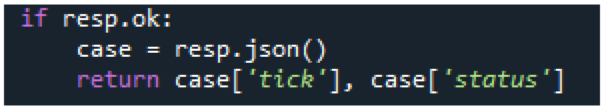
from time import sleep, sleep is an important function for python to run and rest and check the functions go through again and again.

s.header.update() - to avoid soccer connection

s.get
s.post
s.is the connection for Everything send from the RIT to the Python

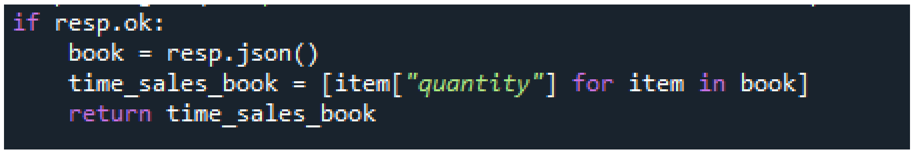
The main function is where our algo proper will be, that where decision making will take place in terms of how to launch trades, why and when, and how to cancel orders etc.
And the other functions are support functions: the support functions are getting our quotes, checking our positions,

Let json() unpack the dictionary
book =j.son()
case=j.som()
news_query=j.son()
Then select the information (values, or variables,data numbers) and return the information Ineed
Two ways:\1.unpaced dictionary after j.son 中括号[‘ ‘] 冒号 variable



```
if resp.ok:
    case = resp.json()
    return case['tick'], case['status']
```

2.define a new variable = [item [“”]要找的variable+ for …item in… book]中括号



```
if resp.ok:
    book = resp.json()
    time_sales_book = [item["quantity"] for item in book]
    return time_sales_book
```

Calling codes in multiple functions

Video 3:
def get_tick():   ⇐ support function
    resp = s.get('http//localhost:9999/v1/case')
    if resp.ok:
        case = resp.json()
        return case['tick'], case['status']
Call ‘case’ dictionary and select specific information (values, or variables, data numbers)

def main():   ⇐ main function
      tick, status = get_tick()
Call back and knows the valuables in different other functions

Transfer the information(variables, values, data) from get_tick() th defined function to our main function, so that we could use the information in our main function

[0] [1] [2] tied with the \variable,\ tied with the function defined, or tied with the dictionary called packed and unpacked , to get the most closed information needed (the first , the best xx, the lasted time)

Video4:

Video5:get.tick()function: get order/tick number and status(‘active’ or ‘stop’)

Indentation are python struction, inside the for loop or the while loop, inside the function but not inside the if statement. Indentation is not accidental.
s.get is part of the request function
resp = s.get() resp = s.post() resp = s.delete () get information or execute

In the support function def and resp = s.get() define and support functions and go to different website for getting information

```
import requests
from time import sleep

s = requests.Session()
s.headers.update({'X-API-key': 'VHXVRO0J'})

def get_tick():
    resp = s.get('http://localhost:9999/v1/case')
    if resp.ok:
        case = resp.json() <= return me with a dictionary
        return case['tick'], case['status']
Call the specific information variable needed out

case
Out[44]:
{'name': 'ALGO2e v2',
 'period': 1,
 'tick' <= a variable name : 107,
 'ticks_per_period': 300,
 'total_periods': 1,
 'status': 'ACTIVE',
 'is_enforce_trading_limits': False}

def main():
    tick, status = get_tick()
```

Run the single line of code, run more than one line can;t get information,

```
tick, status = get_tick()

tick
Out[67]: 0

tick
Out[68]: 0

tick
Out[69]: 0

status
Out[70]: 'STOPPED'

status
Out[71]: 'STOPPED'
```

Video 6: get_bid_ask() function: get all information in the book ~tick is a time-moving track variable, tick range from (1,300), for some dictionary include the time-moving variable

```
def get_bid_ask(ticker):
    payload = {'ticker': 'AC'}  {'ticker': 'tick'} change into  {'ticker': 'AC'}
    resp = s.get ('http://localhost:9999/v1/securities/book', params = payload)
    if resp.ok:
        book = resp.json()
        bid_side_book = book['bids']
        ask_side_book = book['asks']

        bid_prices_book = [item['price'] for item in bid_side_book]
        ask_prices_book = [item['price'] for item in bid_side_book]

        best_bid_price = bid_prices_book[0]
        best_ask_price = ask_prices_book[0]

        return best_bid_price, best_ask_price


def main():
    tick, status = get_tick()
    best_bid_price, best_ask_price = get_bid_ask('AC')
```

