CEIS150

Programming With Objects

Python Stock Tracking Project

DeVry University

Louie Santino Venegas

GitHub

Python Stock Tracking Project

BackgroundThis project provided experience creating applications in Python. I used object-oriented programming in the development of this stock tracking application. I gained an understanding of abstraction, polymorphism and inheritance. My application will generate profit and loss reports by processing historical stock data, retrieved from yahoo finance. Stock data can be retrieved and saved using data storage and working with files. The application will use the Python libraries to create charts. This application has two versions either a console-based version or a graphical user interface(GUI) created with tkinter library. The following slides document the development process.

The project code is accessible via my GitHub account. https://github.com/lvenegas4/CEIS150-Course-Project.git

CEIS150

Programming With Objects

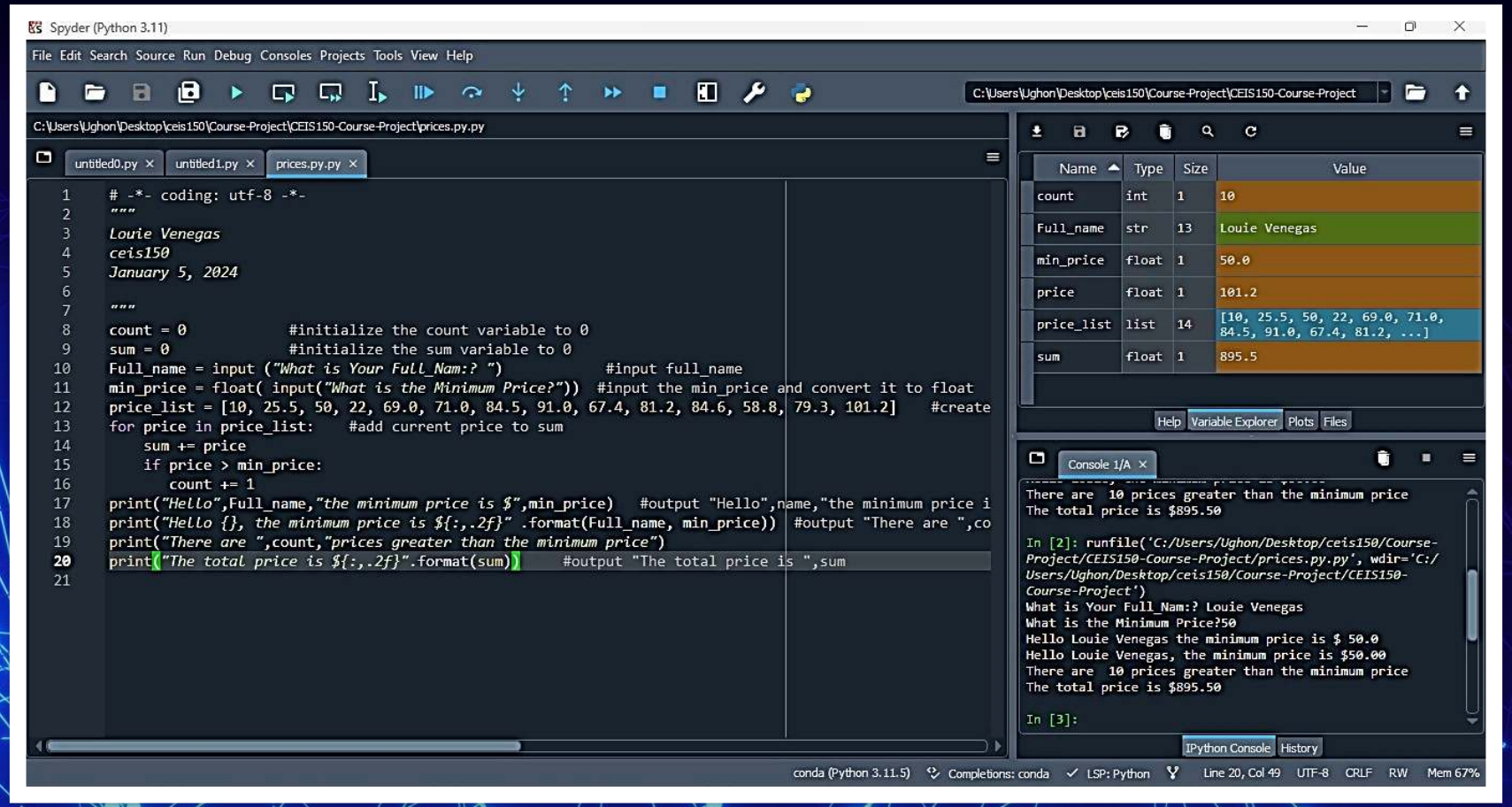
Setting up the Environment

DeVry University

Louie Santino Venegas

Getting Started

The following slide shows a program called prices.py it will find prices greater than a minimum price. Get user to input their name and a minimum price. Essentially this was a test program to make sure the IDE was functioning properly since it was newly installed. It could be used as a filter in a stock screener program.



Prices.py (test program)

CEIS150

Programming With Objects

Creating the Classes

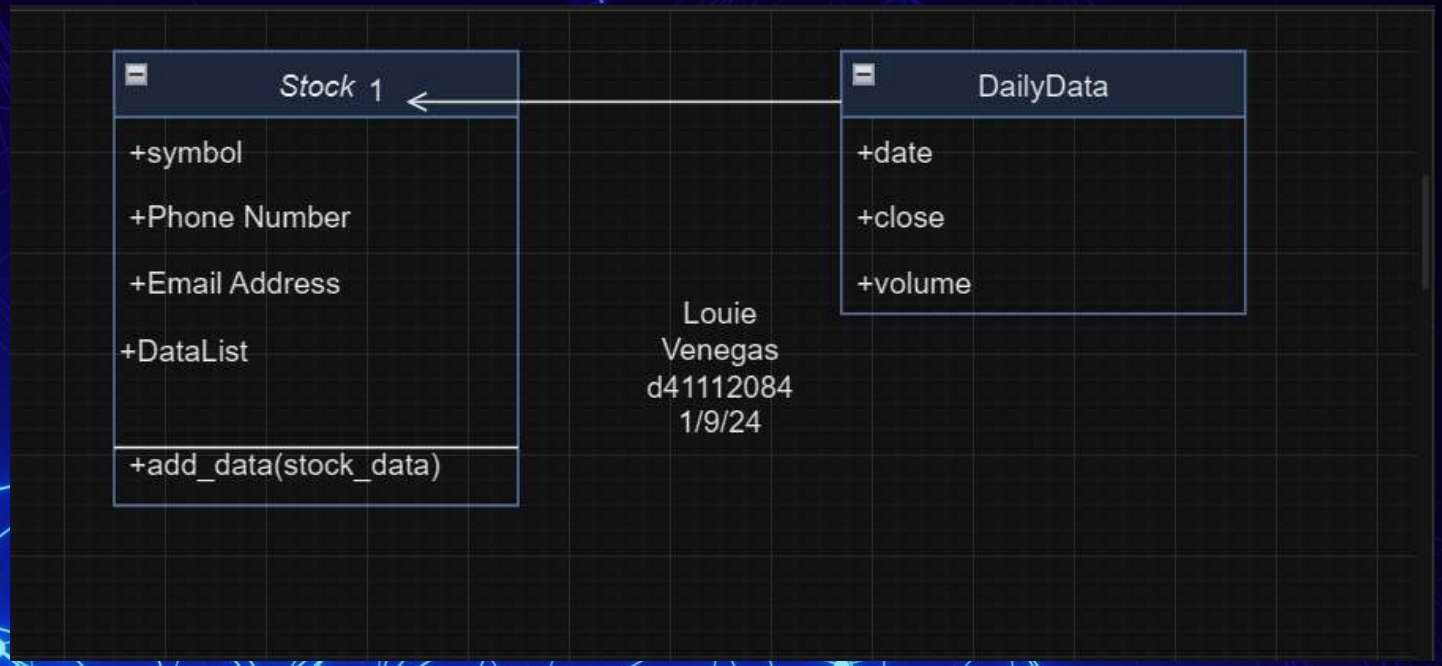
DeVry University

Louie Santino Venegas

Creating the Classes

Here in this section, I laid the foundation of my stock application by creating the classes. For those who don't speak programmer a class is kind of like a model used to create an object that can be called upon at any part of the program. The first slide in this section is a photo of the class diagram to help visualize the flow of the program. The second is a screenshot of the two classes I created. The Third is a photo of a unit test with a successful run. Finally, I am using the git GUI to push my project to my GitHub account.

