Python Finance Fundamentals

PIERIAN DATA

Python for Finance

- Now we will begin to really focus on using Python for finance and trading!
- In this section we will cover some key concepts in regards to trading.

Python for Finance

- We will discuss:
    - Portfolio Allocation
    - Sharpe Ratio
    - Portfolio Optimization
    - Efficient Frontier
    - CAPM - Capital Asset Pricing Model

Python for Finance

- We will also be introducing several other financial topics, like how the market works, how orders get made, and general arbitrage opportunities.

Python for Finance

- We will interchange shorter quick discussions of general finance topics, with longer, more detailed discussions of the actual implementation of these topics.
- Let’s get started!

Portfolio and Statistics

Python for Finance

- So far we've really only analyzed individual stocks, now we will shift our attention to understanding a portfolio of multiple stocks.
- So what is a portfolio?

Python for Finance

- Put simply a portfolio is just a set of allocations in a variety of securities.
- For example:
    - 20 % in APPL
    - 30 % in FB
    - 50% in GOOG

Python for Finance

- These percentages should add up to 100% (or if defined as weights they should add up to 1).
- Let’s review a few key statistics for a portfolio!

Python for Finance

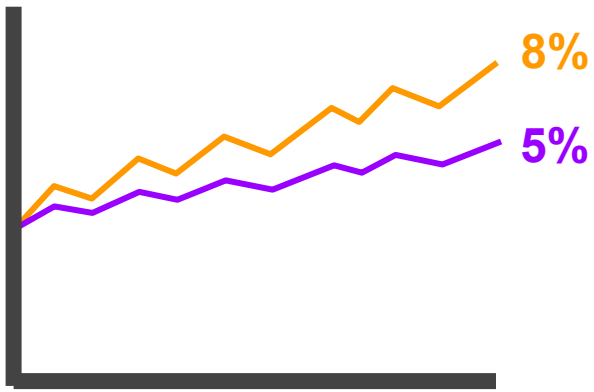
- Daily Returns - The percent returned from 1 day to the next for a stock.
- Cumulative Return - The amount returned after an entire time period.
- Avg. Daily Return - Mean of Daily Returns
- Std. Daily Return - Std. Dev of Daily Returns

Python for Finance

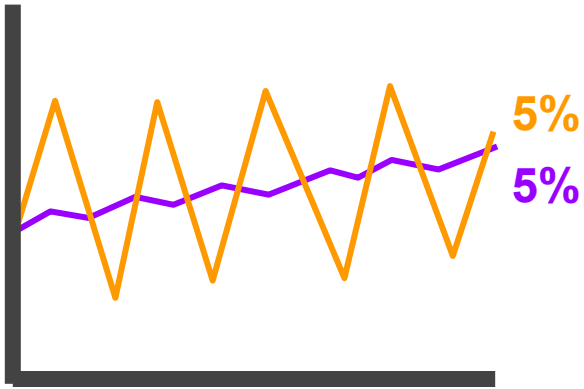
- But there is still one critical statistic we haven't discussed yet, the Sharpe Ratio!
- Before we explain what it is, let's give you an intuitive understanding of why it is necessary!
- Let's imagine 3 portfolio comparisons...

Python for Finance

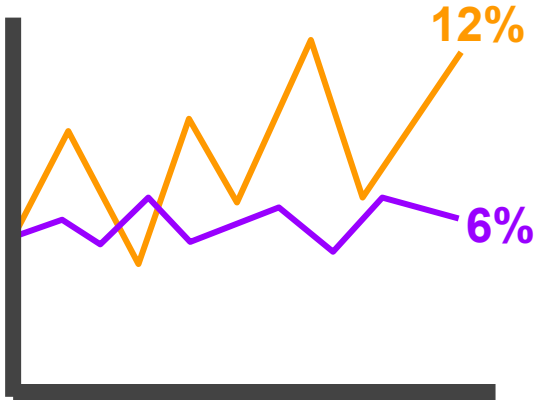
For each case, which portfolio is better?



Case 1



Case 2



Case 3

Python for Finance

- The Sharpe Ratio is a measure for calculating risk-adjusted return, and this ratio has become the industry standard for such calculations.
- It was developed by Nobel laureate William F. Sharpe.

Python for Finance

- The formula for the Sharpe Ratio:

$$S = \left(\frac{R_p - R_f}{\sigma_p} \right)$$

- Rp is Expected Portfolio return
- Rf is Risk-Free Return
- Sigma p is Portfolio Standard Deviation

Python for Finance

- What is risk free return?
- The return you would receive if you put your money in an investment such as a bank savings account, LIBOR, Treasury Bonds that are essentially “risk-free”.

Python for Finance

- Currently in the United States (early 2017), these returns are very close to 0%, so its just easier to approximate Rf as 0.
- Keep in mind though, the Federal Reserve may continue to raise interest rates in the future, effecting this result!

Python for Finance

- For now, assuming Rf is 0, we get:
  - SR = Mean Return / Std. Dev.
- Keep in mind that Sharpe originally thought of this as a yearly metric (as in mean yearly return vs. mean daily return)
- This is easy to fix for us though!

Python for Finance

- The annualized Sharpe Ratio can be obtained by multiplying against a K-Factor based off your Sampling Rate:
  - Daily: K= sqrt(252)
  - Weekly: K=sqrt(52)
  - Monthly: K=sqrt(12)
- So that you just calculate ASR = K * SR

Python for Finance

- Let’s see how to do all of this with Python!

Portfolio Allocation & Sharpe Ratio

Python for Finance

- Let’s jump to the notebook and code it out!
- All the stock information is available as csv files in case Quandl or pandas-datareader are unavailable for you.

Portfolio Allocation & Sharpe Ratio Code Along Part Two

PIERIAN DATA

Portfolio Optimization

Python for Finance

- Now that we have a nice metric to evaluate Portfolio Allocations against each other, how can we optimize the portfolio holdings?
- We could just guess and check a bunch of random allocations and see which one has the best Sharpe Ratio!

Python for Finance

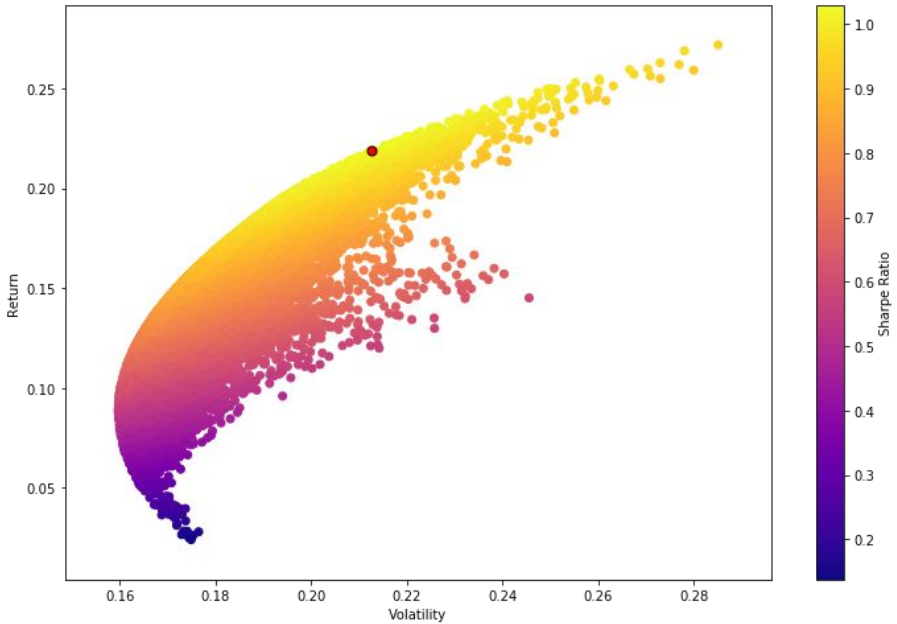
- This is known as Monte Carlo Simulation.
- We randomly assign a weight to each security in our portfolio, then calculate its mean daily return and std. dev. of daily return.

Python for Finance

- This allows us to calculate the Sharpe Ratio for thousands of randomly selected allocations.
- We can then plot the allocations on a chart showing return vs. volatility, colored by the Sharpe Ratio.