Run these lines of codes to get the information and the dictionary

```
 payload = {'ticker': 'AC'}  {'ticker': 'tick'} change into  {'ticker': 'AC'}
    resp = s.get ('http://localhost:9999/v1/securities/book', params = payload)
    if resp.ok:
        book = resp.json()
```

```
payload = {'ticker': 'AC'}

resp = s.get ('http://localhost:9999/v1/securities/book', params = payload)

resp
Out[76]: <Response [200]>

book = resp.json()

book
Out[78]:
{'bids': [{'order_id': 11170,
   'period': 1,
   'tick': 137,
   'trader_id': 'ANON',
   'ticker': 'AC',
   'quantity': 1100.0,
   'price': 28.23,
   'type': 'LIMIT'2
   'action': 'BUY',
   'quantity_filled': 500.0,
   'vwap': 28.23,
   'status': 'OPEN'},
  {'order_id': 11743,
   'period': 1,
   'tick': 144,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.23,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 11952,
 'period': 1,
 'tick': 147,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1300.0,
 'price': 28.23,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9861,
 'period': 1,
 'tick': 120,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1300.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9970,
 'period': 1,
 'tick': 122,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10076,
 'period': 1,
 'tick': 123,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10143,
 'period': 1,
 'tick': 124,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10443,
 'period': 1,
 'tick': 128,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10955,
 'period': 1,
 'tick': 134,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 11087,
 'period': 1,
 'tick': 136,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9424,
 'period': 1,
 'tick': 115,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 300.0,
 'vwap': 28.21,
 'status': 'OPEN'},
{'order_id': 9496,
 'period': 1,
 'tick': 116,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9632,
 'period': 1,
 'tick': 117,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10121,
 'period': 1,
 'tick': 124,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 11855,
 'period': 1,
 'tick': 146,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 11899,
 'period': 1,
 'tick': 147,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.21,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9749,
 'period': 1,
 'tick': 119,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.2,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10057,
 'period': 1,
 'tick': 123,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.2,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10165,
 'period': 1,
 'tick': 124,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.2,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10169,
 'period': 1,
 'tick': 124,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.2,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'}],
'asks': [{'order_id': 11992,
 'period': 1,
 'tick': 148,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.25,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10003,
 'period': 1,
 'tick': 122,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.89,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 1000.0,
 'vwap': 28.89,
 'status': 'OPEN'},
{'order_id': 10179,
 'period': 1,
 'tick': 124,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.89,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10996,
 'period': 1,
 'tick': 135,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.89,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 494,
 'period': 1,
 'tick': 1,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 1044,
 'period': 1,
 'tick': 5,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 4033,
 'period': 1,
 'tick': 44,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 4625,
 'period': 1,
 'tick': 52,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 5676,
 'period': 1,
 'tick': 65,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 6421,
 'period': 1,
 'tick': 75,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 6895,
 'period': 1,
 'tick': 81,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 8225,
 'period': 1,
 'tick': 99,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9069,
 'period': 1,
 'tick': 110,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9129,
 'period': 1,
 'tick': 111,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9416,
 'period': 1,
 'tick': 115,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10238,
 'period': 1,
 'tick': 125,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 900.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10308,
 'period': 1,
 'tick': 126,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 10725,
 'period': 1,
 'tick': 131,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.9,
 'type': 'LIMIT',
 'action': 'SELL',
 'quantity_filled': 0.0,
 'vwap': None,
  'status': 'OPEN'},
 {'order_id': 11402,
  'period': 1,
  'tick': 140,
  'trader_id': 'ANON',
  'ticker': 'AC',
  'quantity': 1100.0,
  'price': 28.9,
  'type': 'LIMIT',
  'action': 'SELL',
  'quantity_filled': 0.0,
  'vwap': None,
  'status': 'OPEN'},
 {'order_id': 11620,
  'period': 1,
  'tick': 143,
  'trader_id': 'ANON',
  'ticker': 'AC',
  'quantity': 1000.0,
  'price': 28.9,
  'type': 'LIMIT',
  'action': 'SELL',
  'quantity_filled': 0.0,
  'vwap': None,
  'status': 'OPEN'}]}
```

