Hi Sunny –

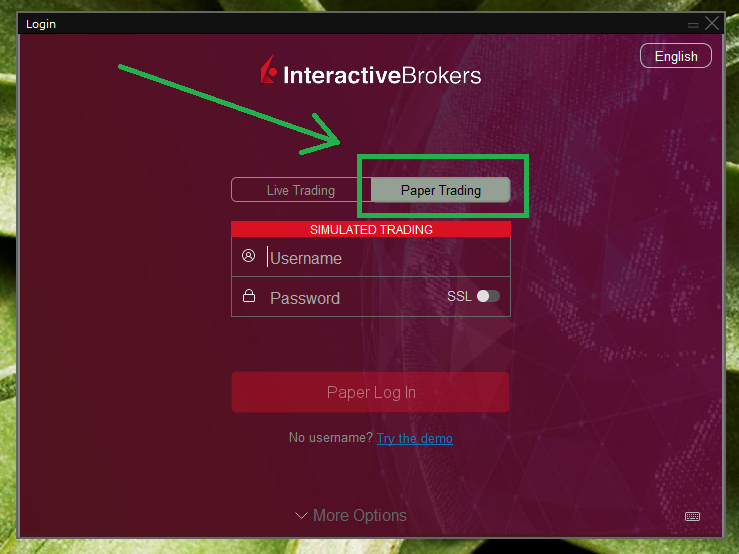
This next project will be much more in-depth than the last few.

For this one, I would like to build a python wrapper around the existing API for Interactive Brokers (an online stock trading brokerage/platform). Their API is very thorough. In fact, it is so thorough that the purpose of this python wrapper I’m seeking is so I can easily leverage it as opposed to using their code from scratch.

Their API documentation can be found here: <https://www1.interactivebrokers.com/en/index.php?f=5039>

To use their API, it’s actually necessary to download and run one of their desktop software programs. The names of these programs are Trader Workstation and IB Gateway. Either will suffice, but I recommend using Trader Workstation because it will allow you to easily see if trades and positions opened/closed by the API were successful or not.

Trader Workstation will give you two options when launching. You can choose to login using “Live Trading” or instead “Paper Trading”. Please choose Paper Trading since this will allow you to open and close positions using fake money (so there’s no consequence to trying anything you’d like). There isn’t any actual money deposited in my account, so no consequence logging into either of the two options. However, the Paper Trading account will have a fake balance of $1,000,000 in it which will be useful to see stock positions opened and closed using the API.



Username: quad8472  
Password: !Data2020!

Functionalities I am trying to achieve…

The API Interactive Brokers provides is almost limitless. But I will try my best to concisely describe the few essential functionalities I am trying to recreate…