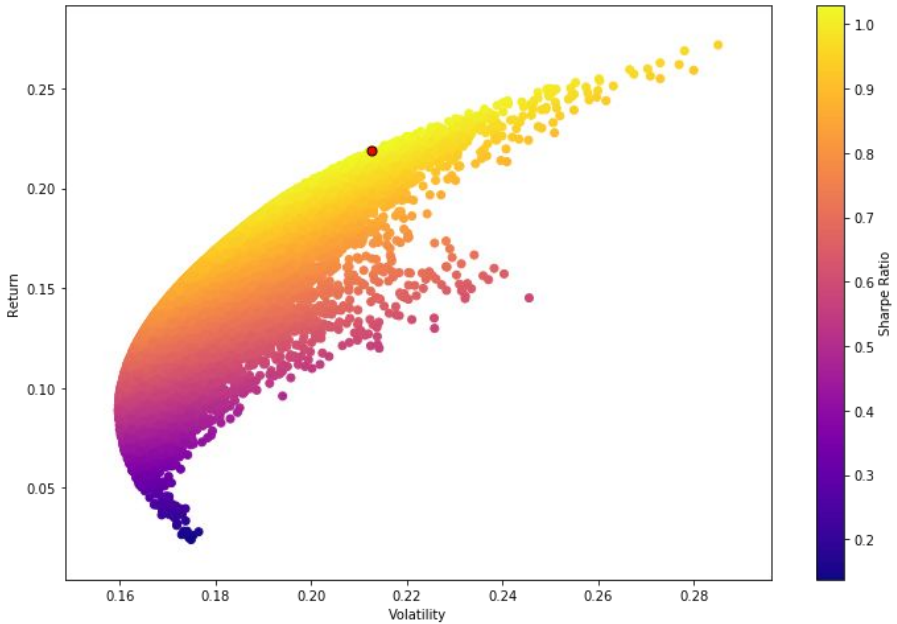
Python for Finance

- We will create something like this:



Python for Finance

- Red dot indicates the highest Sharpe Ratio



Python for Finance

- However guessing and checking is not very efficient, instead we can use math to figure out the optimal Sharpe Ratio for any given portfolio.
- To understand optimization algorithms, we need to understand minimization!

Python for Finance

- Before we discuss optimization directly, let's discuss minimization, which is a very similar concept!
- Let's say we are given simple equations:
  - $y = x^2$
  - $y = (2 - x)^2$

Python for Finance

- What value of x will minimize y?
  - $y = x^2$ ----> $x = 0$
  - $y = (2 - x)^2$ ----> $x = 2$
- This idea of using a minimizer will allow us to build an optimizer.
- Luckily for more complex equations, SciPy can do the heavy math for us!

Python for Finance

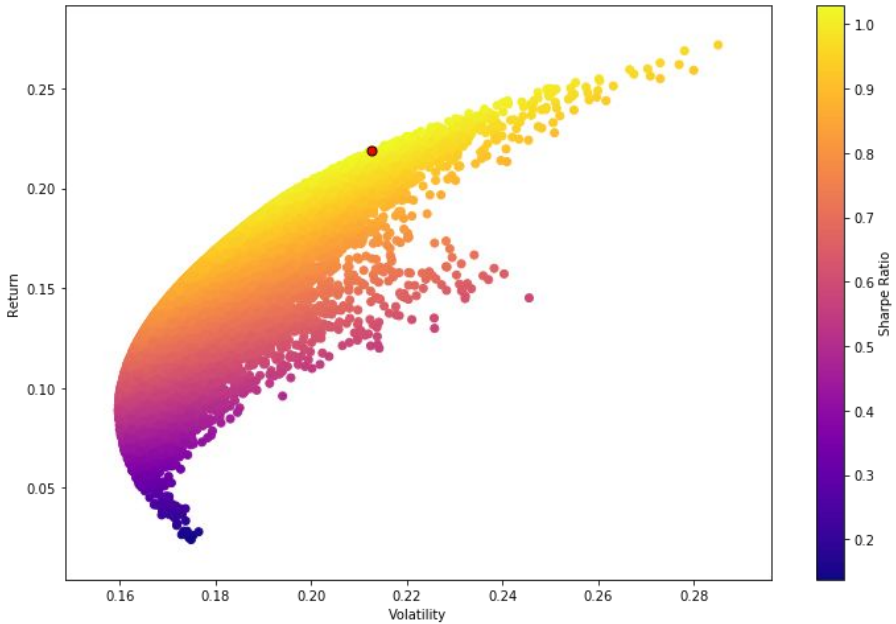
- Referring back to our Sharpe Ratio, we want to actually maximize the Sharpe Ratio, meaning we can create an optimizer that will attempt to minimize the negative Sharpe Ratio!

Python for Finance

- We will use SciPy's built-in optimization algorithms to calculate the optimal weight allocation for our portfolio (optimized by Sharpe Ratio).
- Let's go back to the return vs. volatility plot again

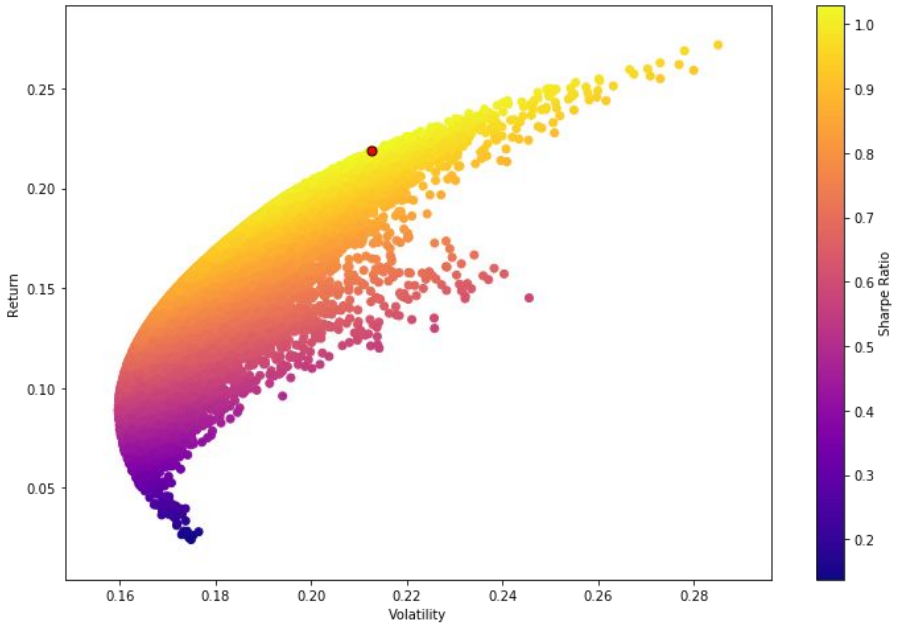
Python for Finance

- Notice the shape of the plot.



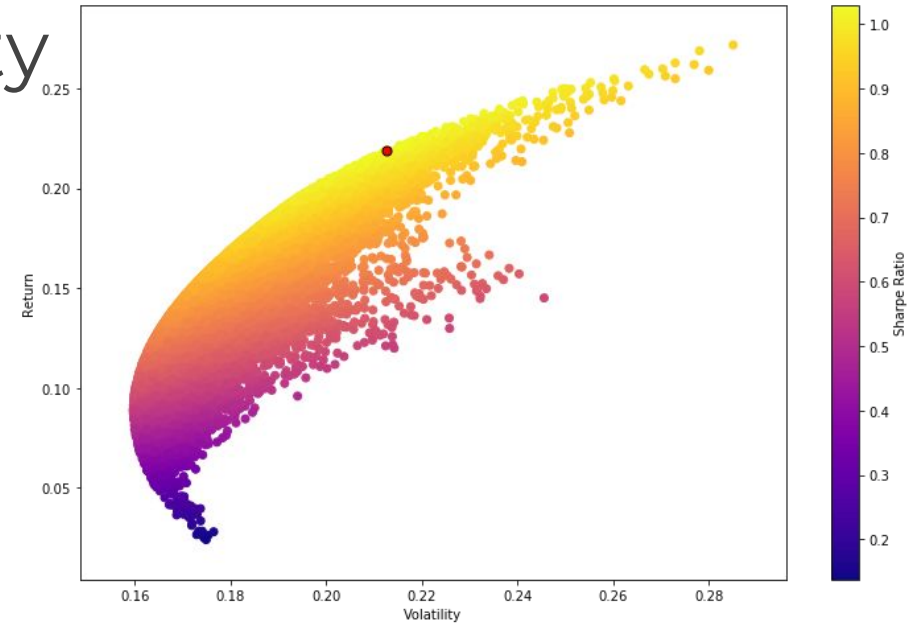
Python for Finance

- It seems to create a border.