Class Diagram



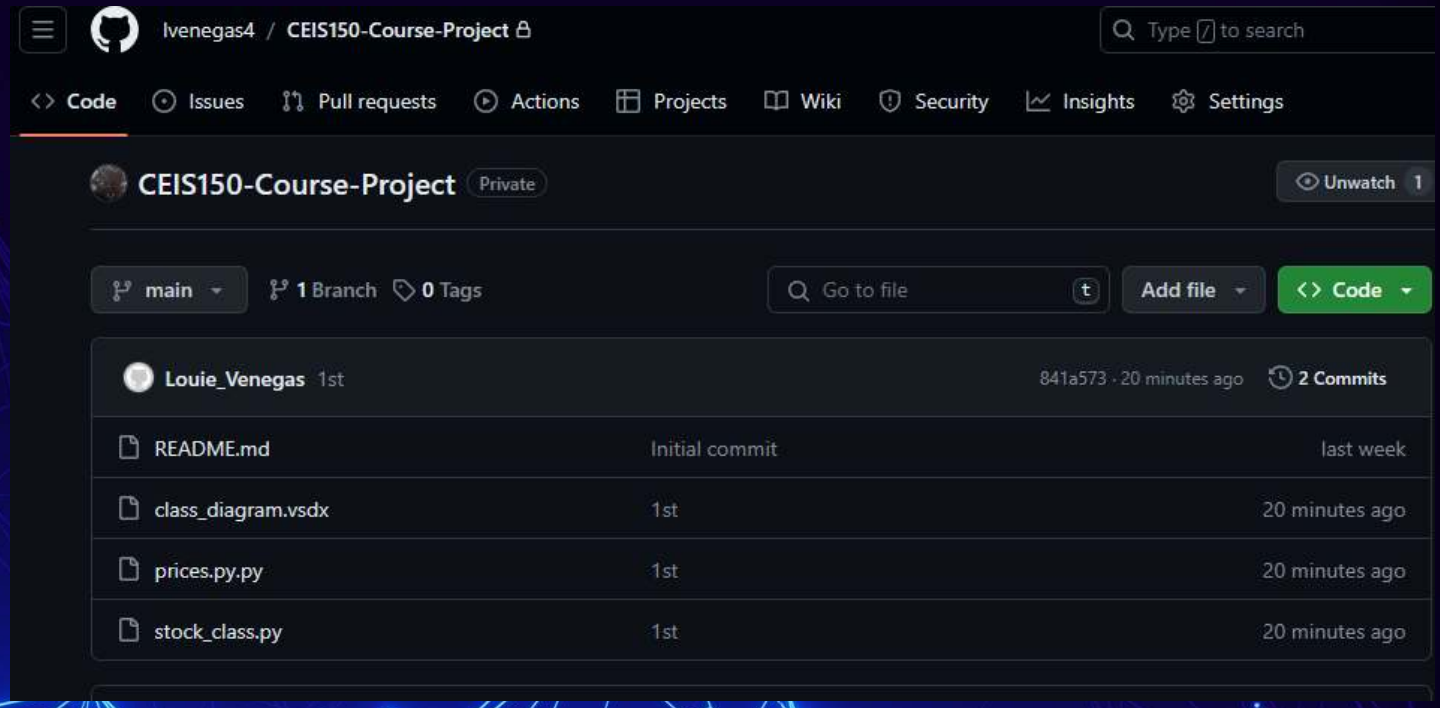
```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jan  8 18:39:14 2024

Louie Santino Venegas
"""
import warnings

class Stock:
    def __init__(self, symbol, name, shares=0.0):
        if not symbol or not name:
           warnings.warn("You must include a symbol and a name.", UserWarning)
        if shares < 0:
            warnings.warn("Shares cannot be negative.", UserWarning)
        self.symbol = symbol
        self.name = name
        self.shares = shares
        self.DataList = [] #DailyData list

    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
```

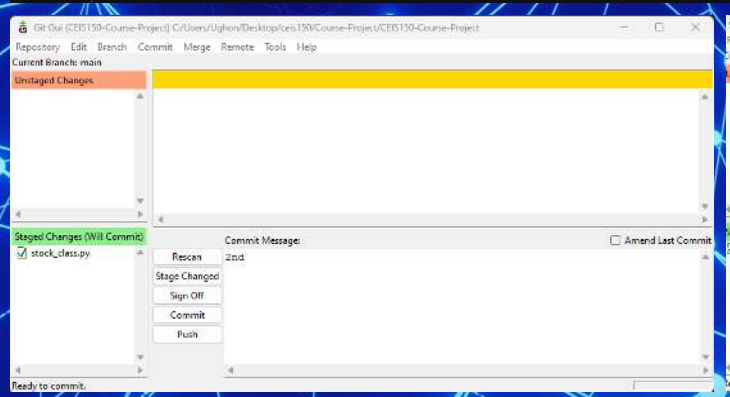
Class Code

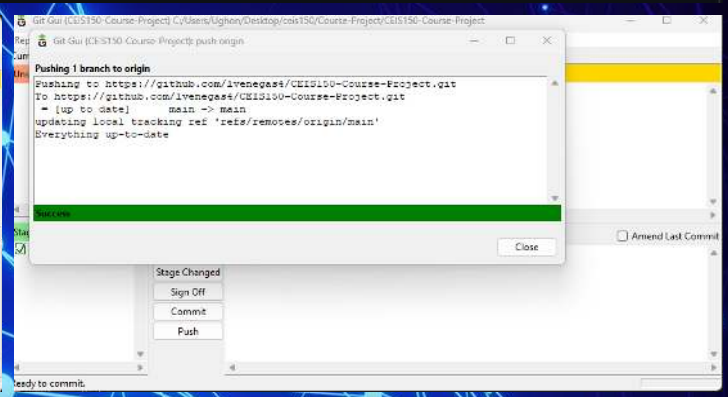
Unit Test

```
In [1]: runfile('C:/Users/Ughon/Desktop/ceis150/Course-
Project/CEIS150-Course-Project/class.py', wdir='C:/
Users/Ughon/Desktop/ceis150/Course-Project/CEIS150-
Course-Project')
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

In [2]:
```