1 – Get Data

2 – Buy Order

3 – Sell Order

4 – Retrieve last price

5 – Get current account holdings

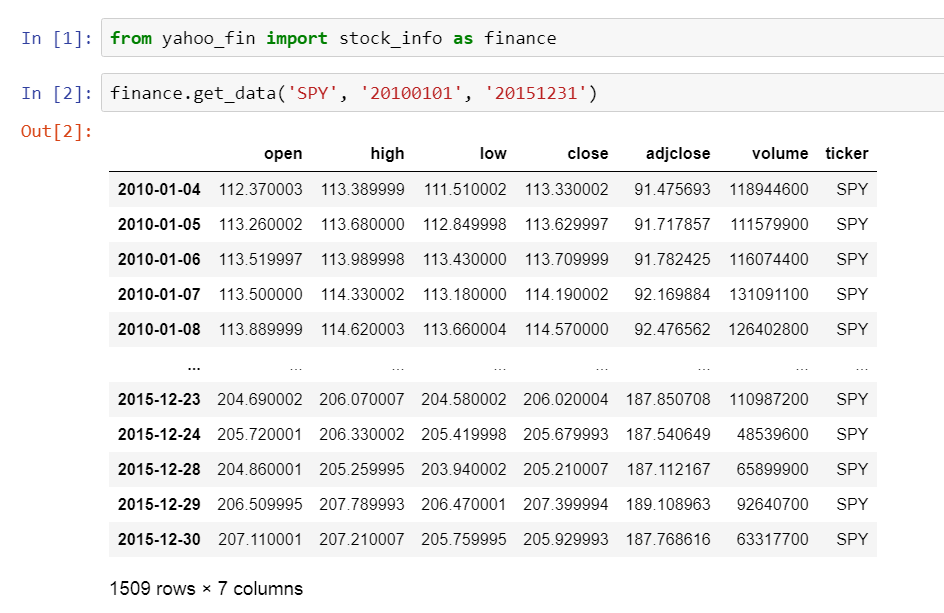
6 – Get any pending orders

7 – Cancel / modify any pending orders

8 – Modify portfolio (custom functionality)

1 – Get Data

This is the first piece of functionality I am trying to have easily replicated with the API wrapper. I currently use the yahoo\_fin python package to retrieve historical stock data (please see screenshot below).



I am looking to replace the yahoo\_fin library I currently rely upon for stock data. Yahoo can sometimes be unreliable and return errors. I believe this will be a much more stable and long term solution. A summary of the functionality within IB’s API can be found here: <https://interactivebrokers.github.io/tws-api/historical_bars.html>

There is one additional piece of functionality I am hoping to achieve. Yahoo stock data is only available in daily increments. However, I believe Interactive Brokers allows for retrieving historical stock data for increments smaller than this (e.g., 5-min, 15-min, 1-hr, etc). Can this functionality be added as well please?

The function could be written to allow for the increment to be define with an increment method. For example: get\_data(‘SPY’, start\_date = ‘20100101’, end\_date = ‘20151231’, increment = ‘D’)

In this example, “increment” could be changed to be ‘5-min’, ’15-min’, etc, etc…

Interactive Brokers requires a market data subscription for historical data. I just subscribed for this. They said it could take up to 48 hours to become active… fyi in case the historical data aspect of their API isn’t working upon first try.

2 – Buy Order

The purpose of this functionality will be to place a simple buy order of a stock. For example:

buy\_order(ticker = ‘SPY’, shares = 100, type = ‘market’, duration = ‘day’)

The above would indicate buying 100 shares of SPY using a “market” order type. A market order type means that whatever the current prevailing price is for SPY is the price that it will be bought for. The duration of ‘day’ means that the order is only good for the current day and will be voided if it was not filled by the end of the day.

Alternatively, I would also like to be able to place “limit” orders using this function as well. An example of submitting a limit function could be: buy\_order(ticker = ‘SPY’, shares = 100, type = ‘limit’, price = 330.1, duration = ‘gtc’)