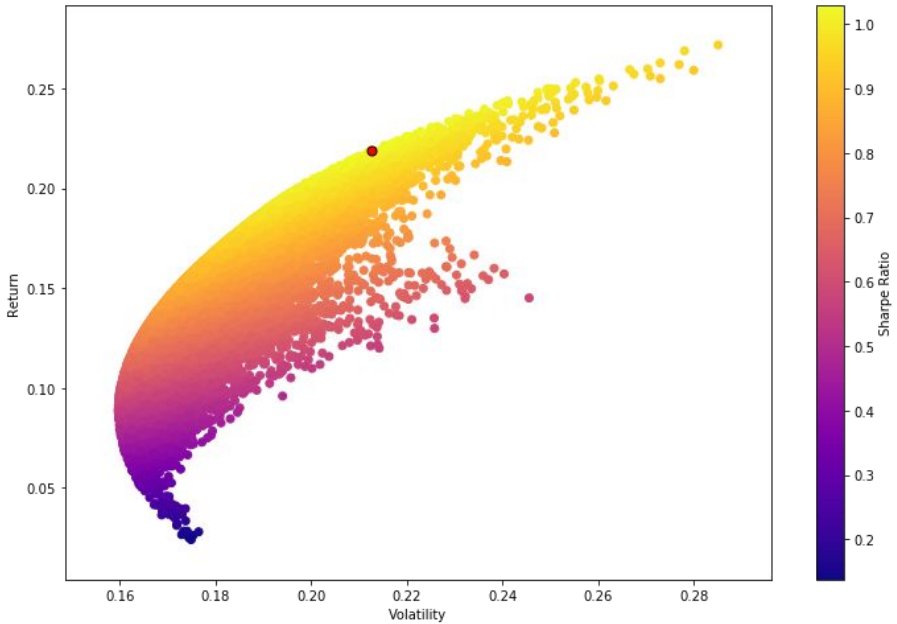
Python for Finance

- This indicates the highest return for volatility



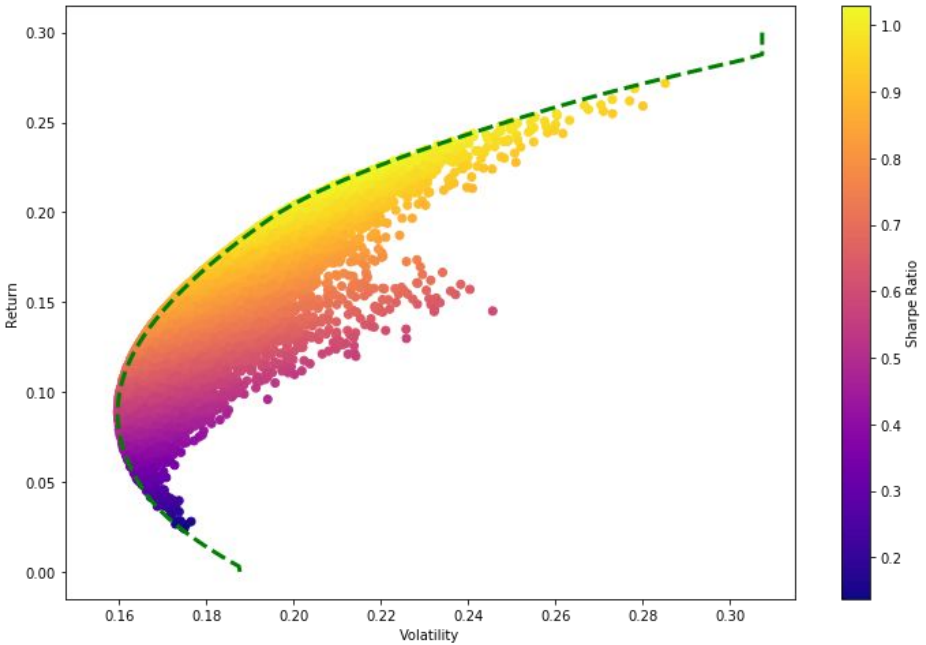
Python for Finance

- Anything below the border is a bad choice.



Python for Finance

- This border is known as the efficient frontier.



Python for Finance

- Let’s walk through using Python to calculate all of these metrics and topics!

Portfolio Optimization Code Along

PIERIAN DATA

Financial Market Concepts

Python for Finance

- In this series of lectures we will go over a few key financial topics
  - Types of Funds
  - Order Books
  - Latency Arbitrage (HFT)
  - Short Selling

Let’s get started!

Types of Funds

Python for Finance

- There are 3 major fund types:
  - ETF - Exchange Traded Funds
  - Mutual Funds
  - Hedge Funds
- These funds vary by fees, transparency, regulation, and more.
- Let’s discuss their differences.

Python for Finance

- ETFs are exchange traded funds that are constituted of a basket of funds, bonds, commodities etc...
- Their holdings are completely public and transparent and individuals can buy and trade the marketable security

Python for Finance

- Typically people investing in ETFs are more interested in a diversified portfolio and want to keep their investment in an ETF for a longer period of time.
- One of the most common ETFs is the Spider (SPY) which tracks the S&P500

Python for Finance

- A mutual fund is an investment vehicle made up of a pool of funds collected from many investors for the purpose of investing in securities such as stocks, bonds, money market instruments and similar assets.

Python for Finance

- Mutual funds are operated by money managers, who invest the fund's capital and attempt to produce capital gains and income for the fund's investors.

Python for Finance

- A mutual fund's portfolio is structured and maintained to match the investment objectives stated in its prospectus.
- Mutual Funds disclose their holdings typically once a quarter, although this can vary by fund.

Python for Finance

- Hedge funds are alternative investments using pooled funds that employ numerous different strategies to earn active return, or alpha, for their investors.

Python for Finance

- Hedge funds may be aggressively managed or make use of derivatives and leverage in both domestic and international markets with the goal of generating high returns (either in an absolute sense or over a specified market benchmark).

Python for Finance

- It is important to note that hedge funds are generally only accessible to accredited investors as they require less SEC regulations than other funds.

Python for Finance

- One aspect that has set the hedge fund industry apart is the fact that hedge funds face less regulation than mutual funds and other investment vehicles.
- Much of what we do in this course, mimics the way a hedge fund would operate.

Python for Finance

- We will try to use python and math to attempt to beat some benchmark.
- We also won't need to disclose our strategy.
- Let's discuss the fee structure for these three fund types.

Python for Finance

- What are the fees associated with each?
- ETF Funds
  - Expense Ratio - 0.01% - 1%
- Mutual Funds
  - Expense Ratio - 0.5% - 3%
- Hedge Funds
  - 2% of Fund. 20% of profits.

Python for Finance

- What is the liquidity with each?
- ETF Funds
  - Buy/Sell just like a stock.
- Mutual Funds
  - Buy/Sell at end of day through broker.
- Hedge Funds
  - Depends on agreement

Python for Finance

- That is all we really need to know about the basics of the different fund types!
- Up next, let’s discuss what actually happens when we buy or sell a stock on an exchange!

Order Books

Python for Finance

- So you've decided to buy/sell a stock!
- You log on to your brokerage account (Robinhood, E*Trade, Ameritrade, etc...).
- You click on the stock you want to buy/sell and then you either pay or receive money.
- But what actually happens when you click?
- Let's go through the process!

Python for Finance

- Making an Order includes the following:
  - Buy or Sell
  - Symbol
  - Number of Shares
  - LIMIT or MARKET
  - Price (only needed for a LIMIT order)

Python for Finance

- Example orders:
  - BUY, AAPL, 200, MARKET
  - SELL, TSLA, 400, MARKET
  - BUY, AMD, 2000, LIMIT, 13.95
  - SELL, NVDA, 150,LIMIT,160.99

Python for Finance

- Once you've sent an order (usually to your broker), it goes to an exchange (for a larger order it can go to multiple exchanges, but more on that later)
- Once an exchange receives your order, it goes into an order book.

Python for Finance

- Let’s build out an order book!

