



CEIS150

Programming With Objects

Python Stock Tracking Project

DeVry University

Louie Santino Venegas

GitHub

Python Stock Tracking Project

Background This 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.

Spyder (Python 3.11)

File Edit Search Source Run Debug Consoles Projects Tools View Help

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

untitled0.py x untitled1.py x prices.py.py x

```
1 # -*- coding: utf-8 -*-
2 """
3 Louie Venegas
4 ceis150
5 January 5, 2024
6 """
7
8 count = 0          #initialize the count variable to 0
9 sum = 0            #initialize the sum variable to 0
10 Full_name = input("What is Your Full Nam:? ")          #input full_name
11 min_price = float(input("What is the Minimum Price?")) #input the min_price and convert it to float
12 price_list = [10, 25.5, 50, 22, 69.0, 71.0, 84.5, 91.0, 67.4, 81.2, 84.6, 58.8, 79.3, 101.2] #create
13 for price in price_list: #add current price to sum
14     sum += price
15     if price > min_price:
16         count += 1
17 print("Hello", Full_name, "the minimum price is $", min_price) #output "Hello", name, "the minimum price i
18 print("Hello {}, the minimum price is ${:,.2f}" .format(Full_name, min_price)) #output "There are ", co
19 print("There are ", count, "prices greater than the minimum price")
20 print("The total price is ${:,.2f}" .format(sum)) #output "The total price is ", sum
21
```

Name	Type	Size	Value
count	int	1	10
Full_name	str	13	Louie Venegas
min_price	float	1	50.0
price	float	1	101.2
price_list	list	14	[10, 25.5, 50, 22, 69.0, 71.0, 84.5, 91.0, 67.4, 81.2, ...]
sum	float	1	895.5

Help Variable Explorer Plots Files

Console 1/A x

There are 10 prices greater than the minimum price
The total price is \$895.50

In [2]: runfile('C:/Users/Ughon/Desktop/ceis150/Course-Project/CEIS150-Course-Project/prices.py.py', wdir='C:/Users/Ughon/Desktop/ceis150/Course-Project/CEIS150-Course-Project')

What is Your Full Nam:? Louie Venegas
What is the Minimum Price?50
Hello Louie Venegas the minimum price is \$ 50.0
Hello Louie Venegas, the minimum price is \$50.00
There are 10 prices greater than the minimum price
The total price is \$895.50

In [3]:

IPython Console History

conda (Python 3.11.5) Completions: conda LSP: Python Line 20, Col 49 UTF-8 CRLF RW Mem 67%

Prices.py (test program)