Git to GitHub







CEIS150

Programming With Objects

Creating a Console-based Interface

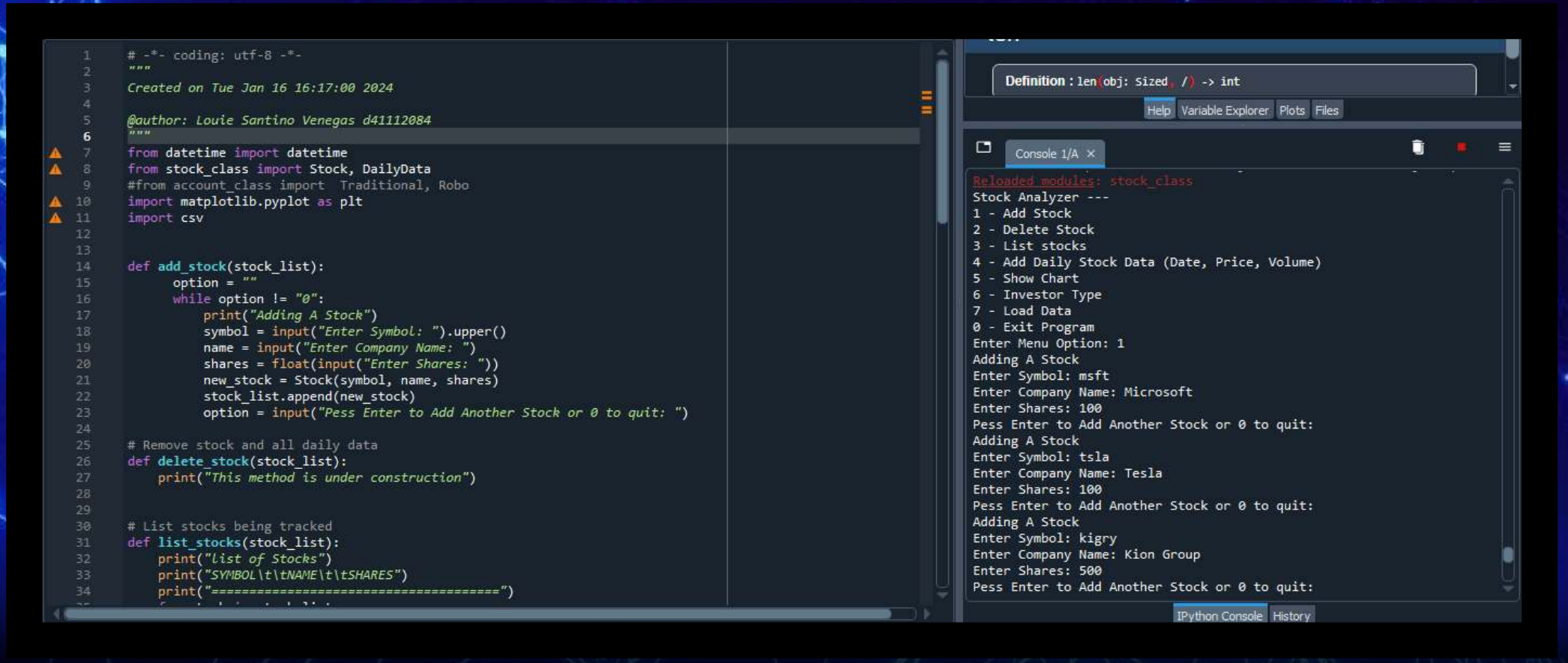
DeVry University

Louie Santino Venegas

Creating a Console-based Interface

The following slides demonstrate the creation of the console-based interface version for my stock application. This will be a Menu-Driven Interface the functionality will be add, list, delete and track both price and volume of stock. I must create functions to handle menu items. The following slides document my solutions working correctly. Followed by myself using a bash terminal and git to push to GitHub.

