Homework 3

Planned Work:

- Prepare the data for analysis (collect data and format it(I.E. websocket JSON object -> Price or Symbol))
- Call upon the accessed data from the API (likely in a separate thread)
- Generate functions which determine critical information that is not supported in the API such as SMA (Simple Moving Average) at specific time frames such as 1M, 5M, 10M, 30M, and 1H
- Introduce a highestPrice and lowestPrice variable
- Rearrange to accurately represent the structure of the program (Classes will be created and organized appropriately)
- Produce 1M, 5M, 10M, 30M, and 1H Simple Moving Averages
- Focus on learning the best possible investment strategy.

Accomplished Work:

- Data has been prepared (websocket JSON is iterated through to collect price and symbol)
- A new thread is created and closed roughly every 15 seconds of the programs execution. (possibly leave it open for deadline? I was struggling having it open to collect data but also allowing executive function outside of the newly opened thread)
- calcAverage function is utilized to calculate average price, average low price, and average high price every 15 seconds. This function is called every 15 seconds and continuously tracks low, average, and high price action from initial execution until termination of the program (which can include 1M, 5M, 10M, 30M, and 1H moving averages)
- highestPrice and lowestPrice variables were introduced in calcAverage function and are implemented into a high and low list which is appended every 15 seconds with the call of calcAverage
- Class Trade was introduced and included beneath it is class Stock and class Crypto. I could have done more work in the class implementation but I ultimately fell short on time in this regard. It is my plan to introduce a JSON array object with all the data pulled in the 15 second thread. If data is a stock, implement a stock class and maybe have a different trading strategy. If data is crypto, implement a crypto class and have a different trading strategy. I will prioritize this task for the next deadline.
- Price Data is produced in the format of lists which hold the respective data at 15 second intervals. Data is also produced in the form of a plot via Matplotlib and represents linear and polynomial regression (to be analyzed further) at 15 second (renewing plot) and 1 minute (continuous plot) intervals.
- I really spent a lot of time researching, tracking, and testing investment strategies. I believe I should have spent a little less time on this (it was very time demanding) and more time on my class hierarchy/construction.

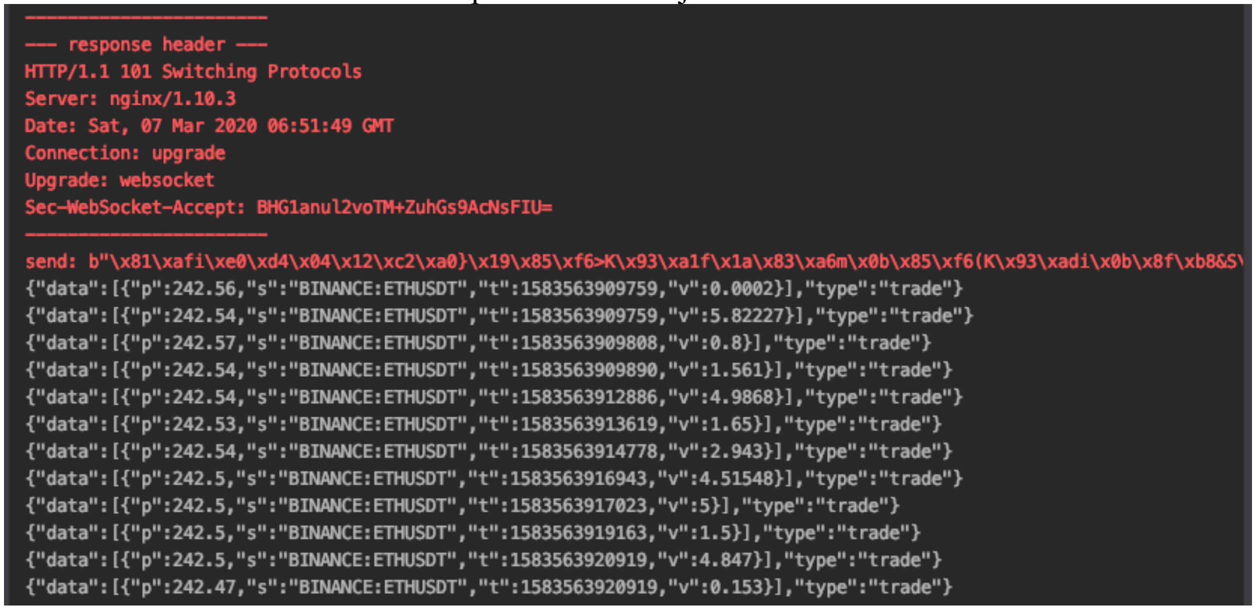
Previous Plan for Next Deadline and Adjustments:

Previous: At this stage I would like to really hone in and improve the trading strategy (optimization). I would like to analyze data capture in the previous stage (Simple Moving Averages, highPrice, lowPrice) to determine relative lows and relative highs. I would also like to use the previously mentioned data to determine a direction of trend. I will determine entry/exit strategies for the investment strategy at this time as well. At this stage relative lows and highs and trend direction will be available to view once the program is executed.