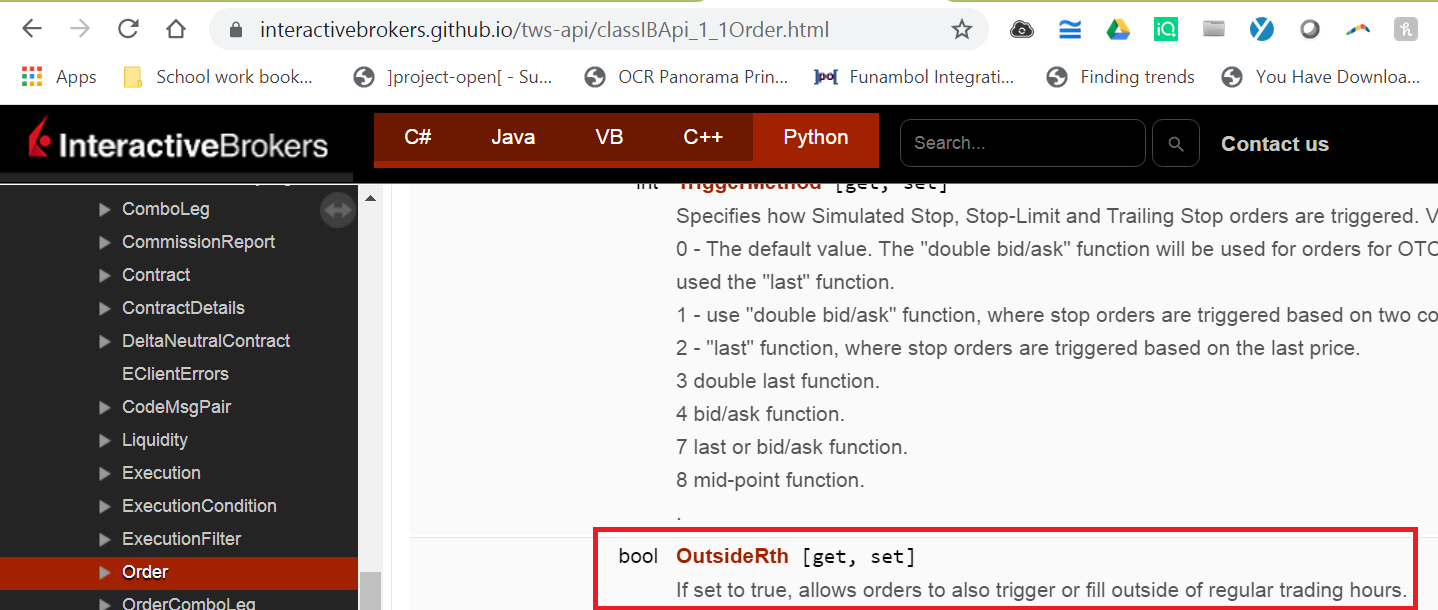
A limit order means that the buy order would only go through if the shares of SPY could be bought for a maximum of $330.10 – if the price remains above this level while the order is open, then the order would not go through. The duration of ‘gtc’ means “good until cancelled’ and means that the order will forever remain outstanding until it is either filled or deliberately cancelled by the user.

There is a third order type I would like to be able to use as well. This one is unique to Interactive Brokers and it’s called “Adaptive Algo”. [https://interactivebrokers.github.io/tws-api/ibalgos.html#adaptive](https://interactivebrokers.github.io/tws-api/ibalgos.html" \l "adaptive)

<https://www1.interactivebrokers.com/en/index.php?f=19091>

There is one more functionality I would like to be able to implement to a buy order. Its something called “Outside Regular Hours”. Interactive Brokers actually allows for buying and selling of stocks outside of regular market hours. Buy and Sell orders by default will not be implemented outside of regular trading hours, however, there is an option to allow order to be filled at any time of day (including the middle of the night). Please see the following link: https://www1.interactivebrokers.com/en/index.php?f=22063

I found the API component that allows for this. Please see below:



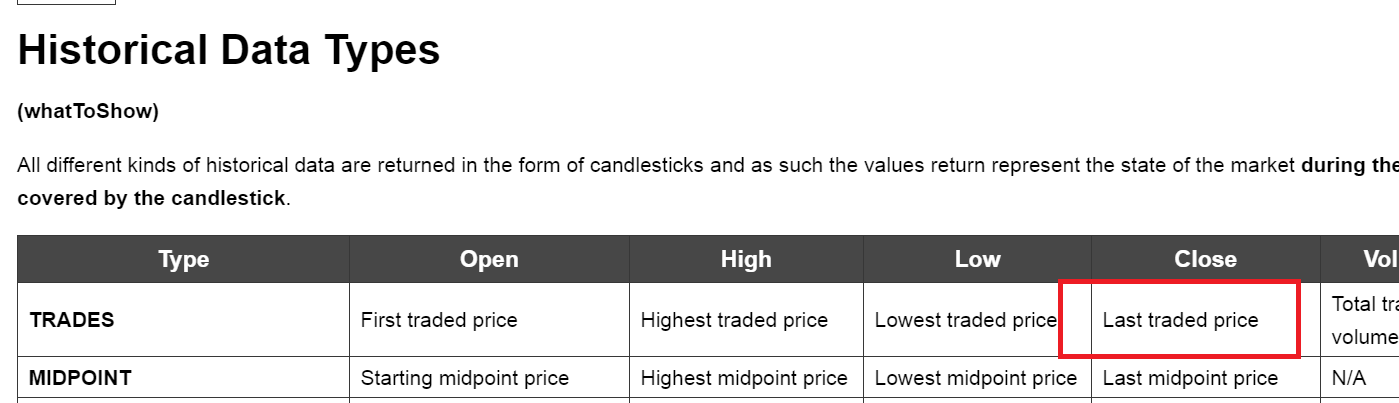
3 – Sell Order

This is the same as the Buy Order function but is intended for selling stocks.