NYSE BAC ORDER BOOK			
BUY ORDERS		**SELL ORDERS**	
SHARES	BID	ASK	SHARES

Python for Finance

- (BUY,BAC,200,LIMIT,199.95)

NYSE BAC ORDER BOOK			
BUY ORDERS		**SELL ORDERS**	
SHARES	BID	ASK	SHARES
200	199.95		

Python for Finance

- (SELL,BAC,100,LIMIT,199.90)

NYSE BAC ORDER BOOK			
BUY ORDERS		**SELL ORDERS**	
SHARES	BID	ASK	SHARES
200	199.95	199.90	100

Python for Finance

- (SELL,BAC,50,LIMIT,199.91)

NYSE BAC ORDER BOOK			
BUY ORDERS		**SELL ORDERS**	
SHARES	BID	ASK	SHARES
200	199.95	199.90	100
		199.91	50

Python for Finance

- (SELL,BAC,50,LIMIT,199.92)

NYSE BAC ORDER BOOK			
BUY ORDERS		**SELL ORDERS**	
SHARES	BID	ASK	SHARES
200	199.95	199.90	100
		199.91	50
		199.92	50

Python for Finance

- Example of an order book:

NASDAQ BOOKVIEWER

Export Help

GOOG

GET STOCK

Last Match

576.53

10:58:01.135

Current stock: GOOG

Today's Activity

Orders 57,513

Volume 285,290

Filter: »

Aggregate By: Price MPID

BUY ORDERS				SELL ORDERS			
TIME	MPID	SHARES	BID	ASK	SHARES	MPID	TIME
10:58:01.021	NSDQ	300	576.42	576.52	80	NSDQ	10:58:01.367
9:30:13.227	CDRG	75	576.40	576.58	100	NSDQ	10:58:01.367
10:45:35.954	UBSS	500	576.40	576.67	100	NSDQ	10:58:00.289
10:58:00.901	NSDQ	10	576.37	576.73	100	TMBR	10:58:00.223
10:58:00.899	NSDQ	10	576.36	576.74	100	NSDQ	10:58:01.369
10:58:00.225	NSDQ	8	576.26	576.80	100	NSDQ	10:58:01.213
9:30:13.227	CDRG	2	576.25	576.85	100	NSDQ	10:58:00.225
9:37:01.074	AUTO	30	576.20	576.85	100	NSDQ	10:58:01.138
10:56:18.648	NITE	100	576.20	576.86	100	NSDQ	10:57:40.157
10:58:00.400	NSDQ	22	576.12	576.87	22	NSDQ	10:58:01.367
10:58:00.175	NSDQ	22	576.10	576.88	8	NSDQ	10:58:01.368
10:56:34.266	NSDQ	50	576.08	576.89	22	NSDQ	10:58:01.136
10:58:01.136	NSDQ	22	576.08	576.90	10	NSDQ	10:58:01.369
10:58:01.136	NSDQ	22	576.06	576.91	10	NSDQ	10:58:00.623
10:57:13.883	NSDQ	100	576.05	576.91	22	NSDQ	10:58:01.137

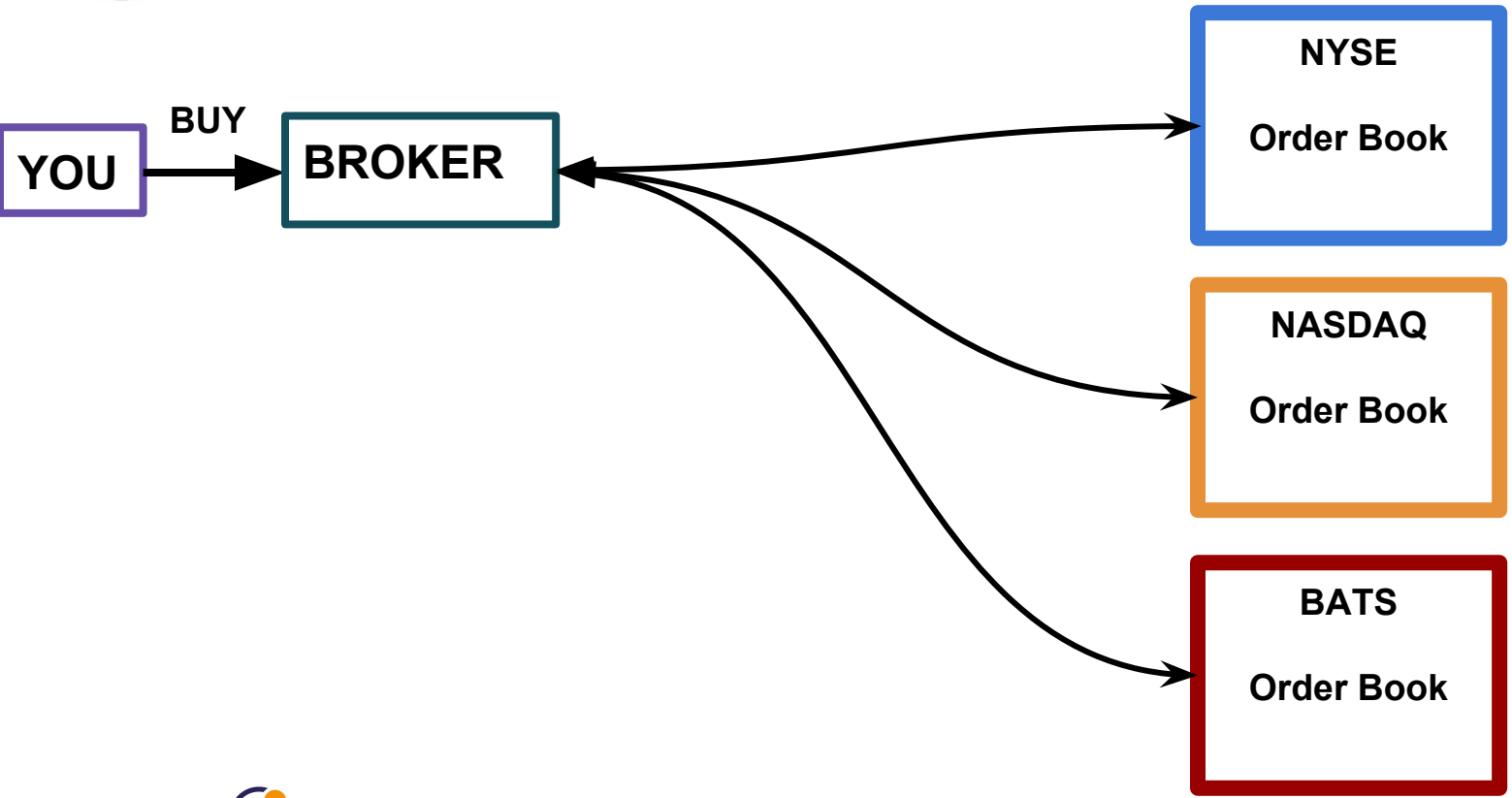
(4512 More) (3357 More) Show More

As of 10:58:01.506 Powered by NASDAQ TotalView

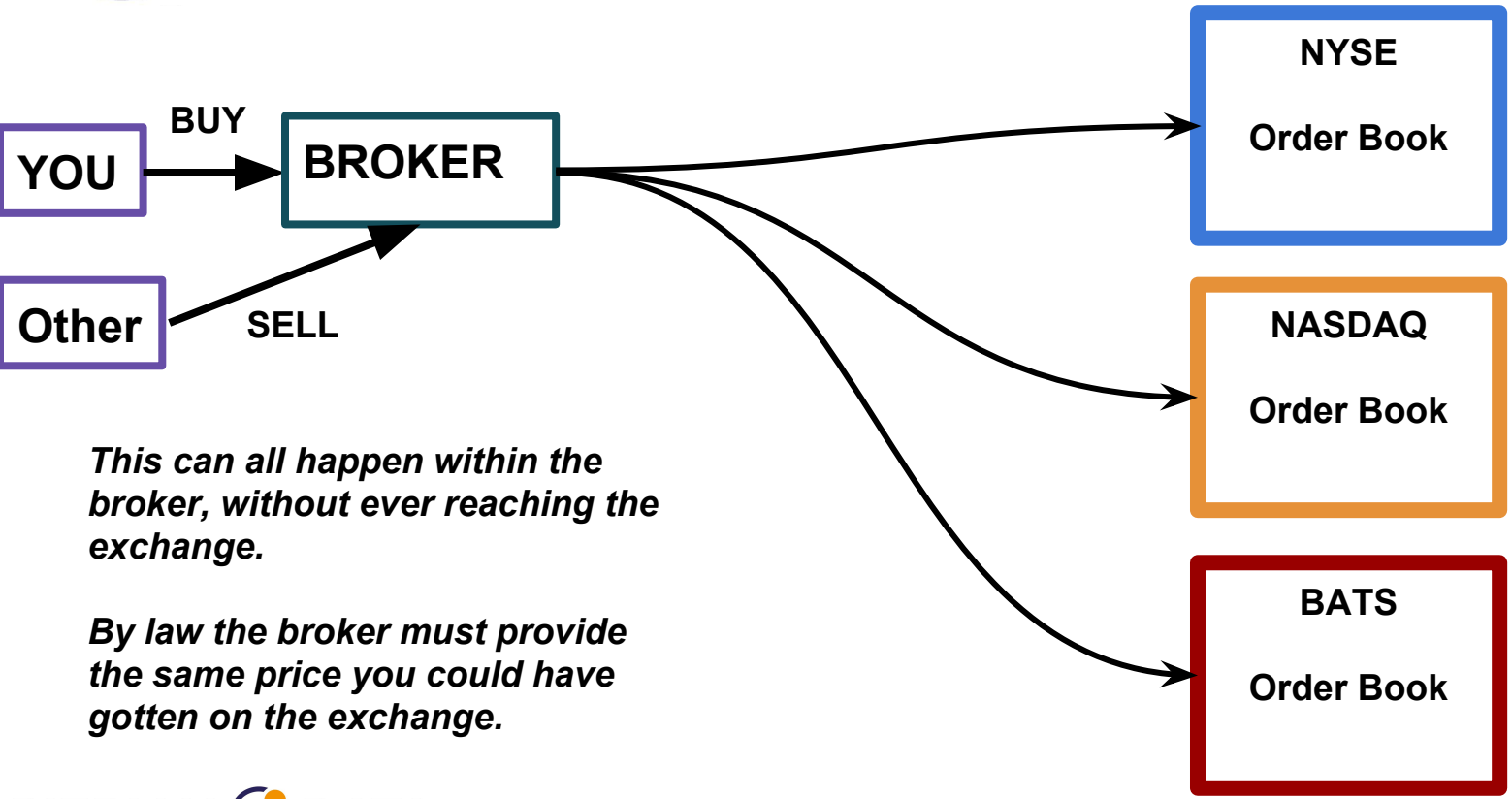
Python for Finance

- So how does an order actually get to the exchange?
- Let’s walk through the steps, we’ll also explain how certain HFT (High Frequency Trading) firms can attempt Latency Arbitrage!