CEIS150

Programming With Objects

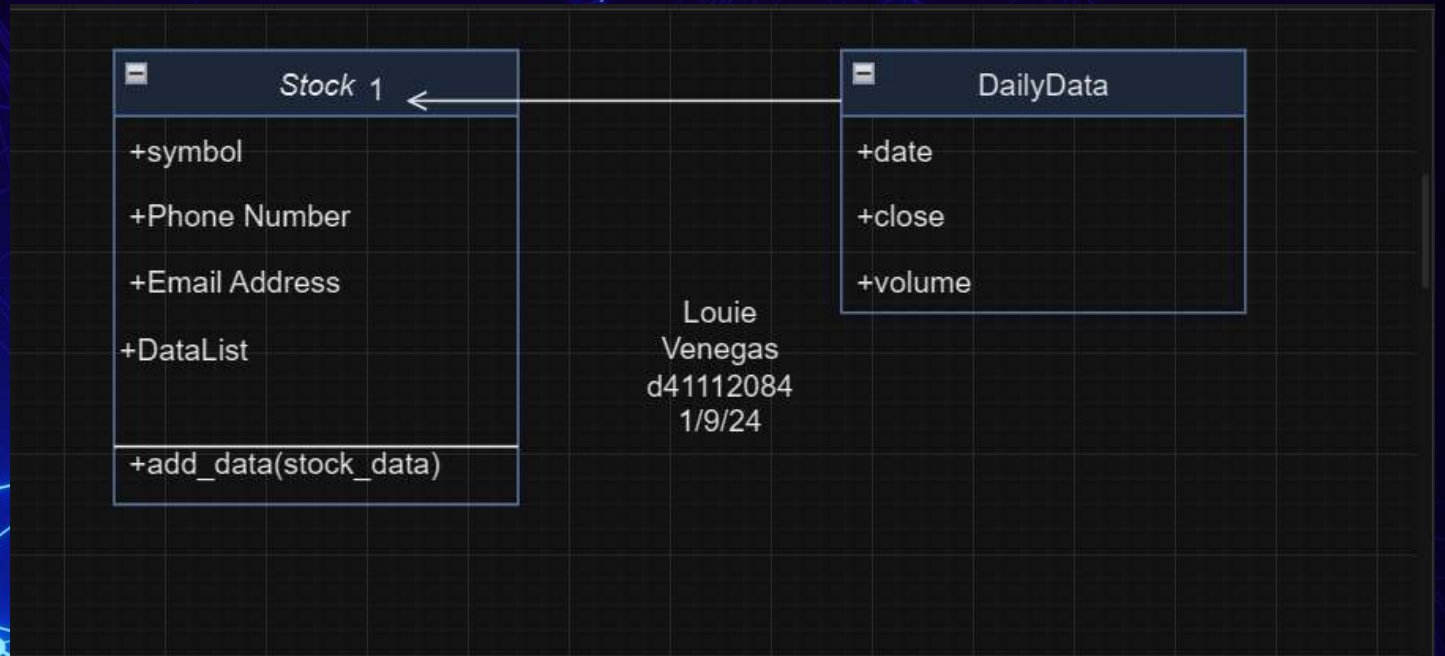
Creating the Classes

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




```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Jan  8 18:39:14 2024
4
5  Louie Santino Venegas
6  """
7  import warnings
8
9  class Stock:
10     def __init__(self, symbol, name, shares=0.0):
11         if not symbol or not name:
12             warnings.warn("You must include a symbol and a name.", UserWarning)
13         if shares < 0:
14             warnings.warn("Shares cannot be negative.", UserWarning)
15         self.symbol = symbol
16         self.name = name
17         self.shares = shares
18         self.DataList = [] #DailyData list
19
20     def add_data(self, stock_data):
21         self.DataList.append(stock_data)
22
23  class DailyData:
24     def __init__(self, date, close, volume):
25         self.date = date
26         self.close = close
27         self.volume = volume
28
```

Class Code

Unit Test

```
In [1]: from sys import path, os; path.append('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

The screenshot shows the GitHub interface for a repository named 'CEIS150-Course-Project' by user 'lvenegas4'. The repository is marked as 'Private'. The 'main' branch is selected, showing 1 branch and 0 tags. A search bar and buttons for 'Go to file', 'Add file', and 'Code' are visible. The commit history shows a single commit by 'Louie_Venegas' 1st, pushed 20 minutes ago. The commit details show four files: 'README.md' (Initial commit, last week), 'class_diagram.vsd' (1st, 20 minutes ago), 'prices.py.py' (1st, 20 minutes ago), and 'stock_class.py' (1st, 20 minutes ago).

The first screenshot shows the Git GUI 'Commit' window. It displays 'Unstaged Changes' and 'Staged Changes (Will Commit)' with 'stock_class.py' checked. The 'Commit Message' field contains '2nd'. The 'Ready to commit' status is shown at the bottom. The second screenshot shows the 'Push' window, indicating 'Pushing 1 branch to origin' to 'https://github.com/lvenegas4/CEIS150-Course-Project.git'. It shows the push progress and the 'Push' button.



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

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jan 16 16:17:00 2024
4
5  @author: Louie Santino Venegas d41112084
6  """
7  from datetime import datetime
8  from stock_class import Stock, DailyData
9  #from account_class import Traditional, Robo
10 import matplotlib.pyplot as plt
11 import csv
12
13
14 def add_stock(stock_list):
15     option = ""
16     while option != "0":
17         print("Adding A Stock")
18         symbol = input("Enter Symbol: ").upper()
19         name = input("Enter Company Name: ")
20         shares = float(input("Enter Shares: "))
21         new_stock = Stock(symbol, name, shares)
22         stock_list.append(new_stock)
23         option = input("Press Enter to Add Another Stock or 0 to quit: ")
24
25 # Remove stock and all daily data
26 def delete_stock(stock_list):
27     print("This method is under construction")
28
29
30 # List stocks being tracked
31 def list_stocks(stock_list):
32     print("List of Stocks")
33     print("SYMBOL\t\tNAME\t\tSHARES")
34     print("=====")
```

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

Help Variable Explorer Plots Files

Console 1/A X

```
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
Press Enter to Add Another Stock or 0 to quit:
Adding A Stock
Enter Symbol: tsla
Enter Company Name: Tesla
Enter Shares: 100
Press Enter to Add Another Stock or 0 to quit:
Adding A Stock
Enter Symbol: kigry
Enter Company Name: Kion Group
Enter Shares: 500
Press Enter to Add Another Stock or 0 to quit:
```

IPython Console History

Adding 3 Stocks

```
"""
Created on Tue Jan 16 16:17:00 2024
@author: 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("Press Enter to Add Another Stock or 0 to quit: ")
    24
```

```
# Remove stock and all daily data
```

```
def remove_stock(stock_list):
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
    """
```

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jan 16 16:17:00 2024
4  Louie Santino Venegas d41112084 ceis150
5  """
6  from datetime import datetime
7  from stock_class import Stock, DailyData
8  #from account_class import Traditional, Robo
9  import matplotlib.pyplot as plt
10 import csv
11
12
13 def add_stock(stock_list):
14     option = ""
15     while option != "0":
16         print("Add Stock---")
17         symbol = input("Enter Ticker Symbol: ")
18         symbol = symbol.upper()
19         name = input("Enter Company Name: ")
20         shares = float(input("Enter Shares: "))
21         new_stock = Stock(symbol, name, shares)
22         stock_list.append(new_stock)
23         option = input("Press Enter to Add Another Stock or 0 to Exit")
24
25 # Remove stock and all daily data
26 def delete_stock(stock_list):
27     print("Delete Stock---")
28     print("Stock List: /", end="")
29     for stock in stock_list:
30         print(stock.symbol, " ", end="")
31     print("\n")
32     symbol = input("Which stock do you want to delete?").upper()
33     found = False
34     i = 0
```

Usage

Usage: You can not have an array object for

Help Variable Explorer Plots Files

Console UJA x

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: 4

Add Daily Stock Data---

Stock List: [KIGRY MSFT TSLA]

Which stock do you want to use?: kigry

Ready to add data for: KIGRY

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: 1/17/2023,10.09,5921

Enter Date,Price,Volume: 1/17/2024,10.13,6756

Enter Date,Price,Volume:

Date Entry Complete

Adding Stock Data