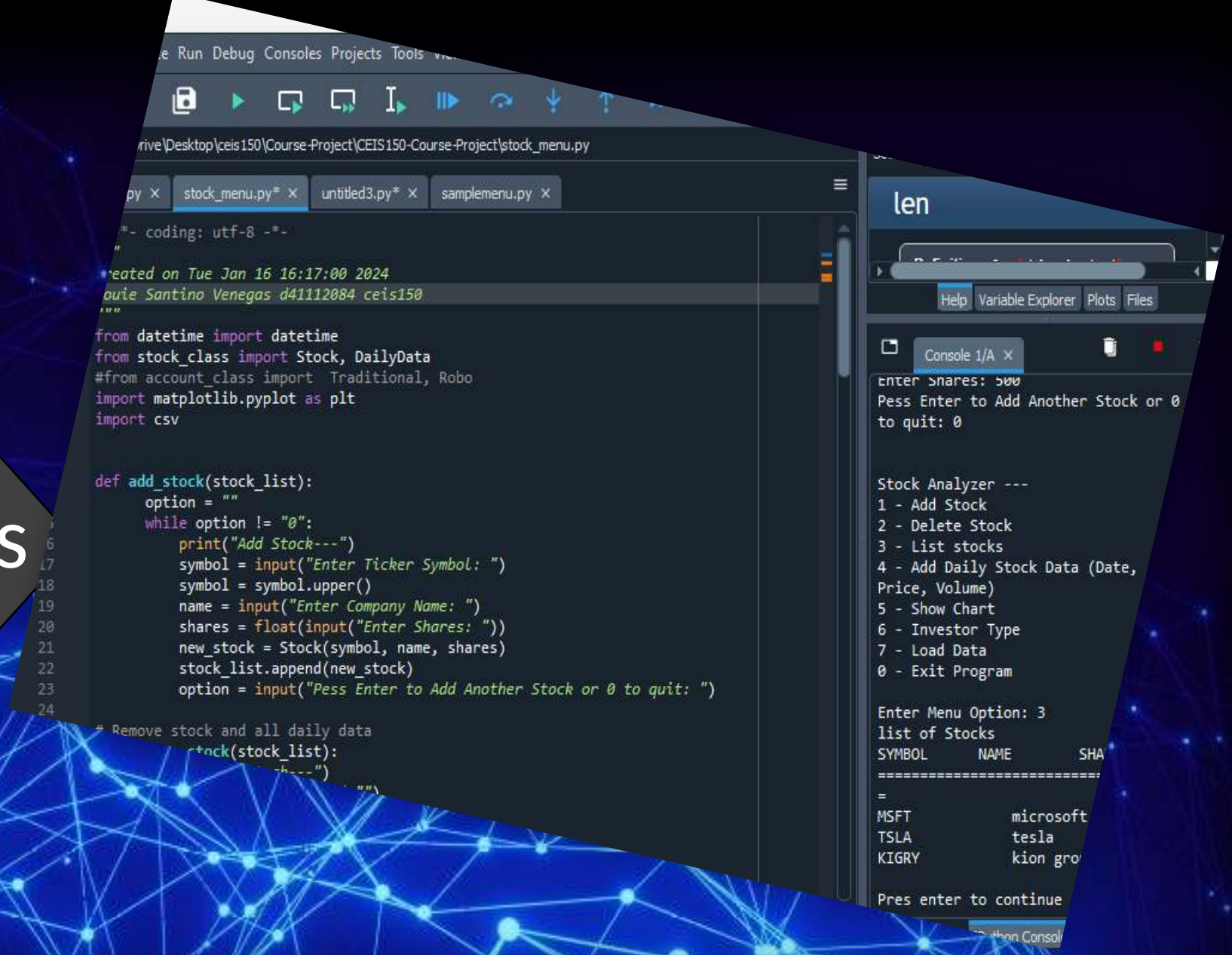
Adding a Stock

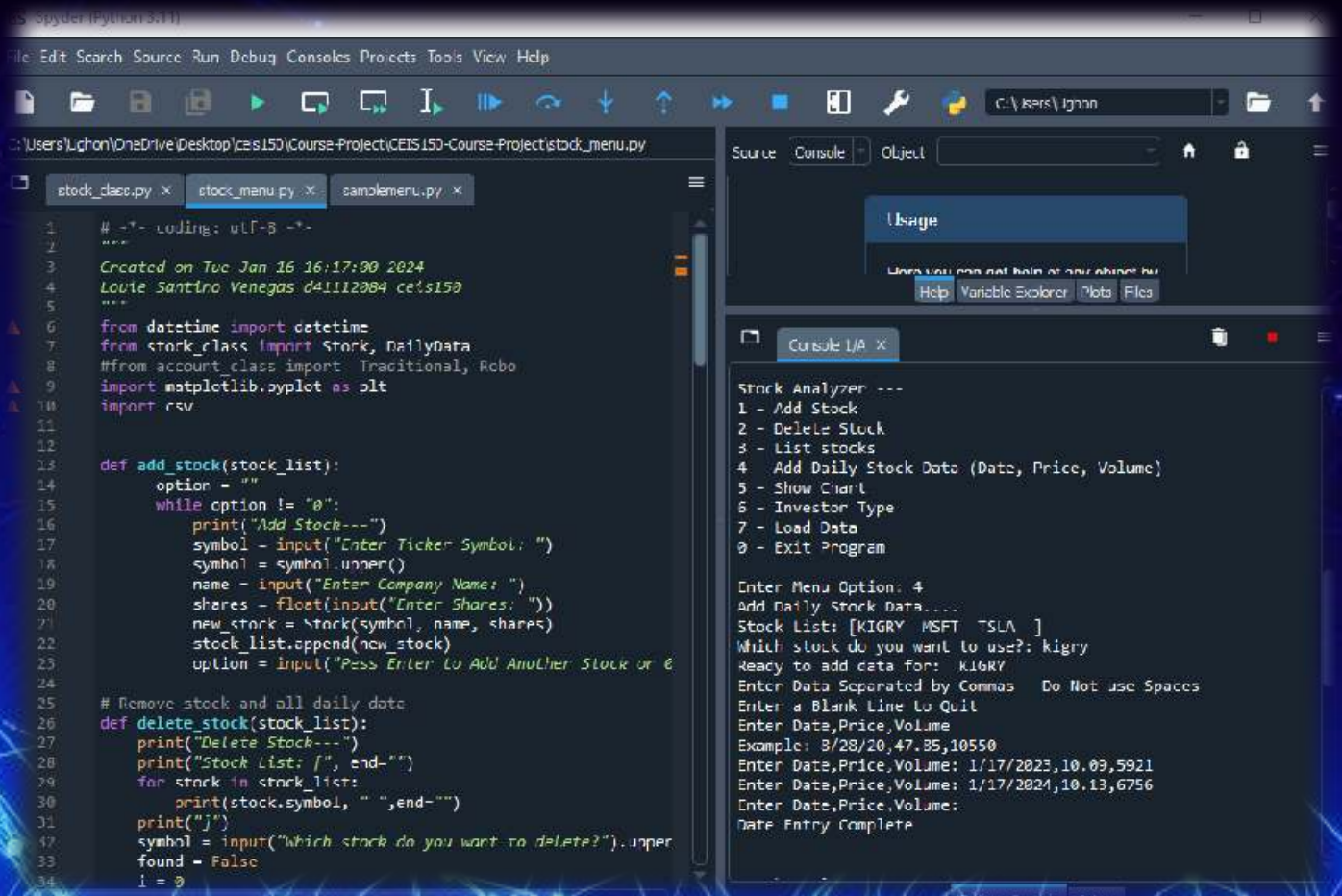


```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 16 16:17:00 2024

@author: Louie Santino Venegas d41112084
"""
from datetime import datetime
from stock_class import Stock, DailyData
#from account_class import  Traditional, Robo
import matplotlib.pyplot as plt
import csv


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("Pess 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("List of Stocks")
    print("SYMBOL\t\tNAME\t\tSHARES")
    print("====================================")
```

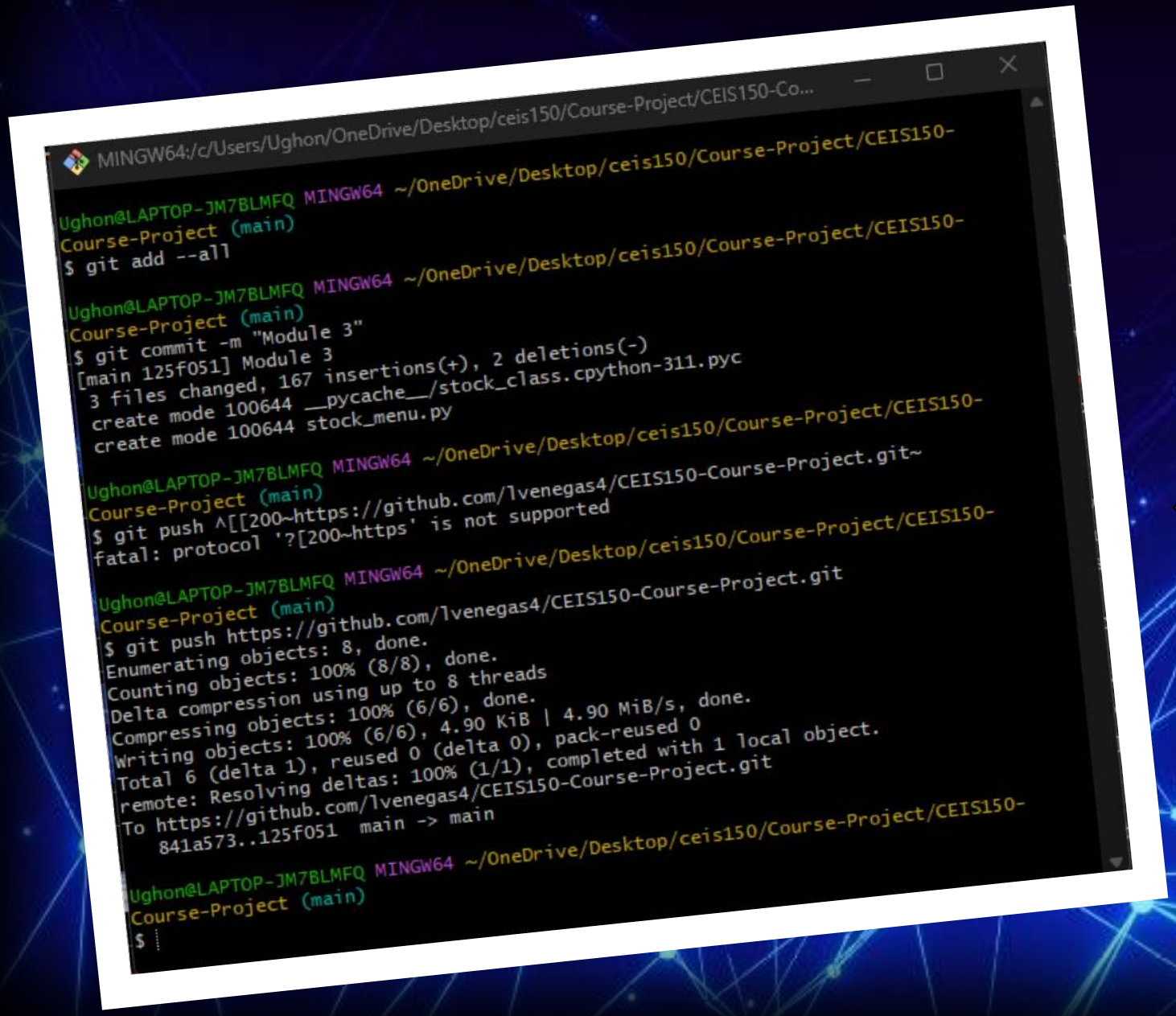
```
Definition : len(obj: Sized, /) -> int

Help | Variable Explorer | Plots | Files

Console 1/A

Reloaded modules: stock_class
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: 1
Adding A Stock
Enter Symbol: msft
Enter Company Name: Microsoft
Enter Shares: 100
Pess Enter to Add Another Stock or 0 to quit:
Adding A Stock
Enter Symbol: tsla
Enter Company Name: Tesla
Enter Shares: 100
Pess Enter to Add Another Stock or 0 to quit:
Adding A Stock
Enter Symbol: kigry
Enter Company Name: Kion Group
Enter Shares: 500
Pess Enter to Add Another Stock or 0 to quit:

IPython Console | History
```

Adding 3 Stocks





Adding Stock Data



Git to GitHub

CEIS150

Programming With Objects

Inheritance Summary Report

DeVry University

Louie Santino Venegas

Inheritance

In this section I documented creating classes inherited from a base class. This is a key part of object-oriented programming. This function of the stock application never makes it to the GUI version but uses the principles of inheritance. In the classes created earlier specifically stock_class imported at the top of the photo in the first succeeding slide. I created another class called retirement_account. The following classes traditional and robo inherit arguments from both stock_class and retirement_account. The following two slides demonstrate that the solutions were functioning properly. Followed by the git push to GitHub.