Scenario #1 : Simple Buy



Scenario #2 : Broker Buy/Sell

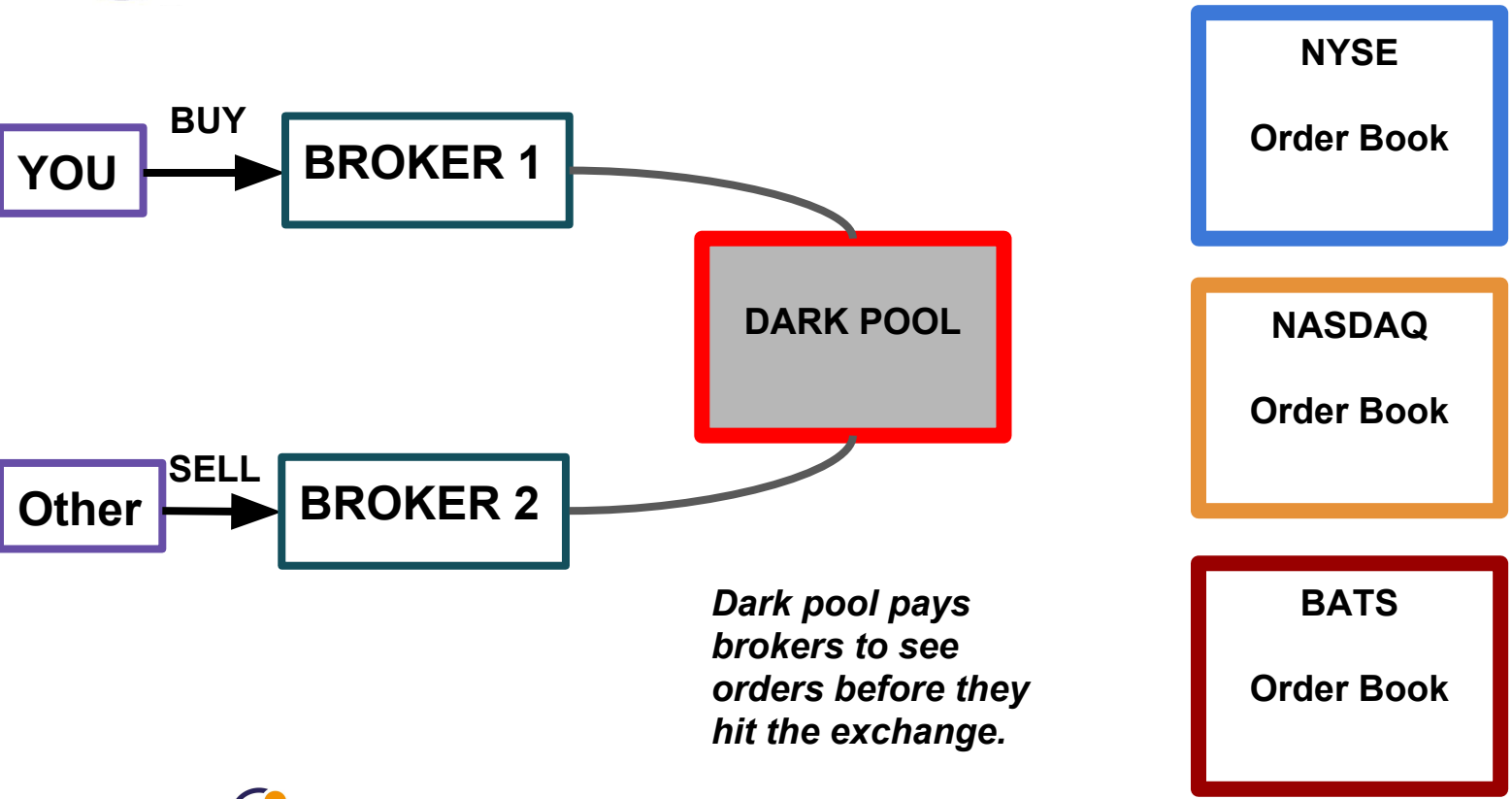


This can all happen within the broker, without ever reaching the exchange.

By law the broker must provide the same price you could have gotten on the exchange.

PIERIAN DATA

Scenario #3 : Dark Pool



Dark pool pays brokers to see orders before they hit the exchange.

Python for Finance

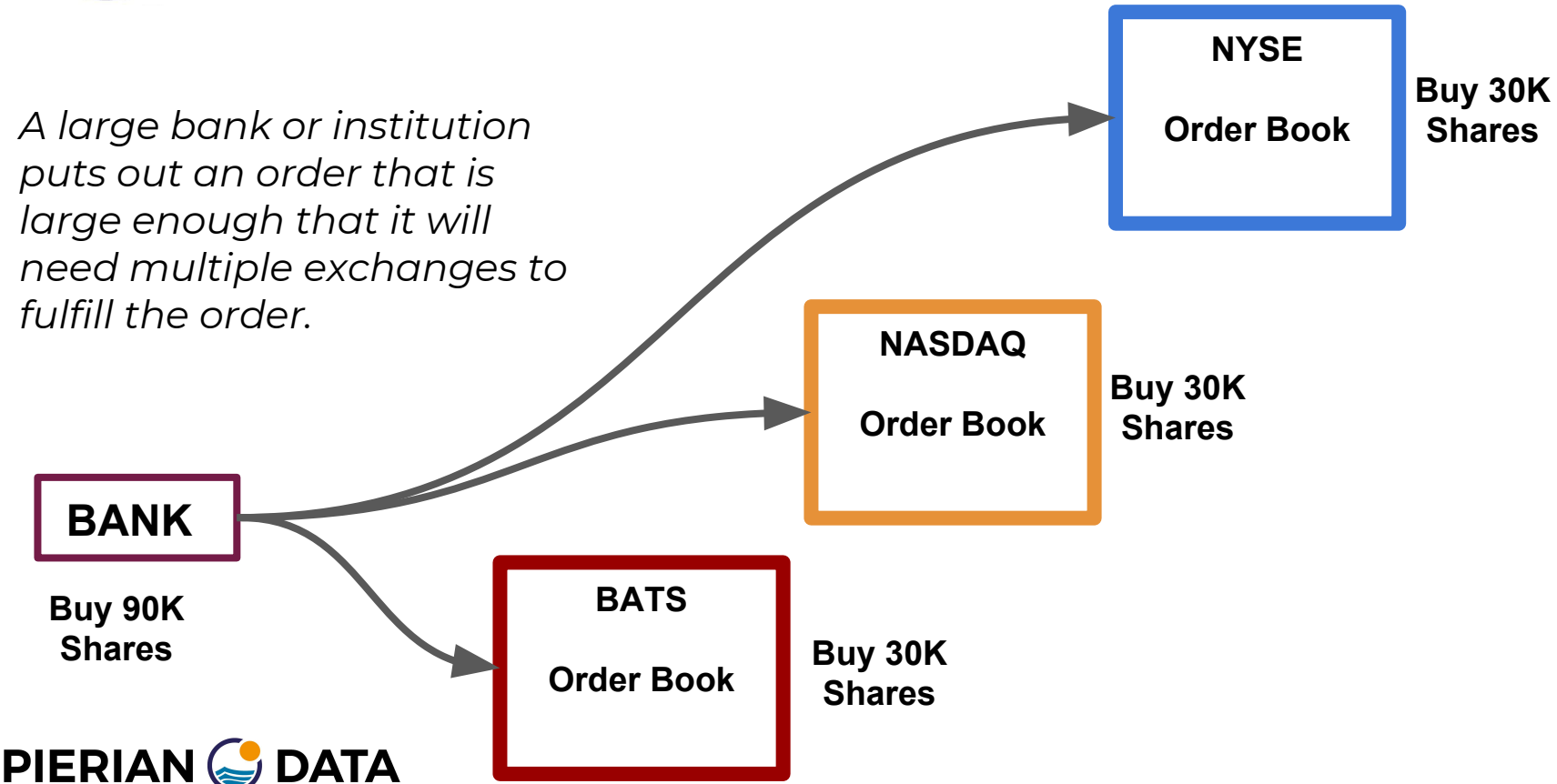
- You may have heard of the term “High Frequency Trading”.
- HFT firms take advantage of latency differences due to geographical distances.
- These times are on the order of microseconds.