```
MINGW64:/c/Users/Ughon/OneDrive/Desktop/ceis150/Course-Project/CEIS150-Co...
Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git add --all

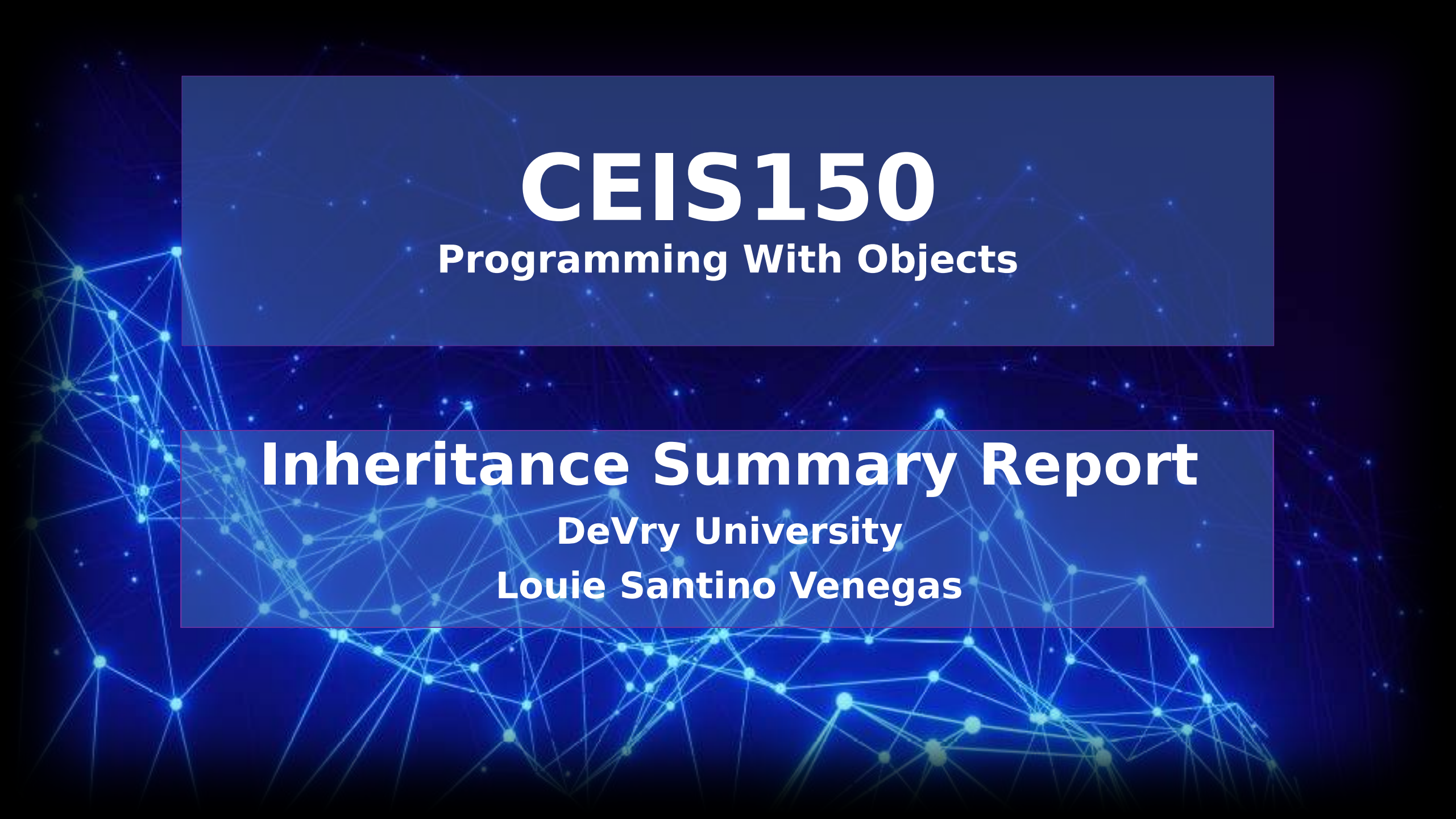
Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git commit -m "Module 3"
[main 125f051] Module 3
3 files changed, 167 insertions(+), 2 deletions(-)
create mode 100644 __pycache__/stock_class.cpython-311.pyc
create mode 100644 stock_menu.py

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git push ^[[200~https://github.com/1venegas4/CEIS150-Course-Project.git~
fatal: protocol '[200~https' is not supported