Adjustment: Elaborating on relative highs and lows, I would like to utilize the polynomial equation that is formed in the plots to determine relative highs and lows. I would like to implement a log of relative lows and highs. I would also likely adjust the time interval of 15 seconds down to 5 seconds in hopes of creating a quicker reacting bot (unless I find way to keep thread open constantly). I would also like to complete the structural design of the algorithm and its class construction through the factory method to have a more object-oriented design. I will remove the current global variables.

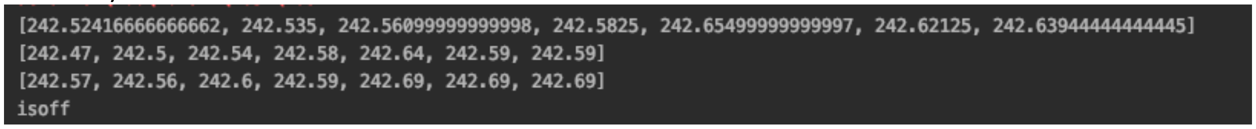
Screenshots/Gifs:

API thread connection and live data pull of JSON objects



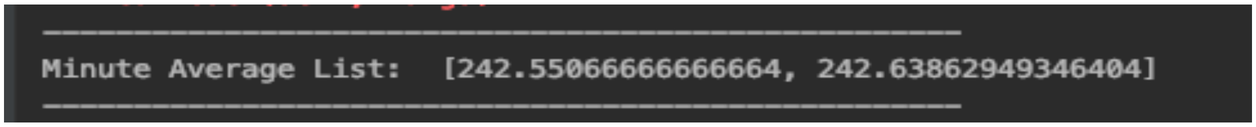
```
-----------------------
--- response header ---
HTTP/1.1 101 Switching Protocols
Server: nginx/1.10.3
Date: Sat, 07 Mar 2020 06:51:49 GMT
Connection: upgrade
Upgrade: websocket
Sec-WebSocket-Accept: BHG1anul2voTM+ZuhGs9AcNsFIU=
-----------------------
send: b"\x81\xafi\xe0\xd4\x04\x12\xc2\xa0}\x19\x85\xf6>K\x93\xa1f\x1a\x83\xa6m\x0b\x85\xf6(K\x93\xadi\x0b\x8f\xb8&S\
{"data":[{"p":242.56,"s":"BINANCE:ETHUSDT","t":1583563909759,"v":0.0002}],"type":"trade"}
{"data":[{"p":242.54,"s":"BINANCE:ETHUSDT","t":1583563909759,"v":5.82227}],"type":"trade"}
{"data":[{"p":242.57,"s":"BINANCE:ETHUSDT","t":1583563909808,"v":0.8}],"type":"trade"}
{"data":[{"p":242.54,"s":"BINANCE:ETHUSDT","t":1583563909890,"v":1.561}],"type":"trade"}
{"data":[{"p":242.54,"s":"BINANCE:ETHUSDT","t":1583563912886,"v":4.9868}],"type":"trade"}
{"data":[{"p":242.53,"s":"BINANCE:ETHUSDT","t":1583563913619,"v":1.65}],"type":"trade"}
{"data":[{"p":242.54,"s":"BINANCE:ETHUSDT","t":1583563914778,"v":2.943}],"type":"trade"}
{"data":[{"p":242.5,"s":"BINANCE:ETHUSDT","t":1583563916943,"v":4.51548}],"type":"trade"}
{"data":[{"p":242.5,"s":"BINANCE:ETHUSDT","t":1583563917023,"v":5}],"type":"trade"}
{"data":[{"p":242.5,"s":"BINANCE:ETHUSDT","t":1583563919163,"v":1.5}],"type":"trade"}
{"data":[{"p":242.5,"s":"BINANCE:ETHUSDT","t":1583563920919,"v":4.847}],"type":"trade"}
{"data":[{"p":242.47,"s":"BINANCE:ETHUSDT","t":1583563920919,"v":0.153}],"type":"trade"}
```

Output of 15 second average list, 15 second low list, 15 second high list (produced every 15 seconds)