Python for Finance

- Discussion about HFTs were popularized by Michael Lewis' book "Flash Boys", about Brad Katsuyama and the IEX.
- Let's briefly explain the basic idea of HFT, but keep in mind this is an area that changes extremely fast due to technology!

Scenario #4 : HFT

A large bank or institution puts out an order that is large enough that it will need multiple exchanges to fulfill the order.

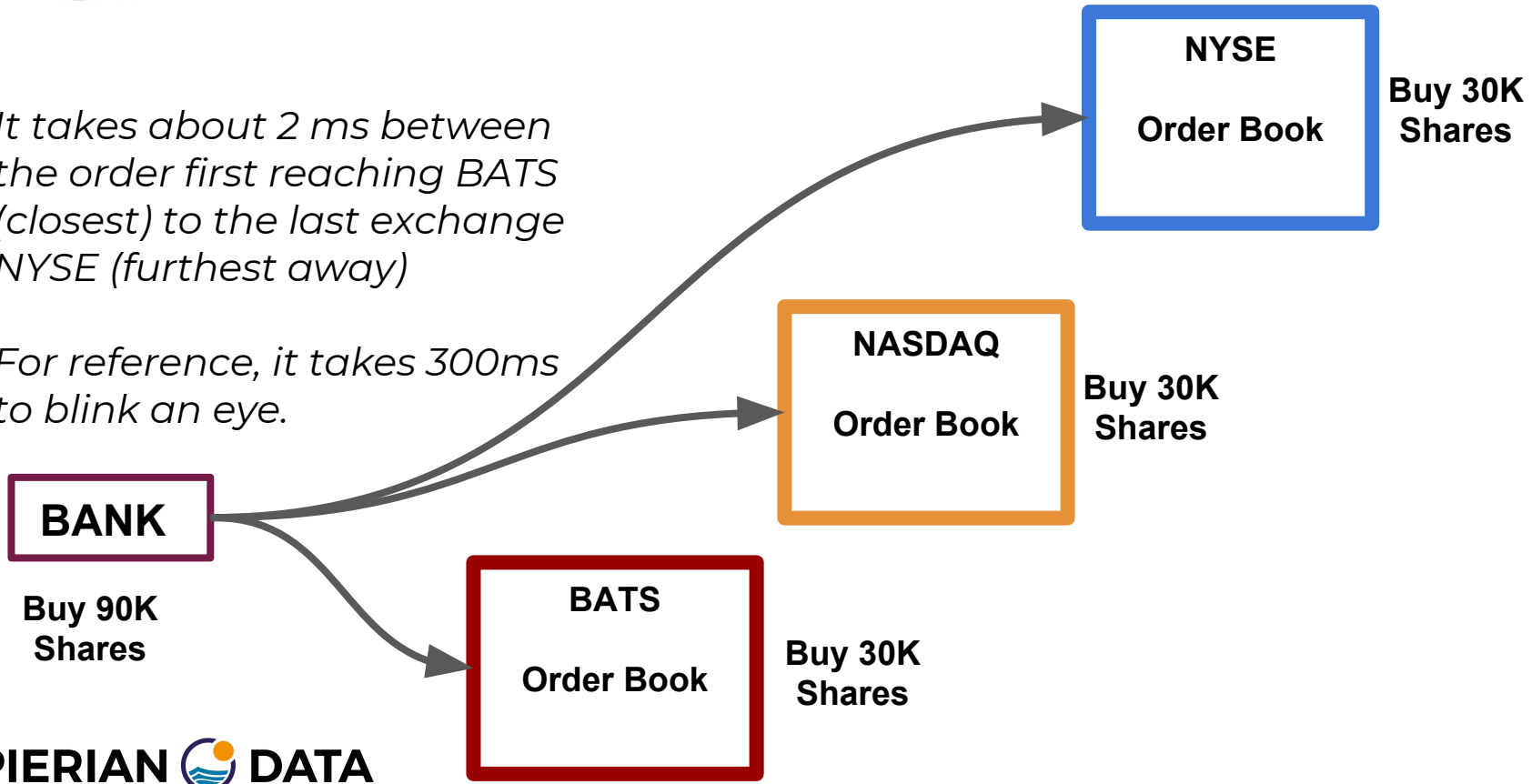


PIERIAN DATA

Scenario #4 : HFT

It takes about 2 ms between the order first reaching BATS (closest) to the last exchange NYSE (furthest away)

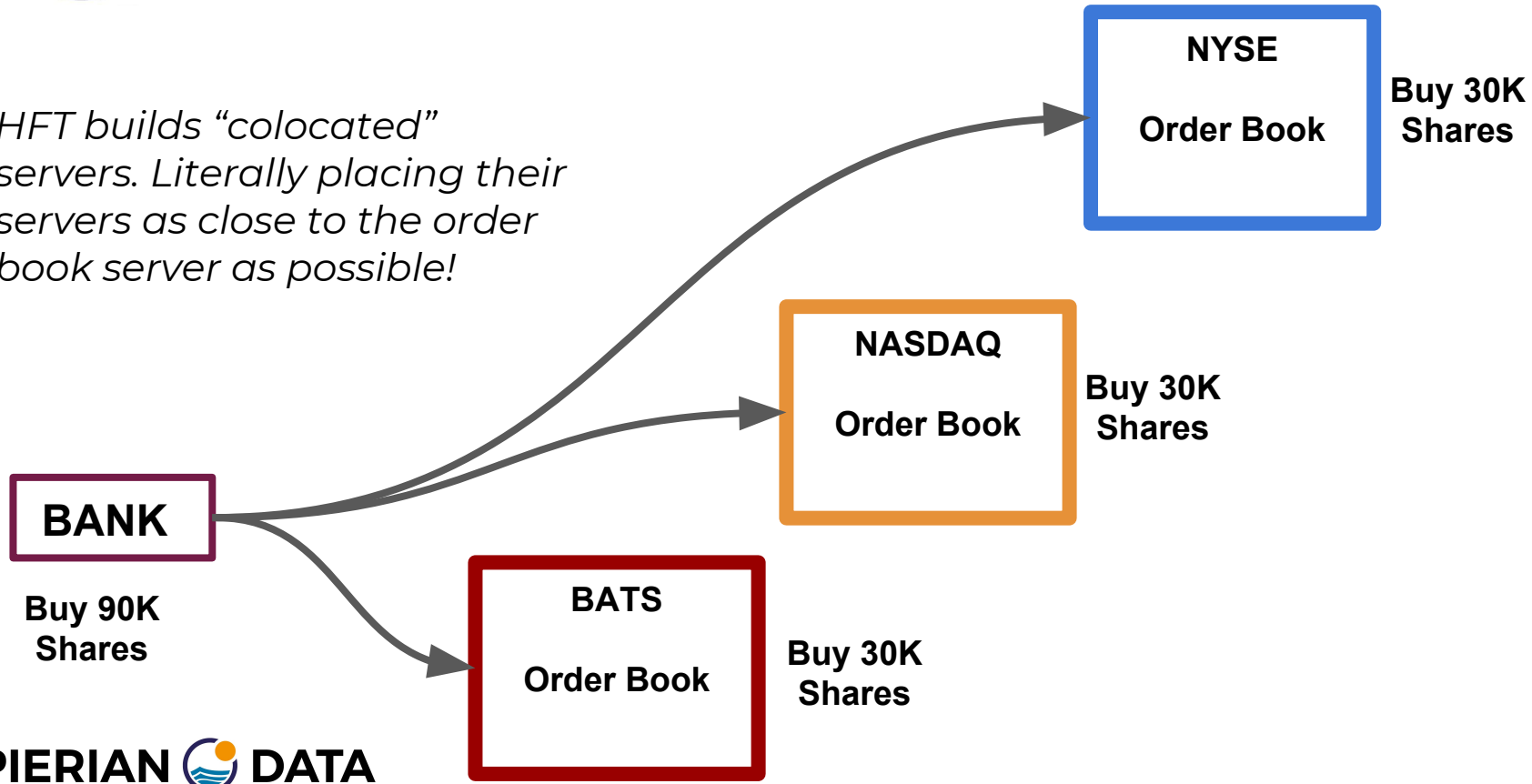
For reference, it takes 300ms to blink an eye.



