Studying Finance with Python

Sam Qunell

February 16 2025

Welcome (back) to the Math Circle! This week, we will be learning how to write your own financial programs. We will be using the yfinance module in Python. This module pulls data from Yahoo Finance and has built-in features for analyzing the data. Be warned that this data is not always completely up to date!

We will use Google Colab to run our code since it is already compatible with yfinance. Some online Python shells (like trinket.io) do not have this feature. If you are running your own Python installation, you may have to install yfinance yourself.

Problem 1. Run the following code on your computer.

```
import yfinance as yf
ticker = 'AAPL'
data = yf.download(ticker, start= '2020-01-01', end='2023-01-01')
data.Close.plot(title='Apple Daily Prices')
```

Let's understand each line of this code snippet.

```
import yfinance as yf
```

This line tells Python to read from the yfinance module and to abbreviate the module's name to yf. Even though our Python environment has yfinance installed, it is not turned on by default in our environment.

```
ticker = 'AAPL'
```

This line creates a variable called 'ticker' whose value is 'AAPL'. A ticker is just how we keep track of the data of an individual company. Each company on the stock market has a ticker symbol, usually no more than 4 letters. Can you figure out which one this is?

```
data = yf.download(ticker, start= '2020-01-01', end='2023-01-01')
```

This line downloads the financial data for the company we have selected and stores it in a variable called 'data'. yf.download() is the command that tells yfinance to download data. The first argument, ticker, is telling yfinance which company we want the data for. This must always be included. There are two optional arguments included afterwards, 'start' and 'end'. We are telling yfinance that we only want the data in a certain range.

```
data.Close.plot(title='Apple Daily Prices')
```

This line plots some of the data that we just downloaded. Specifically, we plot the end-of-day prices of our chosen company. data.Close picks out only closing price data, and data.Close.plot() tells Python to plot this.

Problem 2. Modify the code so that it plots the closing prices for Nvidia from December 1st 2024 to February 10th 2025. The ticker symbol for Nvidia is 'NVDA'.

You might notice that the prices dropped pretty sharply within the past month. This is because of the AI company DeepSeek. Nvidia was making a lot of its money by capitalizing on the rise in AI, but the emergence of a cheaper, more efficient competitor has challenged their supremacy.

We can also print out the numerical stock prices via

```
print(data)
```

Problem 3. Which day had the biggest drop in Nvidia's closing prices? By how much did their stock price fall?

Nvidia's losses on that day were historic. Nvidia lost almost 600 billion dollars, or about 17 percent of its share value. This was the largest single-day U.S. company capital loss in history (so far).

Problem 4. Before January, what was the most recent date that Nvidia's stock was worth less than 120 dollars?

Problem 5. How much has Nvidia's stock price risen since January 27th?

Problem 6. If I invested 1000 dollars into Nvidia on December 1st, 2024, how much would my investment be worth on January 27th 2025? What about if I had instead invested on January 1st 2024?

Problem 7. The 7 most valuable tech companies in the world by market cap are Apple, Nvidia, Meta, Microsoft, Tesla, Alphabet, and Amazon (in some order). How much did each of their stock prices change on January 27th from January 24th? You will have to look up some of their ticker symbols!

Company	January 24th Closing Price	January 27th Closing Price	Total Price Change
Alphabet			
Amazon			
Apple			
Meta			
Microsoft			
Nvidia			
Tesla			

We can download data from multiple companies at once by modifying the first argument of yf.download. The following code will fetch data for both Apple and Nvidia.

```
data = yf.download({'NVDA', 'AAPL'}, start= '2024-09-01', end='2025-02-01')
```

Problem 8. Plot both Nvidia and Microsoft's closing stock prices between February 1st 2024 and February 1st 2025. Both companies should be displayed in the same plot!

When making investments, the exact stock price is not as important as the returns. We can switch to plotting returns with the following adjustment. The code

```
data.Close.pct_change()
```

will retrieve the daily percent change in closing stock price.

Problem 9. Update your code to plot the daily returns on these two companies in the same time period.

Problem 10. From a quick glance, which of the two companies has the higher stock price? Which one has the higher annual returns? Which one has higher daily volatility?