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git push https://github.com/1venegas4/CEIS150-Course-Project.git
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 4.90 KiB | 4.90 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/1venegas4/CEIS150-Course-Project.git
841a573..125f051 main -> main

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$
```

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

```
# -*- 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
```

```
How many years until retirement: 25
Your investment return is 13125.0

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: 6
Investment Account ---
What is your initial balance: 500
What is your account number: LouieVenegasD41112084
Do you want a Traditional (t) or Robo (r) account: t
Choose stocks from the list below:
Stock List: [TSLA KIGRY GMC ]

IPython Console History
P: Python main [4] Line 126, Col 1 UTF-8 CRLF RW Mem 78%
```

Stock Menu Program


```
MINGW64:/c/Users/Ughon/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git add --all

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git commit -m "Module 4"
[main 4f0fe89] Module 4
 5 files changed, 192 insertions(+), 7 deletions(-)
 create mode 100644 __pycache__/account_class.cpython-311.pyc
 create mode 100644 account_class.py

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$ git push https://github.com/lvenegas4/CEIS150-Course-Project.git
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 7.67 KiB | 3.83 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/lvenegas4/CEIS150-Course-Project.git
 125f051..4f0fe89  main -> main

Ughon@LAPTOP-JM7BLMFQ MINGW64 ~/OneDrive/Desktop/ceis150/Course-Project/CEIS150-
Course-Project (main)
$
```

Git to GitHub

GitHub Repository

 Ivenegas4 / CEIS150-Course-Project 

Type  to search

 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

 **CEIS150-Course-Project** Private Unwatch 1

 main  1 Branch  0 Tags  Add file  Code

 **Louie_Venegas** Module 4 4f0fe89 · 1 minute ago  4 Commits

 <code>__pycache__</code>	Module 4	1 minute ago
 <code>README.md</code>	Initial commit	3 weeks ago
 <code>account_class.py</code>	Module 4	1 minute ago
 <code>class_diagram.vsd</code>	1st	2 weeks ago
 <code>prices.py.py</code>	1st	2 weeks ago
 <code>stock_class.py</code>	Module 4	1 minute ago
 <code>stock_menu.py</code>	Module 4	1 minute ago



CEIS150

Programming With Objects

Creating a Stock Chart With matplotlib.pyplot

DeVry University
Louie Santino Venegas

Creating a Stock Chart With matplotlib.pyplot

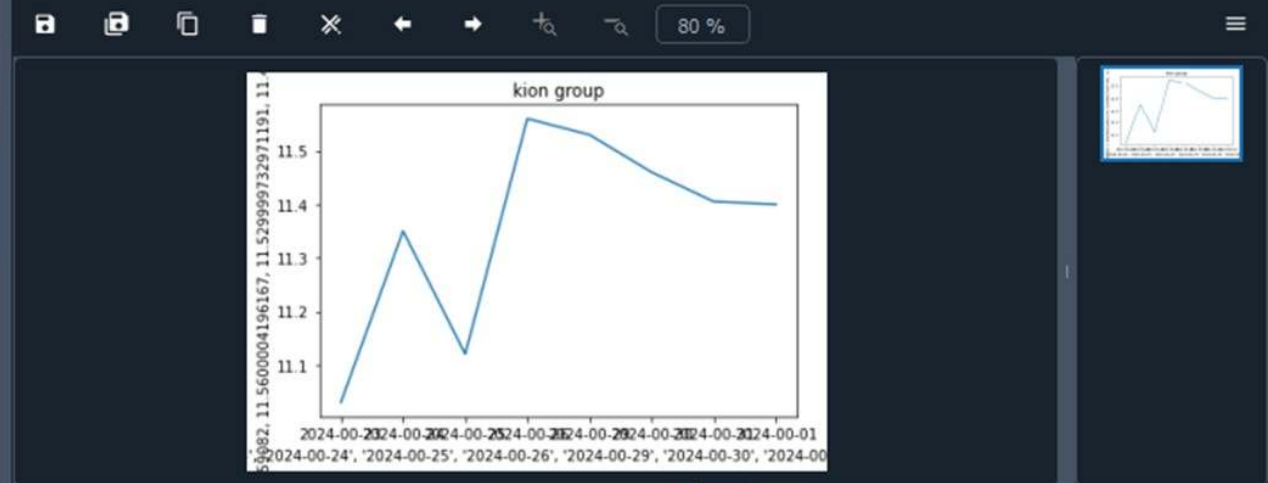
No stock application would be complete unless it could produce a stock chart... Which is menu selection number 5. If you have data inputted into the system about your stock this application will produce a basic price chart using the price change over time to create the price line. In order to produce 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 x account_class.py x stock_class.py x stock_menu.py* x

1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue feb 1, 16:17:00 2024
4  Louie Santino Venegas d41112084 ceis150
5  """
6  from datetime import datetime
7  from stock_class import Stock, DailyData
8  from account_class import Traditional, Robo
9  import matplotlib.pyplot as plt
10 import csv
11
12
13 def add_stock(stock_list):
14     option = ""
15     while option != "0":
16         print("Add Stock---")
17         symbol = input("Enter Ticker Symbol: ")
18         symbol = symbol.upper()
19         name = input("Enter Company Name: ")
20         shares = float(input("Enter Shares: "))
21         new_stock = Stock(symbol, name, shares)
22         stock_list.append(new_stock)
23         option = input("Press Enter to Add Another Stock or 0 to quit: ")
24
25 # Remove stock and all daily data
26 def delete_stock(stock_list):
27     print("Delete Stock---")
28     print("Stock List: [", end="")
29     for stock in stock_list:
30         print(stock.symbol, " ", end="")
31     print("]")
32     symbol = input("Which stock do you want to delete?").upper()
33     found = False
34     i = 0
35     while i < len(stock_list):
36         if stock_list[i].symbol == symbol:
37             del stock_list[i]
38             print("Stock Deleted")
39             found = True
40             break
41         i += 1
42     if not found:
43         print("Stock not found")
44
45 if __name__ == '__main__':
46     stock_list = []
47     while True:
48         print("Menu")
49         print("0 - Exit Program")
50         print("1 - Add Stock")
51         print("2 - Delete Stock")
52         print("3 - Display Stock List")
53         print("4 - Display Stock Data")
54         print("5 - Display Stock Chart")
55         option = input("Enter Menu Option: ")
56         if option == "0":
57             break
58         elif option == "1":
59             add_stock(stock_list)
60         elif option == "2":
61             delete_stock(stock_list)
62         elif option == "3":
63             display_stock_list(stock_list)
64         elif option == "4":
65             display_stock_data(stock_list)
66         elif option == "5":
67             display_stock_chart(stock_list)
68         else:
69             print("Invalid Option")
70
71 def display_stock_list(stock_list):
72     if len(stock_list) == 0:
73         print("No stocks in list")
74     else:
75         print("Stock List")
76         for stock in stock_list:
77             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
78         print()
79
80 def display_stock_data(stock_list):
81     if len(stock_list) == 0:
82         print("No stocks in list")
83     else:
84         print("Stock Data")
85         for stock in stock_list:
86             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
87             print(stock.get_daily_data())
88             print()
89
90 def display_stock_chart(stock_list):
91     if len(stock_list) == 0:
92         print("No stocks in list")
93     else:
94         print("Stock Chart")
95         for stock in stock_list:
96             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
97             stock.get_daily_data()
98             plt.plot(stock.get_daily_data())
99             plt.show()