PIERIAN DATA

Scenario #4 : HFT

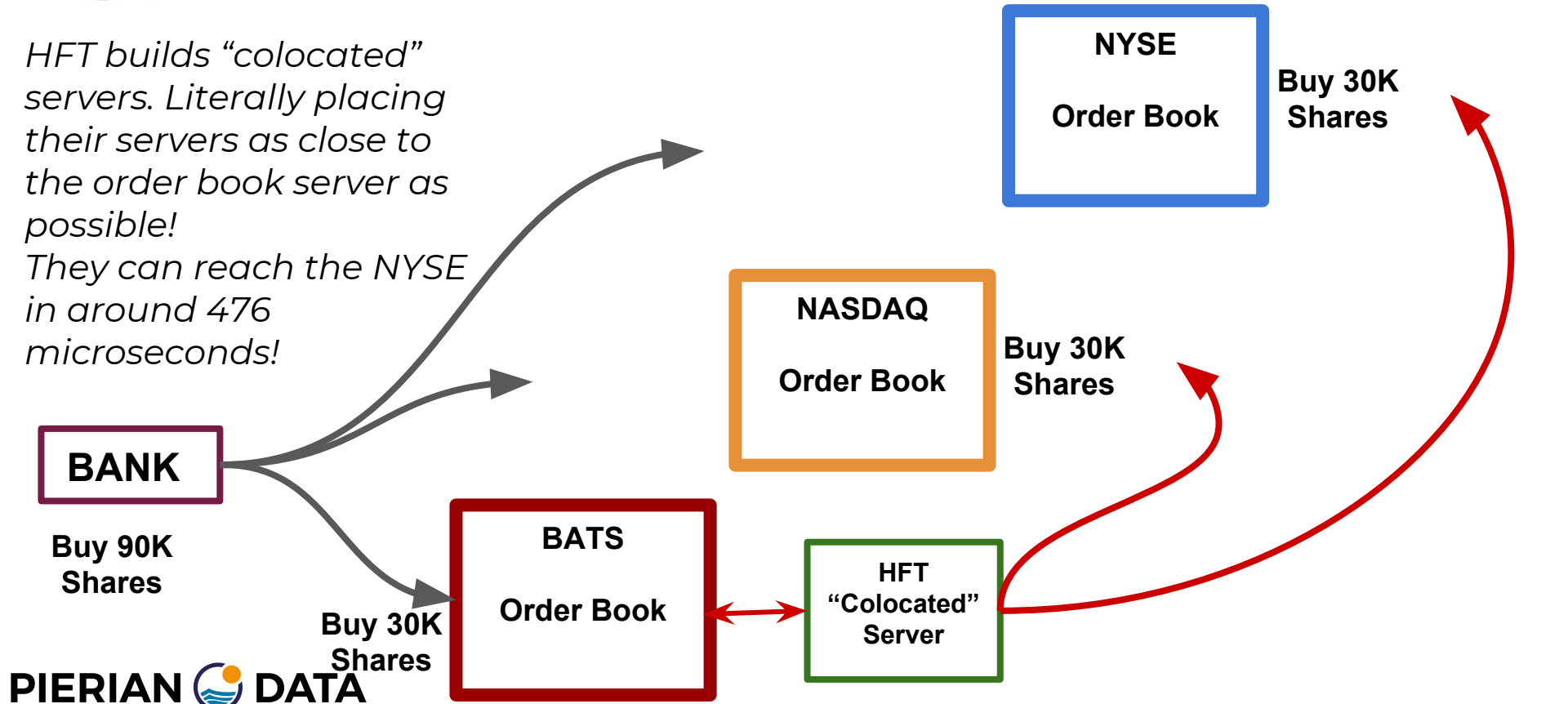
HFT builds "colocated" servers. Literally placing their servers as close to the order book server as possible!



PIERIAN DATA

Scenario #4 : HFT

HFT builds "colocated" servers. Literally placing their servers as close to the order book server as possible!
They can reach the NYSE in around 476 microseconds!



PIERIAN DATA

Python for Finance

- This was a simplified overview, check out the resource links if this is a topic that interests you.
- HFT in general is not really relevant for our scale or approach to trading.

Short Selling

Python for Finance

- Let’s quickly discuss short-selling!
- Short-selling allows you to profit if a stock drops in price, however it comes with a great risk, due to the fact that there can be no ceiling to the amount of money you could possibly lose!

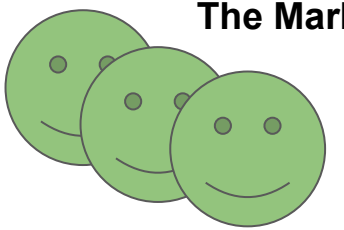
Python for Finance

You

10 Shares Of GOOG

Step 0: GOOG is currently valued at $500. You think it will drop in the future!

RED has 10 shares of GOOG.



The Market

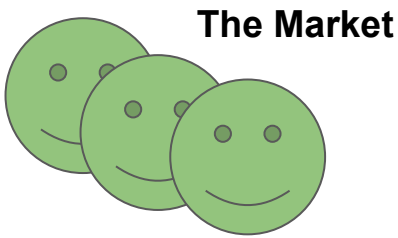
Current Price of GOOG
$500

PIERIAN DATA

Python for Finance

Step 1: You borrow RED's shares of GOOG.

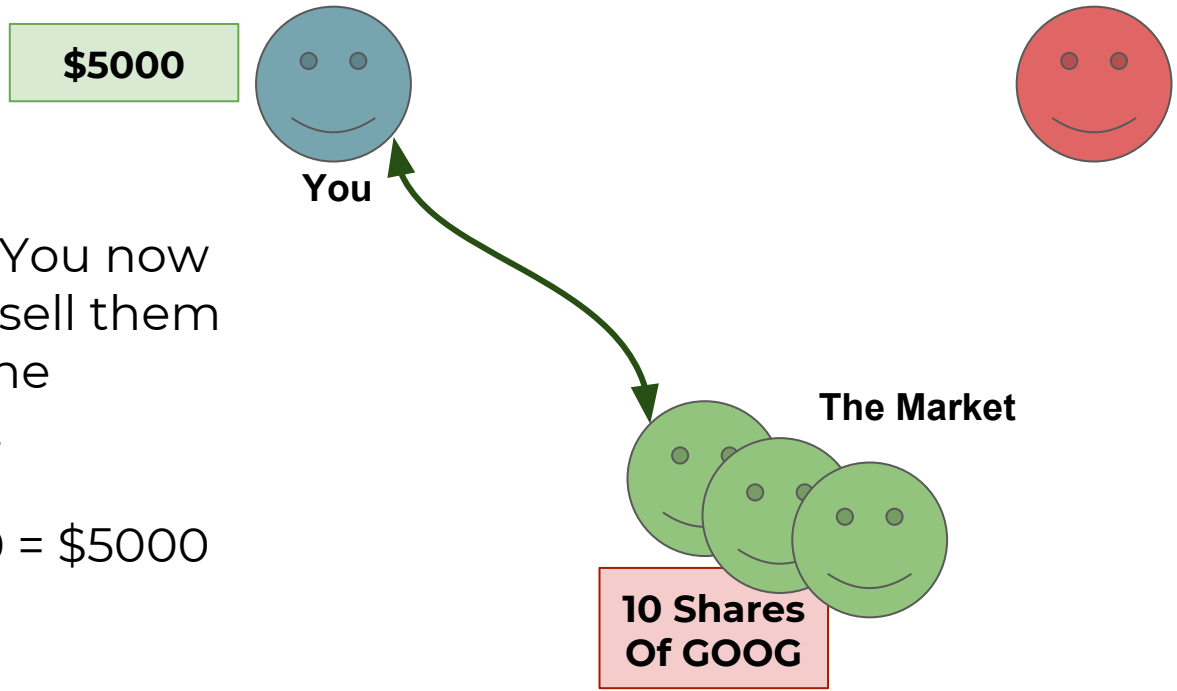
Promise to return them at a future date!



Current Price of GOOG
$500

PIERIAN DATA

Python for Finance

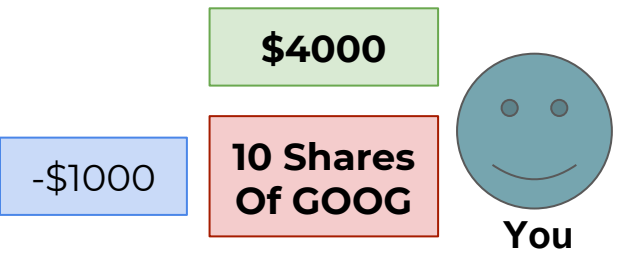


Step 2: You now go and sell them out in the market.