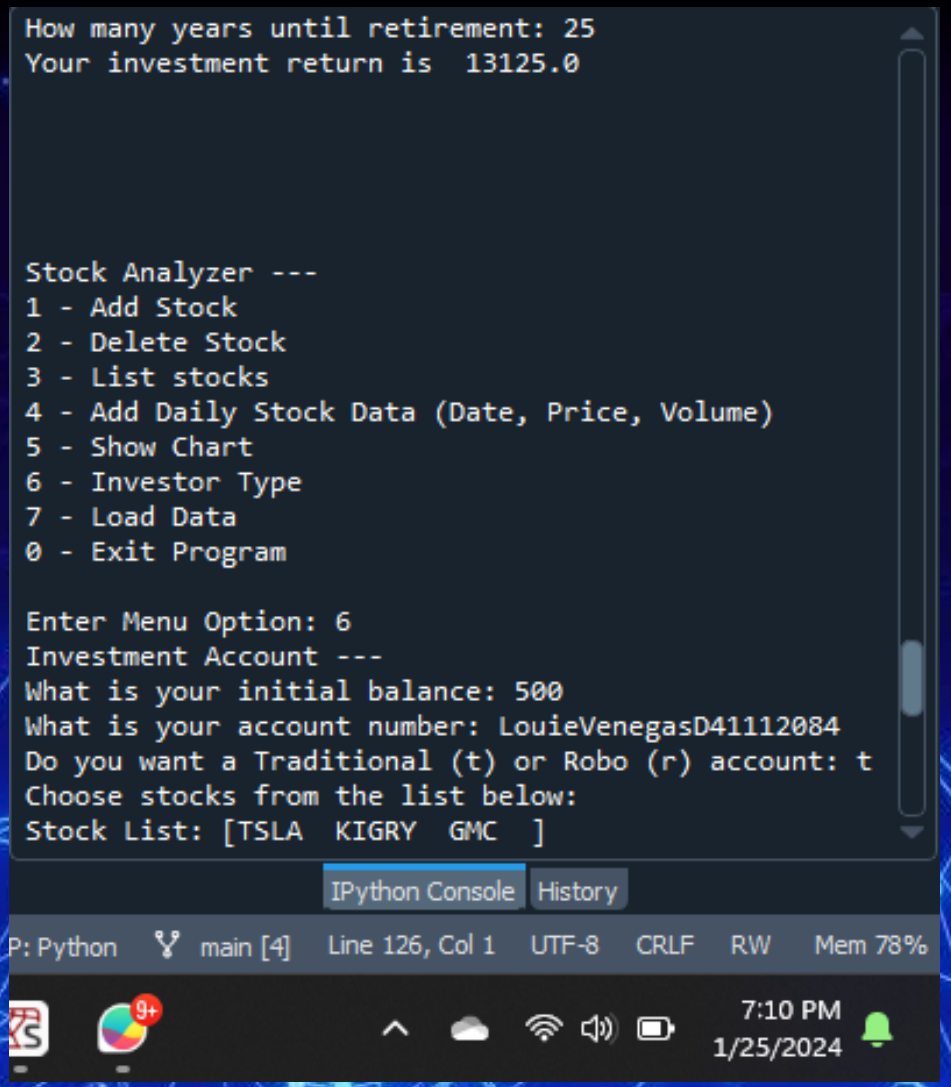
GitHub

```python
# -*- coding: utf-8 -*-
"""
1/24/24
louievenegas d41112084 ceis150
"""

from stock_class import Stock

# parent -- base class
class Retirement_Account:
    def __init__(self, balance, number):
        self.balance = balance
        self.number = number

#create child class that has a stock list
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 child robo class
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)
```

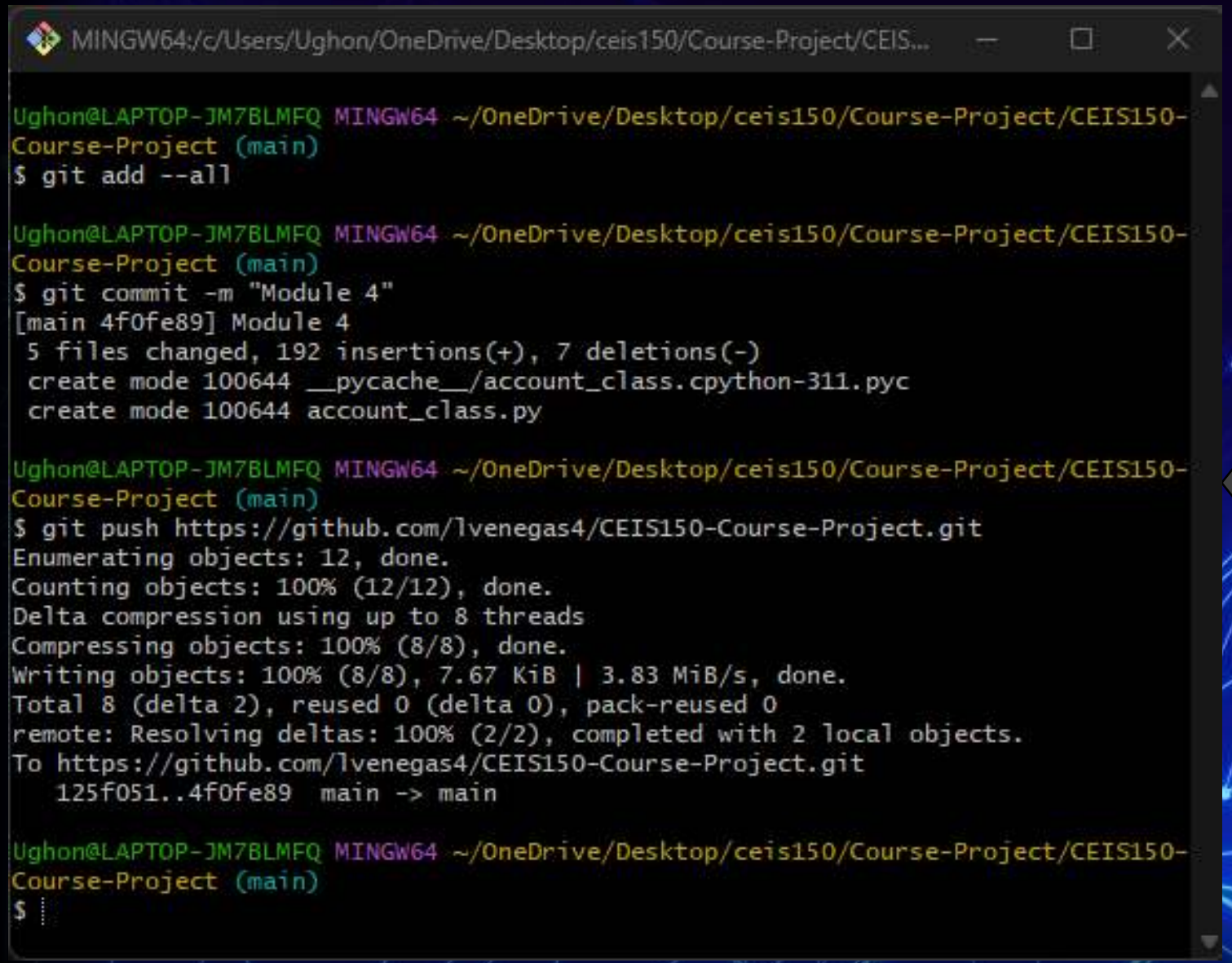
Inherited Classes

Unit Test

```
In [5]: runfile('C:/Users/Ughon/
OneDrive/Desktop/ceis150/Course-
Project/CEIS150-Course-Project/
account_class.py', wdir='C:/Users/
Ughon/OneDrive/Desktop/ceis150/Course-
Project/CEIS150-Course-Project')
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
Louie Venegas d41112084 1/24/24
```