Problem 11. Nvidia was actually worth more as a company than Microsoft was before the crash. Why do you think the stock prices don't reflect this?

Problem 12. Compute the mean stock price of Nvidia from February 16th 2024 to February 16th 2025. You can do this by adding

```
.mean()
```

to the end of whatever list you want the mean of . Do you think this is a good measure for the strength of an investment? Why or why not?

We will compute some more statistics for these 7 tech companies. We will start with the annualized returns. If we are making a long-term investment, then day-to-day fluctuations may not concern us. We are only interested in how much our investment will be worth in the next few years. In python, we can compute the annual returns in a few steps.

1. Retrieve the data at the start and at the end of the year you want. We will be comparing February 16 2024 to February 16 2025.

2. Compute $R_{ann} = \frac{P_{2025} - P_{2024}}{P_{2024}}$

Problem 13. Compute the annual returns (February 16 2024 to February 16 2025) for each of the 7 tech companies from before.

Company	Annual Returns
Alphabet	
Amazon	
Apple	
Meta	
Microsoft	
Nvidia	
Tesla	

We will now look at the volatility. Recall that the volatility is the standard deviation of the daily returns over the course of a year. It is a measurement of how much these returns can change, or of how 'chaotic' the returns can be. In python, we can compute the annual volatility in four steps.

1. Retrieve the last year's worth of data for the company. You can do this with yfinance with the code

```
df.tail(252).Close.pct_change()
```

   Try to remember what the number 252 means!

2. Multiply by 100. This means that returns of 50 percent will have value in our list of 50 instead of .5.

3. Take the standard deviation. If you have a list of numbers in Python that you want to take the standard deviation of, you just need to add

```
.std()
```

   at the end.

4. Multiply by $\sqrt{252}$. This is where we convert from a daily to an annual measurement. You can compute the square root of, say, 49 in Python via the command

```
np.sqrt(49)
```

   Make sure to run

```
import numpy as np
```

   at the top of your program!

Problem 14. Compute the annual volatility (February 16 2024 to February 16 2025) for each of the 7 tech companies from before. Make sure that the data you download includes at least the past year of trades. You will probably want to do the companies one at a time.

Company	Annualized Volatility
Alphabet	
Amazon	
Apple	
Meta	
Microsoft	
Nvidia	
Tesla	

When investing, one usually wants to consider both the expected returns and the volatility together. If an investment has high returns but high volatility, then we may not earn as much as we expect. We will compute what is known as the Sharpe ratio to determine how much 'additional' returns there are per unit of volatility. Of course, we need to compare to the risk-free returns. We will assume that the risk-free returns were about 4.5 percent in 2024.

$$S = \frac{\text{Expected Yearly Returns} - \text{Risk-free yearly returns}}{\text{Volatility}}$$

Problem 15. Based on your previous computations, compute the Sharpe Ratio for each of the 7 tech companies from before.

Company	Sharpe Ratio
Alphabet	
Amazon	
Apple	
Meta	
Microsoft	
Nvidia	
Tesla	

Problem 16. Based on the historical data from the previous three tables, which company do you think is the better investment?

Bonus Problems

All of the problems from this page forward are bonus problems. Some of these may be much more difficult than were the previous problems, some are much more open-ended and conceptual, and some are just on unrelated material that you may find interesting. Feel free to skip around this section.

Problem 17. In early 2013, a certain man lost a hard drive that contained 8000 Bitcoin. How much is that wallet worth today?

Problem 18. If we assume the Bitcoin in this hard drive was all obtained on August 1st 2013, then what would be the return on his investment if he finds the wallet today?

Problem 19. Mining a block of Bitcoin means getting a computer to solve a math problem very quickly. Every 4 years, the Bitcoin reward for mining a block is halved. In 2009, mining a block would earn you 50 Bitcoin. What is the first year when mining a block will earn you less than 1 Bitcoin?

Problem 20. Approximately 144 blocks of Bitcoin are mined every day. This is by design, so that the value does not deflate too quickly. Since the the total number of Bitcoin per mined block halves every four years, we can use geometric series to deduce that the total amount of Bitcoin on the market is bounded above. Suppose that mining started on January 1st 2009. What is the maximum amount of bitcoin that will ever exist?