Order types of market, limit and “adaptive” orders should optionally be able to be chosen. And so should either ‘day’ or ‘gtc’ options be selectable.

4 – Last Traded Price

This functionality will be much of the same as the get\_data function. For this one, I would like to be able to retrieve the last price that shares of the stock traded for. I believe this is available per the screenshot below:



5 – Get Account Holdings

For this function, I would like to be able to retrieve the current stocks currently in the portfolio. The returned value could be something as simple as a list of lists or dictionary summarizing the current portfolio holdings. For example:

[  
[‘SPY’, 100], [‘ORCL’, 30], [‘QQQ’, -25]  
]

If you’ll notice, the portfolio outlined above is actually short 25 shares of the QQQ (-25 shares). Just to provide another example… if I wanted to close out this position, I could then use the buy\_order function to purchase 25 shares of QQQ to bring this number to 0 (just as an example of the logic behind these API functions).

6 – Get any pending orders

This functionality will be important after submitting limit orders. A limit order may not be filled immediately and may potentially sit idle for quite some time. The purpose of this function will be to check if any orders are pending to be filled.

7 – Cancel or modify a pending order

After retrieving a list (or dictionary, etc) of any pending orders… this function can be used to cancel or modify a pending order.

8 – Modify existing portfolio

This is the custom functionality I had alluded to above. I would like to have a function that can implement a desired portfolio. The input will be a list of lists of what the final desired portfolio will be. For example, an input could look like this:

[  
[‘SPY’, 30], [‘MSFT’, 45], [‘EXPD’, -21], [‘ORCL’, -6], [‘AMZN’, 12]  
]

In this example, the above list of lists is what I’d like the *final* portfolio to look like after the function has completed.

The logic behind this will need to be smarter than just closing all existing stock positions and opening the new ones as defined by the list of lists.

The function will first need to query the API to find what the current account holdings are. Then, it will need to be determined which stock positions need to be modified instead of completely closed. For example, if the portfolio is currently holding 25 shares of SPY prior to this list of lists being submitted, then the function would need to submit a buy\_order for 5 shares to bring the total position of SPY to 30 (as defined in the list of lists). We can call this step “modification”.

Along with modification, the function will also need to determine which stocks are not being currently held. Positions for these stocks can be opened using either a buy\_order (for a long position) or sell\_order (for a short position). We can call this step “new holdings”.

Every share bought or sold results in a fee being charged. So the purpose of not completely closing all the positions and then opening new ones is to save on fees (this really adds up over time).

There is one last detail I am going to throw in here for this functionality. We discussed the different order types above. They were market, limit and adaptive orders. I would like the option to have this function carry out these modifications using either market or adaptive order types (but not limit orders… I think these would be too tough).

Market order is straight forward. These orders are almost always filled immediately… so if the function needs to close a position, or partially close a position, it can easily and quickly do so and move on to the next task.

Adaptive order types may take more nuance. For example, the function could easily submit all its initial modification and buy/sell orders immediately… however, the adaptive order function does not necessarily finish buying or selling shares immediately. This order type is smart and seeks to find the best price for the stock. So the function will need to submit the orders and then wait for a period of time before checking if there are any pending orders still open… and waiting to see if these eventually get filled etc. When trading stocks after hours (at nighttime for example), sometimes stock positions take a long time to get filled because there’s so much less trading activity… hence maybe the need for the function to wait around a bit using a while loop or something to check on order statuses. Not sure what the best bet is here… so may take some brainstorming.