100             print()
101
102 def display_stock_summary(stock_list):
103     if len(stock_list) == 0:
104         print("No stocks in list")
105     else:
106         print("Stock Summary")
107         for stock in stock_list:
108             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
109             print(stock.get_summary())
110             print()
111
112 def display_stock_report(stock_list):
113     if len(stock_list) == 0:
114         print("No stocks in list")
115     else:
116         print("Stock Report")
117         for stock in stock_list:
118             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
119             print(stock.get_report())
120             print()
121
122 def display_stock_analysis(stock_list):
123     if len(stock_list) == 0:
124         print("No stocks in list")
125     else:
126         print("Stock Analysis")
127         for stock in stock_list:
128             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
129             print(stock.get_analysis())
130             print()
131
132 def display_stock_forecast(stock_list):
133     if len(stock_list) == 0:
134         print("No stocks in list")
135     else:
136         print("Stock Forecast")
137         for stock in stock_list:
138             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
139             print(stock.get_forecast())
140             print()
141
142 def display_stock_recommendation(stock_list):
143     if len(stock_list) == 0:
144         print("No stocks in list")
145     else:
146         print("Stock Recommendation")
147         for stock in stock_list:
148             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
149             print(stock.get_recommendation())
150             print()
151
152 def display_stock_alert(stock_list):
153     if len(stock_list) == 0:
154         print("No stocks in list")
155     else:
156         print("Stock Alert")
157         for stock in stock_list:
158             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
159             print(stock.get_alert())
160             print()
161
162 def display_stock_news(stock_list):
163     if len(stock_list) == 0:
164         print("No stocks in list")
165     else:
166         print("Stock News")
167         for stock in stock_list:
168             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
169             print(stock.get_news())
170             print()
171
172 def display_stock_events(stock_list):
173     if len(stock_list) == 0:
174         print("No stocks in list")
175     else:
176         print("Stock Events")
177         for stock in stock_list:
178             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
179             print(stock.get_events())
180             print()
181
182 def display_stock_transactions(stock_list):
183     if len(stock_list) == 0:
184         print("No stocks in list")
185     else:
186         print("Stock Transactions")
187         for stock in stock_list:
188             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
189             print(stock.get_transactions())
190             print()
191
192 def display_stock_holdings(stock_list):
193     if len(stock_list) == 0:
194         print("No stocks in list")
195     else:
196         print("Stock Holdings")
197         for stock in stock_list:
198             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
199             print(stock.get_holdings())
200             print()
201
202 def display_stock_performance(stock_list):
203     if len(stock_list) == 0:
204         print("No stocks in list")
205     else:
206         print("Stock Performance")
207         for stock in stock_list:
208             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
209             print(stock.get_performance())
210             print()
211
212 def display_stock_risk(stock_list):
213     if len(stock_list) == 0:
214         print("No stocks in list")
215     else:
216         print("Stock Risk")
217         for stock in stock_list:
218             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
219             print(stock.get_risk())
220             print()
221
222 def display_stock_valuation(stock_list):
223     if len(stock_list) == 0:
224         print("No stocks in list")
225     else:
226         print("Stock Valuation")
227         for stock in stock_list:
228             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
229             print(stock.get_valuation())
230             print()
231
232 def display_stock_dividend(stock_list):
233     if len(stock_list) == 0:
234         print("No stocks in list")
235     else:
236         print("Stock Dividend")
237         for stock in stock_list:
238             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
239             print(stock.get_dividend())
240             print()
241
242 def display_stock_splits(stock_list):
243     if len(stock_list) == 0:
244         print("No stocks in list")
245     else:
246         print("Stock Splits")
247         for stock in stock_list:
248             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
249             print(stock.get_splits())
250             print()
251
252 def display_stock_earnings(stock_list):
253     if len(stock_list) == 0:
254         print("No stocks in list")
255     else:
256         print("Stock Earnings")
257         for stock in stock_list:
258             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
259             print(stock.get_earnings())
260             print()
261
262 def display_stock_revenue(stock_list):
263     if len(stock_list) == 0:
264         print("No stocks in list")
265     else:
266         print("Stock Revenue")
267         for stock in stock_list:
268             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
269             print(stock.get_revenue())
270             print()
271
272 def display_stock_expenses(stock_list):
273     if len(stock_list) == 0:
274         print("No stocks in list")
275     else:
276         print("Stock Expenses")
277         for stock in stock_list:
278             print(stock.symbol, " ", stock.name, " ", stock.shares, " ")
279             print(stock.get_expenses())
280             print()
275
```