Dictionary hierarchy: Book -> ‘bids’ -> [0] the best bid order information -> [‘price’] the bid price

Try to Run lines of codes to get all the bid side order information in the dictionary book

book[‘bids’]

```
out[79]{'bids': [{'order_id ': 11170, ⇐this is the best bid order
  'period': 1,
  'tick': 137,<= the tick number
  'trader_id': 'ANON',
  'ticker': 'AC', <= the ticker name
  'quantity': 1100.0, <= buy in trade quantity is 1100
  'price': 28.23, <= the best bid price
  'type': 'LIMIT', ⇐the order type
  'action': 'BUY', <= the market trade action is BUY or SELL
  'quantity_filled': 500.0, <= the quantity it has been filled is 500.
```

```
 'vwap': 28.23, <= the volume weighted average price
 'status': 'OPEN'}, <= current trading statues
{'order_id': 11743, <= this is the second best price order , buying price descending
 'period': 1,<=...
 'tick': 144,<=...
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1200.0,
 'price': 28.23,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 11952,
 'period': 1,
 'tick': 147,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1300.0,
 'price': 28.23,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9861,
 'period': 1,
 'tick': 120,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1300.0,
 'price': 28.22,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 0.0,
 'vwap': None,
 'status': 'OPEN'},
{'order_id': 9970,
 'period': 1,
 'tick': 122,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1000.0,
  'price': 28.22,
  'type': 'LIMIT',
  'action': 'BUY',
  'quantity_filled': 0.0,
  'vwap': None,
  'status': 'OPEN'},

book['bids'][0]
Out[80]:
{'order_id': 11170,
 'period': 1,
 'tick': 137,
 'trader_id': 'ANON',
 'ticker': 'AC',
 'quantity': 1100.0,
 'price': 28.23,
 'type': 'LIMIT',
 'action': 'BUY',
 'quantity_filled': 500.0,
 'vwap': 28.23,
 'status': 'OPEN'}

book['bids'][0]['price']
Out[81]: 28.23
```

```
    bid_prices_book = [item['price'] for item in bid_side_book]
    ask_prices_book = [item['price'] for item in bid_side_book]
```

For these two lines you could change into any other variables you want to detect in the dictionary on the ticker AC's bid side book.

Run these lines of codes to get the information and the dictionary

```
bid_prices_book = [item['price'] for item in bid_side_book]
ask_prices_book = [item['price'] for item in bid_side_book]

bid_prices_book
Out[92]:
[26.97,
 26.8,
 26.73,
 26.72,
 26.72,
 26.72,
 26.71,
 26.71,
 26.7,
 26.7,
 26.7,
 26.7,
 26.69,
 26.69,
 26.69,
 26.68,
 26.68,
 26.67,
 26.67,
 26.67]
```

```
        best_bid_price = bid_prices_book[0]
        best_ask_price = ask_prices_book[0]
```

These lines of codes find the best bid price for all the ticke’s bids prices listed in the book.

Change   payload = {'ticker': ‘AC’} to payload = {'ticker': ticker}
Run

```
def main():
  best_bid_price, best_ask_price = get_bid_ask('AC')
```

This will guide to get the best bid price for ticker AC in the main function.

Video seven: get_position() function: get securities info: This lists all the current order ticks, ticker names, current position: best stock (bid and ask) price, trade volumes, maximized trade size. api order per second.