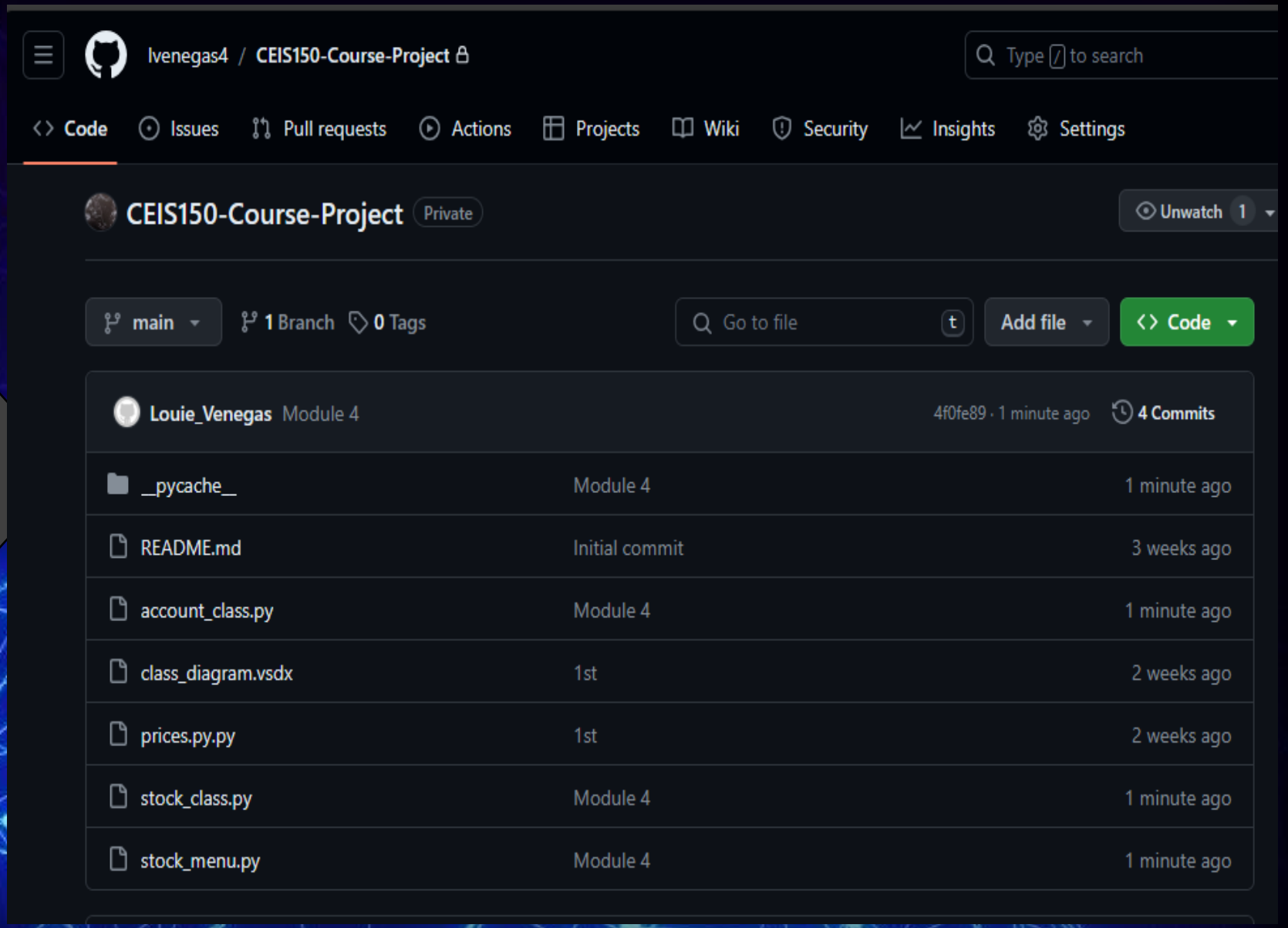


Stock Menu Program



Git to GitHub

GitHub Repository



CEIS150

Programming With Objects

Creating a Stock Chart With matplotlib.pyplot

DeVry University

Louie Santino Venegas

Creating a Stock Chart With matplotlib.pyplot

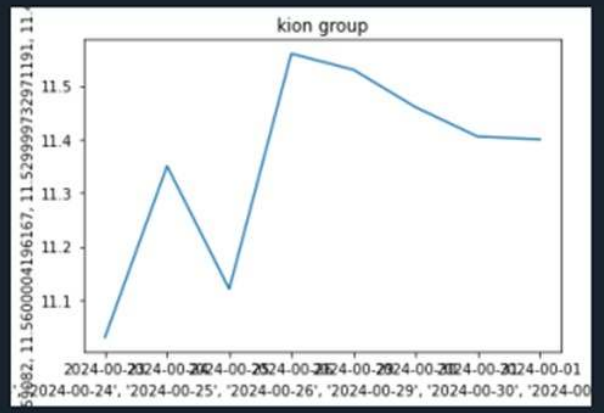
No stock application would be complete unless it could produce a stock chart… Which is menu selection number 5. If you have data inputted into the system about your stock this application will produce a basic price chart using the price change over time to create the price line. In order to produce this awesome feature I incorporated or imported the matplotlib.pyplot library. This feature makes it to both versions of the application. You can examine the code used and the chart produced by the code in the following slide.

Stock Chart generated with matplotlib.pyplot

C:\Users\Ughon\OneDrive\Desktop\ceis150\Course-Project\CEIS150-Course-Project\stock_menu.py

untitled0.py × | account_class.py × | stock_class.py × | stock_menu.py* ×

```
# -*- coding: utf-8 -*-
"""
Created on Tue feb 1, 16:17:00 2024
Louie Santino Venegas d41112084 ceis150
"""
from datetime import datetime
from stock_class import Stock, DailyData
from account_class import  Traditional, Robo
import matplotlib.pyplot as plt
import csv


def add_stock(stock_list):
      option = ""
      while option != "0":
          print("Add Stock---")
          symbol = input("Enter Ticker Symbol: ")
          symbol = 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("Pess Enter to Add Another Stock or 0 to quit: ")

# Remove stock and all daily data
def delete_stock(stock_list):
    print("Delete Stock---")
    print("Stock List: [", end="")
    for stock in stock_list:
        print(stock.symbol, " ",end="")
    print("]")
    symbol = input("Which stock do you want to delete?").upper()
    found = False
    i = 0
```



Help | Variable Explorer | Plots | Files

Console 1/A ×

```
0 - Exit Program

Enter Menu Option: 5
Stock List: [KIGRY ]
Enter stock symbol: kigry
```

Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

```
Press Enter to continue...
```

IPython Console | History

conda (Python 3.11.5) | Completions: conda | LSP: Python | main [3] | Line 3, Col 22 | UTF-8 | CRLF | RW | Mem 86%