Help Variable Explorer Plots Files

Console 1/A x

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



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

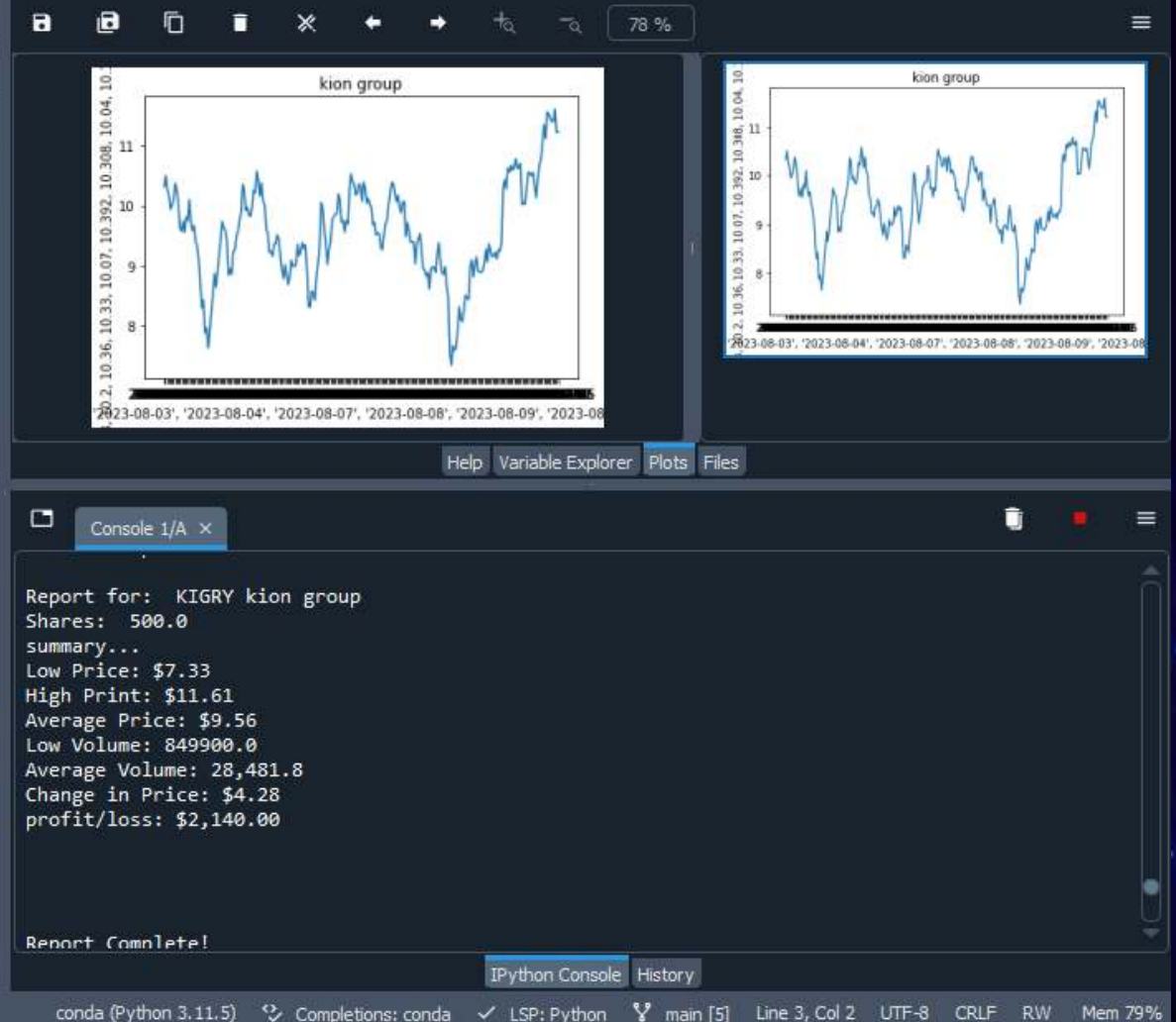
The image shows a Windows File Explorer window and a code editor. The File Explorer window displays the contents of the 'CEIS150-Course-Project' directory, including folders like '.git', '__pycache__', and files like 'KIGRY.csv', 'prices.py', 'README', 'stock_class', 'stock_menu', 'test', 'week-3car-racing-game', and 'week3-output-syntax-in-python'. The code editor shows the contents of 'KIGRY.csv', which is a CSV file containing stock price data for KIGRY. The data is organized into columns: Date, Open, High, Low, Close, Adj Close, and Volume. The data spans from 2023-02-06 to 2023-03-22.

