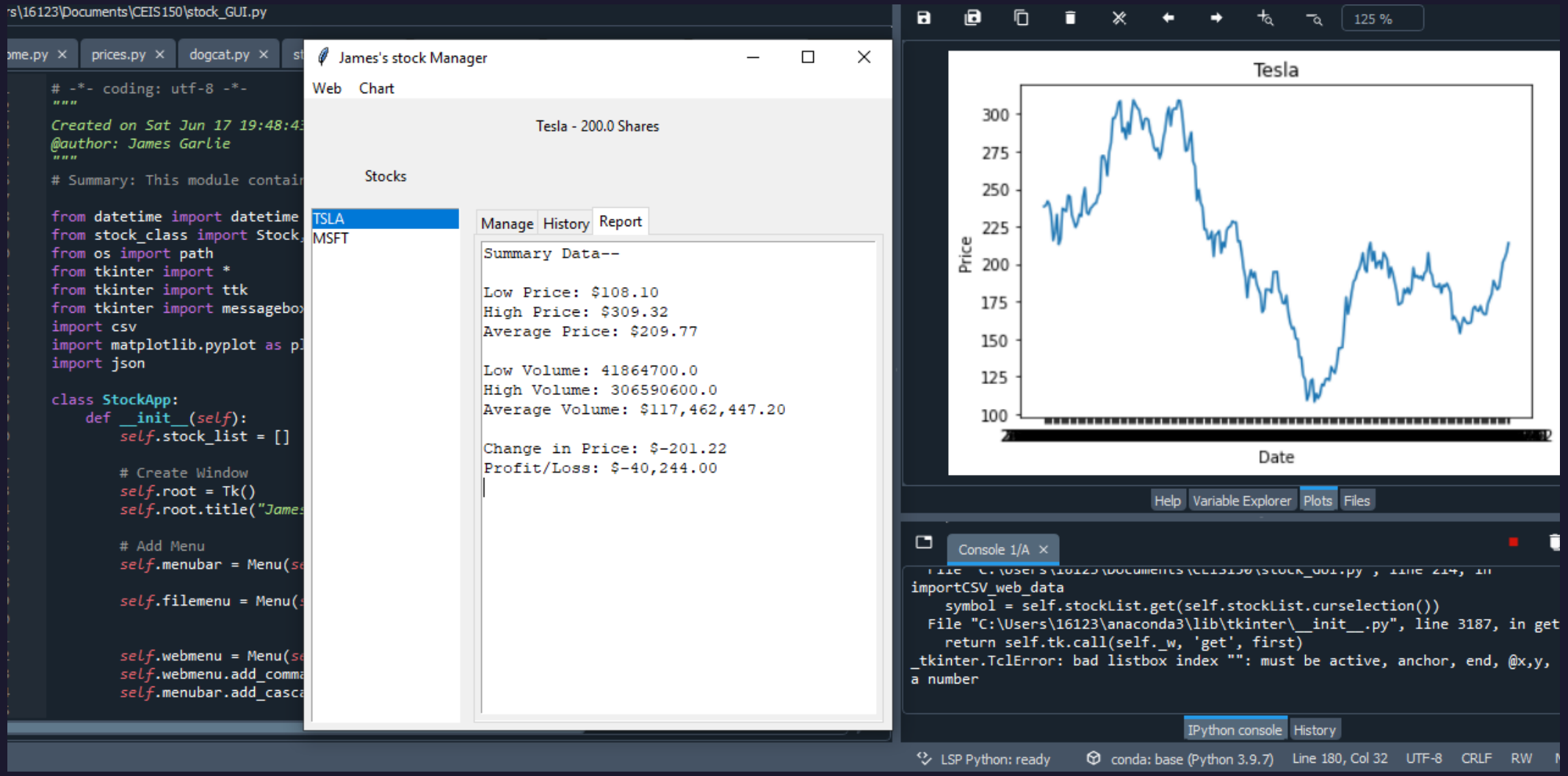
Chart with Graph

This Screen shot shows the Chart and Graph.



Report Complete

This Screen shot shows the Report complete.



Challenges

Identifying the proper login procedures.

Learning how to work with new programs and devices.

Testing the additions at each stage.

Learning how to upload and import new data.

Career Skills

Writing code using Python and Anaconda with Spyder.

Using, configuring, and reading a Spectrum Analyzer.

Understanding data and signals.

Further developed basic and advanced computer skills.

Conclusion

Programming Objects and writing code with Python & Spyder is truly an exciting field.

I found learning about Programming Objects and learning more about writing code using Python with Spyder to be very rewarding.

This project will be of tremendous benefit in the future.