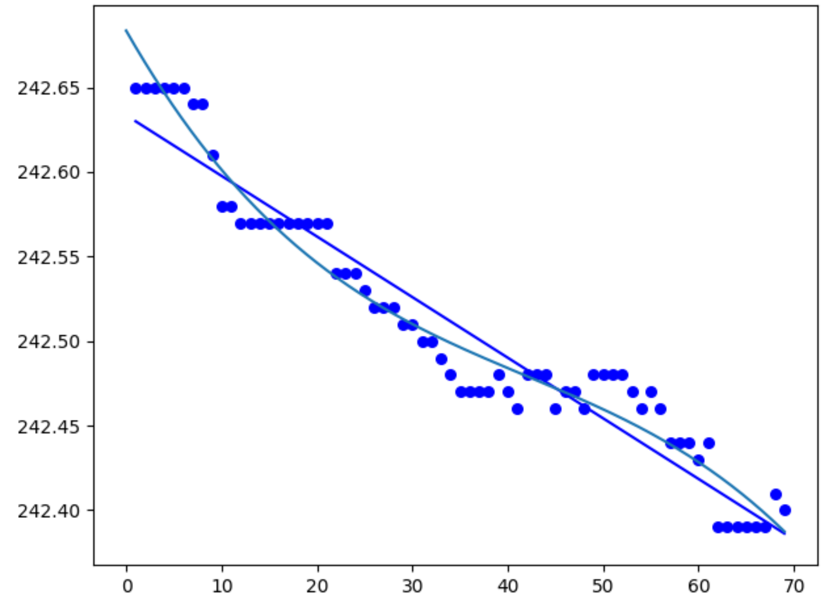
```
[242.52416666666662, 242.535, 242.56099999999998, 242.5825, 242.65499999999997, 242.62125, 242.63944444444445]
[242.47, 242.5, 242.54, 242.58, 242.64, 242.59, 242.59]
[242.57, 242.56, 242.6, 242.59, 242.69, 242.69, 242.69]
isoff
```

Output of Minute average list (produced every minute)

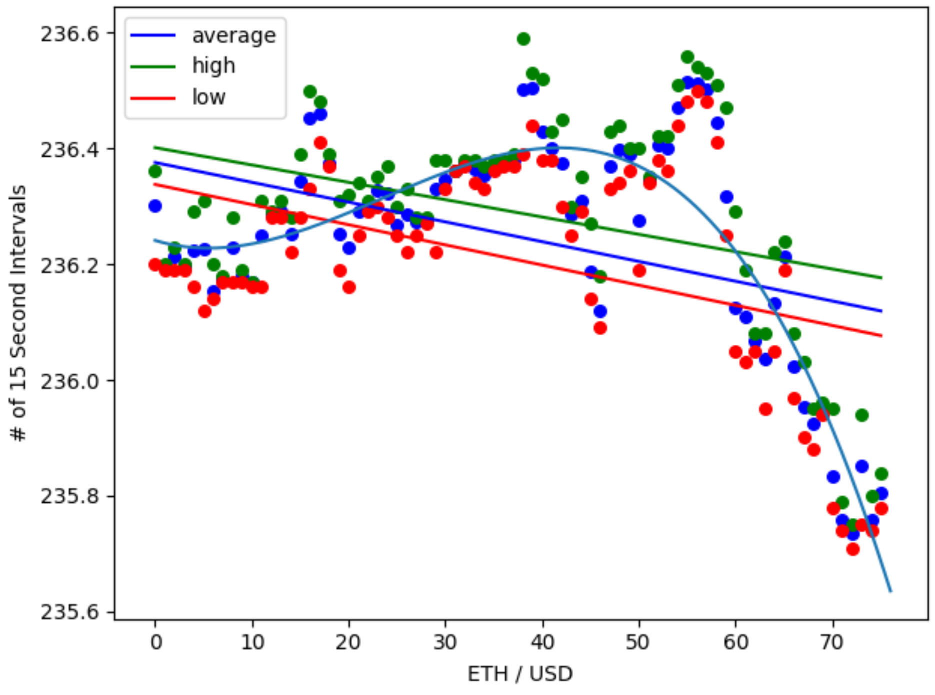


```
--------------------------------------------------------------
Minute Average List:  [242.55066666666664, 242.63862949346404]
--------------------------------------------------------------
```

15 Second plot (using aggregate data from 15 second data pulls and produced every 15 second)



1 Minute plot (using low,average,high list data and produced every minute)(labeled backwards)


average
high
low
# of 15 Second Intervals
ETH / USD