Date	Open	High	Low	Close	Adj Close	Volume
2023-02-06	10.500000	10.500000	10.310000	10.320000	10.262666	31300
2023-02-07	10.340000	10.500000	10.290000	10.500000	10.441667	25200
2023-02-08	10.470000	10.498000	10.240000	10.270000	10.212945	12500
2023-02-09	10.390000	10.430000	10.140000	10.170000	10.113500	26200
2023-02-10	9.960000	10.030000	9.899000	9.940000	9.884777	50500
2023-02-13	10.000000	10.030000	9.963000	10.010000	9.954389	6200
2023-02-14	9.900000	10.090000	9.880000	10.090000	10.033944	6200
2023-02-15	10.180000	10.370000	10.180000	10.370000	10.312388	93900
2023-02-16	10.140000	10.370000	10.100000	10.270000	10.212945	9200
2023-02-17	9.970000	10.050000	9.960000	10.030000	9.974277	20000
2023-02-21	9.740000	9.800000	9.610000	9.630000	9.576500	76600
2023-02-22	9.630000	9.670000	9.570000	9.570000	9.516833	9600
2023-02-23	9.840000	9.840000	9.670000	9.740000	9.685888	20400
2023-02-24	9.550000	9.590000	9.470000	9.552000	9.498933	8100
2023-02-27	9.850000	9.865000	9.750000	9.813000	9.758483	33500
2023-02-28	9.780000	9.835000	9.750000	9.750000	9.695833	16000
2023-03-01	10.100000	10.190000	10.035000	10.099000	10.042894	43400
2023-03-02	9.830000	9.840000	9.620000	9.730000	9.675943	13200
2023-03-03	9.625000	9.625000	9.468000	9.580000	9.526777	28400
2023-03-06	9.665000	9.715000	9.640000	9.665000	9.611305	14400
2023-03-07	9.590000	9.590000	9.375000	9.395000	9.342806	17300
2023-03-08	9.120000	9.360000	9.120000	9.320000	9.268222	6700
2023-03-09	9.280000	9.290000	9.080000	9.100000	9.049445	15900
2023-03-10	8.910000	8.910000	8.670000	8.670000	8.621833	73400
2023-03-13	8.000000	8.390000	7.930000	8.290000	8.243944	18200
2023-03-14	8.334000	8.489000	8.313000	8.420000	8.373222	85500
2023-03-15	7.740000	7.880000	7.710000	7.875000	7.831250	19300
2023-03-16	7.880000	7.960000	7.690000	7.960000	7.915778	20700
2023-03-17	7.750000	7.750000	7.570000	7.620000	7.577666	27100
2023-03-20	7.628000	7.886000	7.628000	7.870000	7.826278	6900
2023-03-21	8.180000	8.285000	8.065000	8.220000	8.174334	62400
2023-03-22	8.420000	8.570000	8.360000	8.508000	8.460733	8200