CEIS150

Programming With Objects

Working with Files

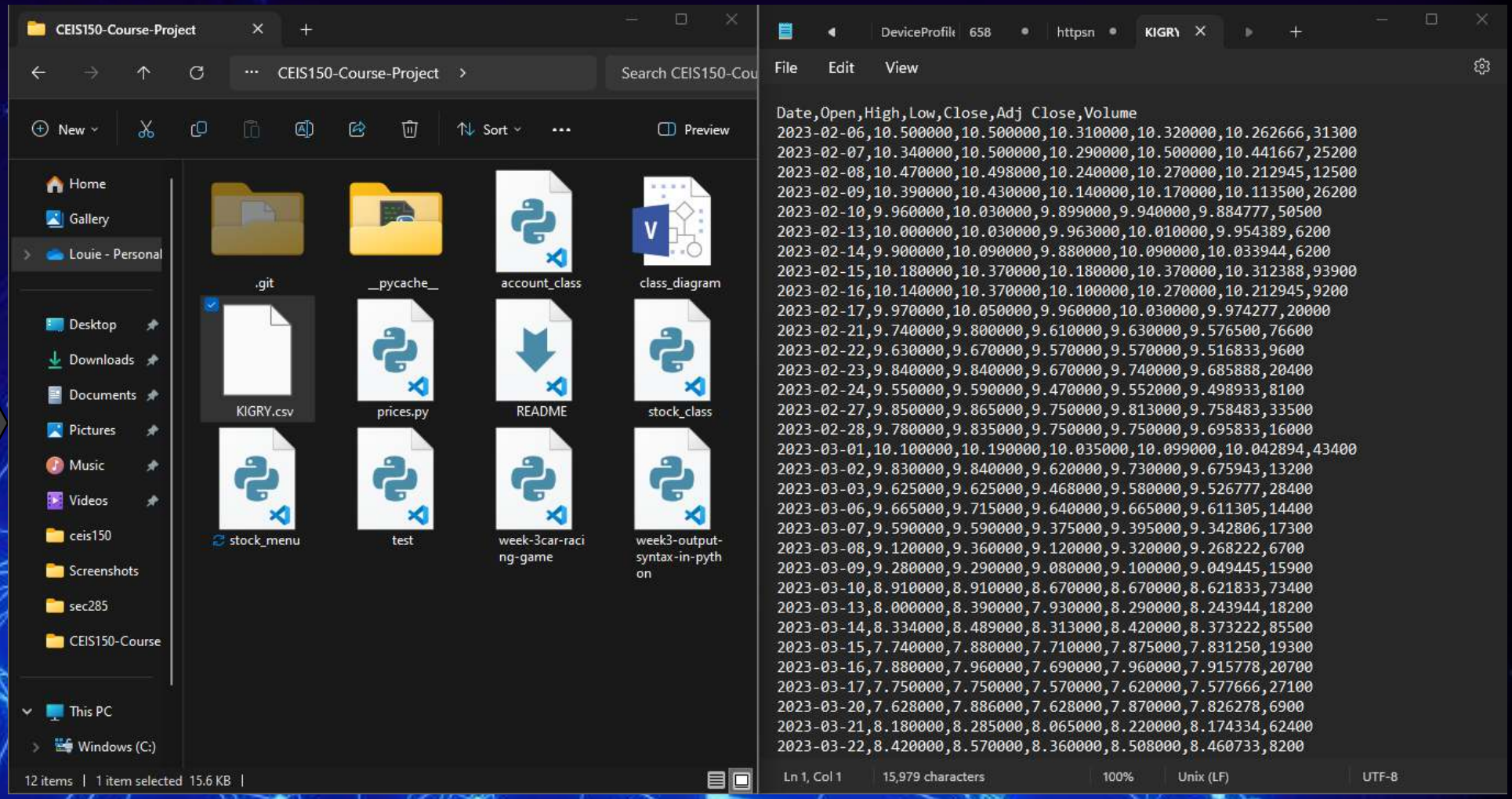
DeVry University

Louie Santino Venegas

Working with Files

Saving data and reading data from files is documented in this section. I used the CSV built in module to open read and extract data from a CSV file. The historical data was retrieved via yahoo finance. I did not use the yfinance library to pull from the web site with python, but I did experiment with it. The following slide is a picture of the raw imported text file and the location of the file in my file system. The following slide shows the stock data being used for two purposes to generate a profit and loss report and create a price chart.

Imported CSV from yahoo finance



Importing data from files generating profit loss reports and a price chart

C:\Users\Ughon\OneDrive\Desktop\ceis150\Course-Project\CEIS150-Course-Project\stock_menu.py

stockscrape.py × | stock_class.py × | account_class.py × | prices.py.py × | stock_menu.py* ×

```
# -*- coding: utf-8 -*-
"""
02/09/2024
Louie Santino Venegas d41112084 ceis150
"""
from datetime import datetime
from stock_class import Stock, DailyData
from account_class import  Traditional, Robo
import matplotlib.pyplot as plt
import csv


def add_stock(stock_list):
      option = ""
      while option != "0":
          print("Add Stock---")
          symbol = input("Enter Ticker Symbol: ")
          symbol = symbol.upper()
          name = input("Enter Company Name: ")
          try:
              shares = float(input("Enter Shares: "))
          except:
              print("Number Values Only!")
          new_stock = Stock(symbol, name, shares)
          stock_list.append(new_stock)
          option = input("Pess Enter to Add Another Stock or 0 to quit: ")

# Remove stock and all daily data
def delete_stock(stock_list):
    print("Delete Stock---")
    print("Stock List: [", end="")
    for stock in stock_list:
        print(stock.symbol, " ",end="")
    print("]")
    symbol = input("Which stock do you want to delete?").upper()
```

Console 1/A ×

```
Report for:  KIGRY kion group
Shares:  500.0
summary...
Low Price: $7.33
High Print: $11.61
Average Price: $9.56
Low Volume: 849900.0
Average Volume: 28,481.8
Change in Price: $4.28
profit/loss: $2,140.00



Report Complete!
```

IPython Console | History

conda (Python 3.11.5) | Completions: conda | LSP: Python | main [5] | Line 3, Col 2 | UTF-8 | CRLF | RW | Mem 79%

CEIS150

Programming With Objects

Creating a Graphical User Interface(GUI)