$500*10 = $5000

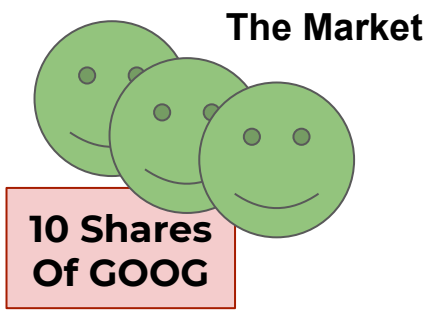
Current Price of GOOG
$500

Python for Finance



Step 3: The price of google has now dropped to $100.

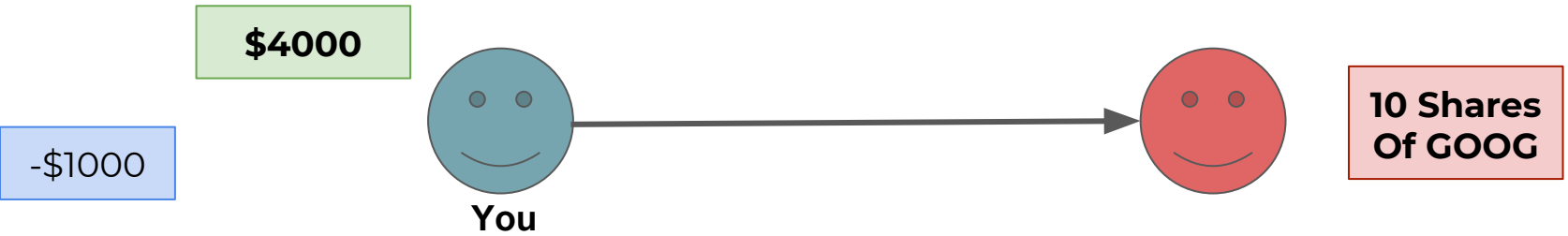
You go out and buy 10 Shares for $1000.



Current Price of GOOG
$100

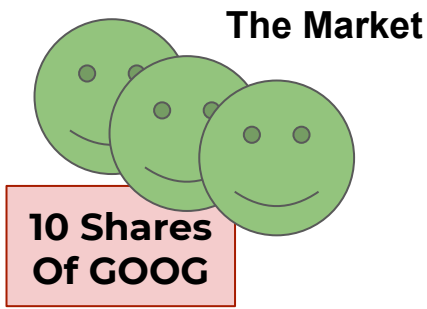
PIERIAN DATA

Python for Finance



Step 4: Now you can give back the 10 Shares you owe back to RED.

And you've made $4000!



Current Price of GOOG
$100

PIERIAN DATA

Python for Finance

- You should note, that if the price goes up instead of down, you will lose money buying back the stock to give it back to RED!
- Also all these transactions would happen through a broker, not directly.

Python for Finance

- Let’s now move on to discussing CAPM--Capital Asset Pricing Model!

CAPM

Python for Finance

- The Capital Assets Pricing Model (CAPM) is one of the most fundamental topics in investing!
- It is a model that helps describe risk and separating market return versus your portfolio return.

Python for Finance

- Remember that a portfolio is a set of weighted securities. We can define the returns as the following:

$$r_p(t) = \sum_{i}^{n} w_i r_i(t)$$

Python for Finance

- We can also imagine the entire market as a portfolio, for example the S&P500
- How do we get weights for each company?
- Using th)

$$w_i = \frac{MarketCap_i}{\sum_j^n MarketCap_j}$$

Python for Finance

- The CAPM equation is states the following:

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- There are two terms here, the Beta term and the Alpha term.
- Note how this models a simple regression line: y = mx +b

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- The CAPM equation describes the return of some individual stock i.
- Comprised of two terms, Beta and Alpha.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- The Beta term implies that the return of a stock is equivalent to the return of the market multiplied by this Beta factor plus some residual alpha term.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- You could imagine that is Beta=1 than this stock moves in line with the market.
- If Beta=2 , then this stock moves up and down twice as much as the general market.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- Since the return for a stock “i” is not going to match exactly with the Beta term, we add some alpha term.
- CAPM states that you expect this alpha term to be zero.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- With CAPM stating that alpha should be expected to be zero, it basically implies that you can not beat the general market!
- CAPM also says that alpha is random and can not be predicted!

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- This basically creates the argument of passive versus active investment.
- Active investors (like us) believe we can predict alpha to some degree.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- Remember that we don't need to be 100% correct on alpha, we should just be able to predict it correctly more than 50% of the time for our portfolio.

$$r_i(t) = \beta_i r_m(t) + \alpha_i(t)$$

Python for Finance

- Since we believe we can have a good idea for values of alpha, we can break it up per security:

$$r_p(t) = \beta_p r_m(t) + \sum_i^n w_i \alpha_i(t)$$

Python for Finance

- Since we believe we can have a good idea for values of alpha, we can break it up per security:

$$r_p(t) = \beta_p r_m(t) + \sum_i^n w_i \alpha_i(t)$$

Python for Finance

- We could even continue to break up Beta into various Betas per sector of the market (Energy, Manufacturing, Tech, Finance, etc)

$$r_p(t) = \beta_p r_m(t) + \sum_{i}^{n} w_i \alpha_i(t)$$

Python for Finance

- The main point here we want to get across is that there will be some relationship between our portfolio return and the overall market return.
- This is the Beta term.

Python for Finance

- CAPM says that the alpha term can not be predicted, however, we will fundamentally disagree with that model of thinking.
- We will see if we can create strategies that allow us to have significant alpha terms, meaning our strategy beats the market.

Python for Finance

- These two terms (beta and alpha) will come up again once we learn how to use the Quantopian platform.
- For now let's go ahead and quickly use Python to create an example of CAPM in the next lecture!

CAPM Code Along

Stock Splits & Dividends

Python for Finance

- Let’s quickly discuss why companies split their stock and dividends.
- Stock splits usually occur if the price for an individual stock becomes unreasonably high, so companies essentially just create a ratio split (e.g. 2:1 , 3:1 , 4:1).

Python for Finance

- Because stock splits could cause errors for algorithms looking for price changes, we use the Adj. Close , which adjusts the historical prices to match up and take into account the stock splits.

Python for Finance

- Often stocks also pay dividends, for each unit of stock, each shareholder receives some payout.
- This causes the price to jump before the dividend announcement, and then drop after the dividend is paid out.

Python for Finance

- The adjusted close price also takes this into account.
- It is important to always use adjusted close prices, otherwise you may be affected by stock splits and dividends in your pricing strategies.

Python for Finance

- One last thing to take into account is “Survivorship Bias”.
- If we take a look at the S&P500 benchmark today, it is a different set of companies than say in the year 1999 (the peak of the dot-com bubble).

Python for Finance

- After the dot-com crash many companies that were in the S&P500 no longer existed or became to small to be part of the 500.
- Meaning the time you pick S&P500 could possibly matter!

Python for Finance

- If we were creating strategies in the early 2000s, we may have heavily used the S&P500, but if we didn't take into account the earlier make-up of the S&P500 we would end up having a bias in our results!

Python for Finance

- Many of the strategies we provide later on won’t be affected by this, but keep it in mind in case you end up using a benchmark as part of your strategy and you backtest to 2007 (financial crisis).

Python for Finance

- You can buy Survivor Bias Free data from a variety of sources (including Quandl), keep in mind it is usually not free (although it isn’t very expensive either, check Quandl premium or other sources for the latest pricing).

Efficient Market Hypothesis

Python for Finance

- EMH is an investment theory that states it is impossible to "beat the market" because stock market efficiency causes existing share prices to always incorporate and reflect all relevant information.

Python for Finance

- According to the EMH, stocks always trade at their fair value on stock exchanges, making it impossible for investors to either purchase undervalued stocks or sell stocks for inflated prices.

Python for Finance

- As such, it should be impossible to outperform the overall market through expert stock selection or market timing, and the only way an investor can possibly obtain higher returns is by purchasing riskier investments.

Python for Finance

- So is EMH true?
- If the EMH is true, then a lot of what we will learn in this course doesn't have a point to it.
- It would also imply that many hedge funds are just extremely lucky.

Python for Finance

- The success of different strategies and hedge funds implies the the strongest interpretation of EMH is probably not true.
- There are also events that have shown the market to be overvalued at certain points in history (financial crisis, dot com bubble, etc)

Python for Finance

- While some aspects of EMH certainly are true, especially as information is more widely acceptable, we will continue on with the assumption that the market is not 100% efficient!