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 x stock_class.py x account_class.py x prices.py.py x stock_menu.py* x
1 # -*- coding: utf-8 -*-
2 """
3 02/09/2024
4 Louie Santino Venegas d41112084 ceis150
5 """
6 from datetime import datetime
7 from stock_class import Stock, DailyData
8 from account_class import Traditional, Robo
9 import matplotlib.pyplot as plt
10 import csv
11
12
13 def add_stock(stock_list):
14     option = ""
15     while option != "0":
16         print("Add Stock---")
17         symbol = input("Enter Ticker Symbol: ")
18         symbol = symbol.upper()
19         name = input("Enter Company Name: ")
20         try:
21             shares = float(input("Enter Shares: "))
22         except:
23             print("Number Values Only!")
24         new_stock = Stock(symbol, name, shares)
25         stock_list.append(new_stock)
26         option = input("Press Enter to Add Another Stock or 0 to quit: ")
27
28 # Remove stock and all daily data
29 def delete_stock(stock_list):
30     print("Delete Stock---")
31     print("Stock List: [", end="")
32     for stock in stock_list:
33         print(stock.symbol, " ", end="")
34     print("]")
35     symbol = input("Which stock do you want to delete?").upper()
```





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

Stock Manager(louie Venegas d41112084 ceis 150 2/13/2024)

Web Chart

No Stock Selected

Stocks

KIGRY
tsla

Manage History Report

Add Stock

Symbol

Name

Shares

Delete Stock

Stock Manager(louie Venegas d41112084 ceis 150 2/13/2024)

Web Chart

kion group - 500.0 Shares

Stocks

KIGRY
tsla

Manage History Report

- Date -	- Price -	- Volume -
2023-02-06	\$10.32	31300.0
2023-02-07	\$10.50	25200.0
2023-02-08	\$10.27	12500.0
2023-02-09	\$10.17	26200.0
2023-02-10	\$9.94	50500.0
2023-02-13	\$10.01	6200.0
2023-02-14	\$10.09	6200.0
2023-02-15	\$10.37	93900.0
2023-02-16	\$10.27	9200.0
2023-02-17	\$10.03	20000.0
2023-02-21	\$9.63	76600.0
2023-02-22	\$9.57	9600.0
2023-02-23	\$9.74	20400.0
2023-02-24	\$9.55	8100.0
2023-02-27	\$9.81	33500.0
2023-02-28	\$9.75	16000.0
2023-03-01	\$10.10	43400.0
2023-03-02	\$9.73	13200.0
2023-03-03	\$9.58	28400.0
2023-03-06	\$9.66	14400.0
2023-03-07	\$9.39	17300.0
2023-03-08	\$9.32	6700.0

Historical Data Tab

temp.py × stock_GUI.py* × stock_menu.py × sto

```
1  # -*- coding: utf-8
2  # Summary: This module contains the use
3  # Author: Louie Venegas d41112084
4  # Date: 02/13/2024
5
6
7  from datetime import datetime
8  from stock_class import Stock, DailyData
9  from os import path
10 from tkinter import *
11 from tkinter import ttk
12 from tkinter import messagebox, simplified
13 import csv
14 import matplotlib.pyplot as plt
15 import json
16
17
18 class StockApp:
19     def __init__(self):
```

Stock Manager LOUIE VENEGAS D...

Web Chart

kion group - 500.0 Shares

Stocks

kigry
tsla

Manage History Report

Summary Data--

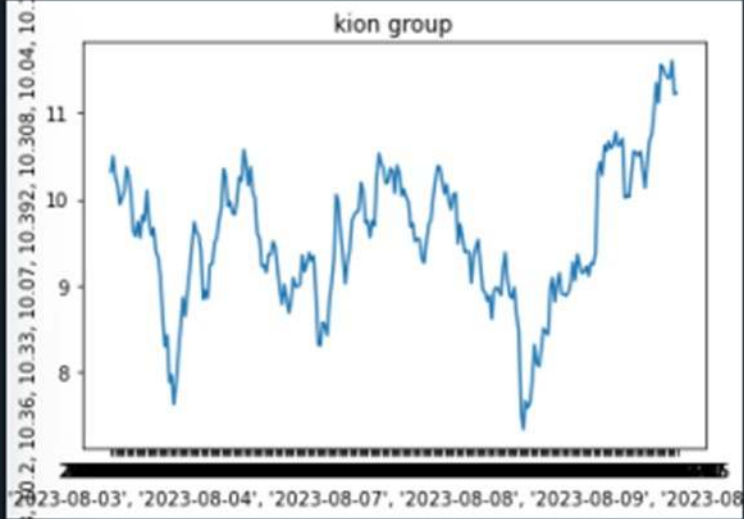
Low Price: \$7.33
High Price: \$11.61
Average Price: \$9.56

Low Volume: 325.0
High Volume: 849900.0
Average Volume: \$28,481.85

Change in Price: \$4.28
Profit/Loss: \$2,140.00

...Ughon\OneDrive\Desktop\ceis150

80 %



Help Variable Explorer Plots Files

Profit/loss report
And graph
Working in the GUI

Final Push to GitHub!

 **CEIS150-Course-Project** Private Unwatch 1

 main  1 Branch  0 Tags Add file <> Code

 **Louie_Venegas** Module #7 97f45de · 1 minute ago 6 Commits

 <code>__pycache__</code>	Module #7	1 minute ago
 <code>KIGRY.csv</code>	Module 7	yesterday
 <code>README.md</code>	Initial commit	last month
 <code>account_class.py</code>	Module 4	3 weeks ago
 <code>class_diagram.vsd</code>	1st	last month
 <code>prices.py.py</code>	1st	last month
 <code>stock_GUI.py</code>	Module #7	1 minute ago
 <code>stock_class.py</code>	Module 4	3 weeks ago
 <code>stock_menu.py</code>	Module #7	1 minute ago
 <code>test.py</code>	Module 7	yesterday
 <code>week-3car-racing-game.py</code>	Module 7	yesterday
 <code>week3-output-syntax-in-python.py</code>	Module 7	yesterday