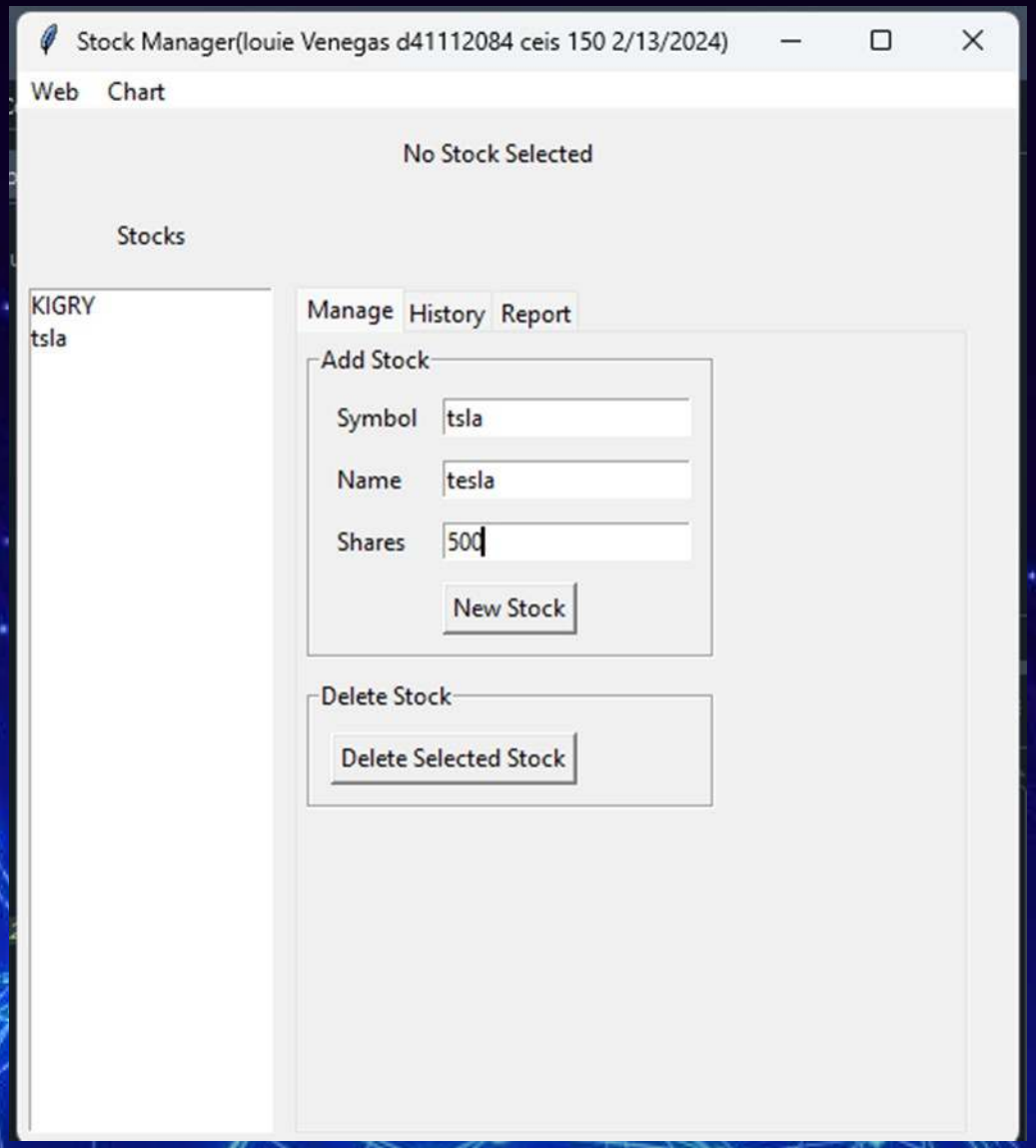
DeVry University

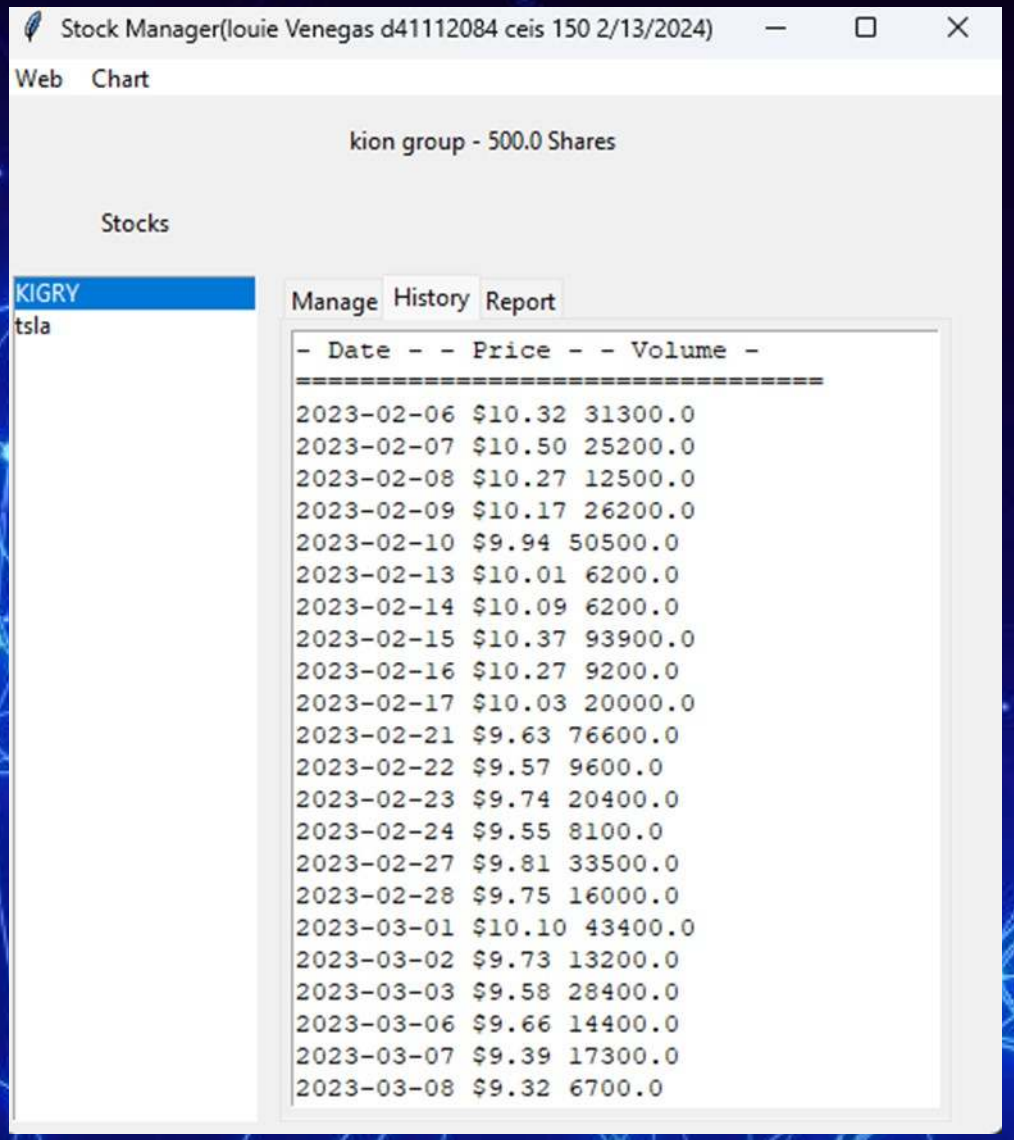
Louie Santino Venegas

Creating a Graphical User Interface

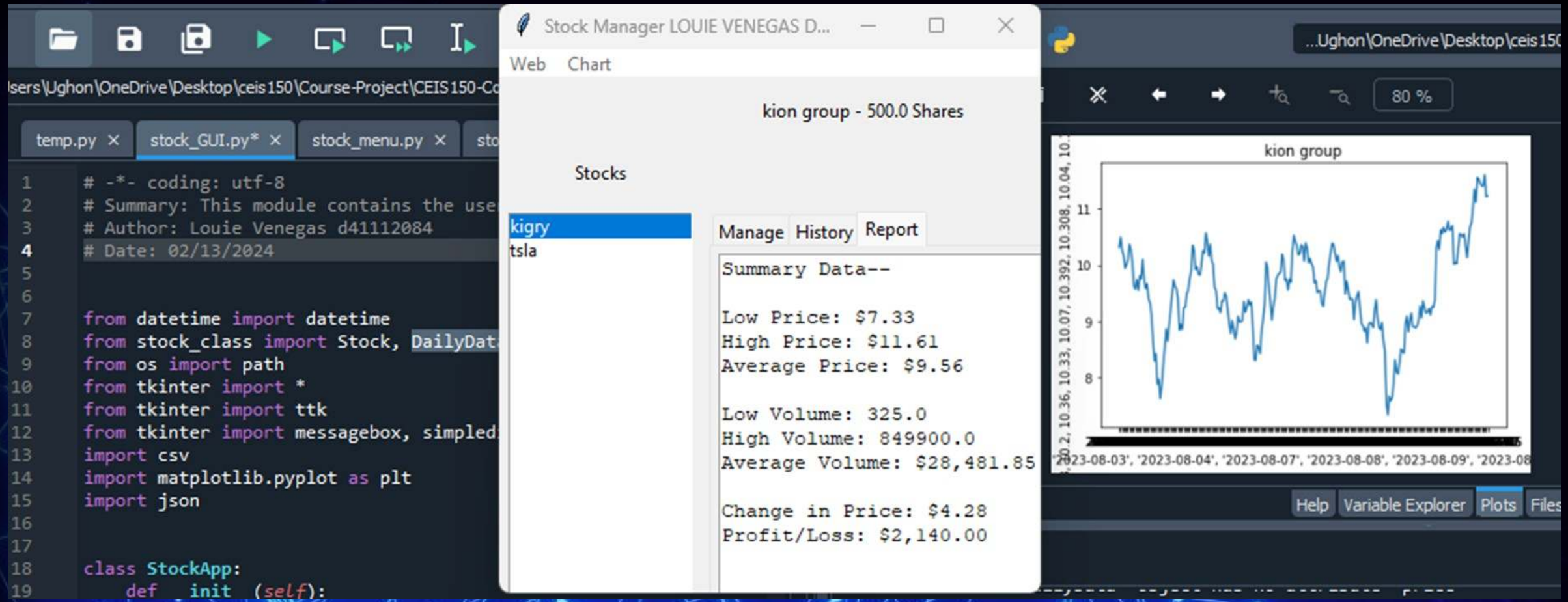
Using the tkinter library I created a version of this application using a graphical user interface to make it more user friendly. The following slides show the same functionalities created in the console-based version. In the creating of the GUI version, I used the same classes and recycled as much code as I could from the menu functions with slight modifications since tkinter requires alternate keywords.

My Stock Application’s Graphical User Interface





Historical Data Tab



Profit/loss report
And graph
Working in the GUI

Final Push to GitHub!