Define lines of codes

```
def get_position():
    resp = s.get('http://localhost:9999/v1/securities')
    if resp.ok:
        book = resp.json()
```

Run these lines of codes to get the information and the dictionary

```
resp = s.get('http://localhost:9999/v1/securities')

book = resp.json()

book = resp.json()

return abs(book[0]['position']) + abs(book[1]['position']) + abs(book[2]['position'])
  Cell In[115], line 1
    return abs(book[0]['position']) + abs(book[1]['position']) + abs(book[2]['position'])
    ^
SyntaxError: 'return' outside function


book = resp.json()

book
Out[117]:
[{'ticker': 'CNR', <= this is the ticker name
  'type': 'STOCK',<= this is the security type
  'size': 1, <=  this is the quantity of the securities
  'position': 0.0,<=this is  the number of shares held
  'vwap': 0.0,<=this is the volume weighted average price
  'nlv': 0.0,<= this is the net liquidation value
  'last': 149.0,<= this is the last trading price
  'bid': 149.0,<=this is the current highest bid price
  'bid_size': 6000.0,<=this is the number of shares traded in the bid price
  'ask': 149.56,
  'ask_size': 2900.0,<= this is the size of shares in the highest aks
  'volume': 6694800.0,<= this is the total trading volume for the securitis
  'unrealized': 0.0,<= this is the unrealized profit/loss
  'realized': 0.0,<=this is the realized profit/ loss
  'currency': '',<=this is the currency traded
  'total_volume': 6694800.0,<= this is the tota training volumes
  'limits': [{'name': 'LIMIT-STOCK', 'units': 1.0}],
  'interest_rate': 0.0,
  'is_tradeable': True,
  'is_shortable': True,
  'start_period': 1,
  'stop_period': 1,
  'description': 'Shares of CNR Corp.',
  'unit_multiplier': 1,
  'display_unit': 'share',
  'start_price': 151.0,<=this is the starting price of the securities
  'min_price': 1.0,<=the minimized price allowed
  'max_price': 1000.0,<=the maximized price allowed
  'quoted_decimals': 2,
  'trading_fee': 0.0027,<= this is the commission fee per trade
  'limit_order_rebate': 0.005,
  'min_trade_size': 0,
  'max_trade_size': 50000,
  'required_tickers': None,
  'underlying_tickers': None,
  'bond_coupon': 0.0,
  'interest_payments_per_period': 0,
  'base_security': '',
  'fixing_ticker': None,
  'api_orders_per_second': 100,<= this is the maximized number of API orders per second
  'execution_delay_ms': 0,
  'interest_rate_ticker': None,
  'otc_price_range': 0.0},
 {'ticker': 'RY',
  'type': 'STOCK',
  'size': 1,
  'position': 0.0,
  'vwap': 0.0,
  'nlv': 0.0,
  'last': 102.93,
  'bid': 102.84,
  'bid_size': 23200.0,
  'ask': 102.93,
  'ask_size': 14500.0,
  'volume': 60219800.0,
  'unrealized': 0.0,
  'realized': 0.0,
  'currency': '',
  'total_volume': 60219800.0,
  'limits': [{'name': 'LIMIT-STOCK', 'units': 1.0}],
  'interest_rate': 0.0,
  'is_tradeable': True,
  'is_shortable': True,
  'start_period': 1,
  'stop_period': 1,
  'description': 'Shares of RY',
  'unit_multiplier': 1,
  'display_unit': 'share',
  'start_price': 103.0,
  'min_price': 1.0,
  'max_price': 1000.0,
  'quoted_decimals': 2,
  'trading_fee': -0.0014,
  'limit_order_rebate': -0.0034,
  'min_trade_size': 0,
  'max_trade_size': 50000,
  'required_tickers': None,
  'underlying_tickers': None,
  'bond_coupon': 0.0,
  'interest_payments_per_period': 0,
  'base_security': '',
  'fixing_ticker': None,
  'api_orders_per_second': 100,
  'execution_delay_ms': 0,
  'interest_rate_ticker': None,
  'otc_price_range': 0.0},
 {'ticker': 'AC',
  'type': 'STOCK',
  'size': 1,
  'position': 0.0,
  'vwap': 0.0,
  'nlv': 0.0,
  'last': 19.94,
  'bid': 19.78,
  'bid_size': 1600.0,
  'ask': 19.85,
  'ask_size': 200.0,
  'volume': 6948500.0,
  'unrealized': 0.0,
  'realized': 0.0,
  'currency': '',
  'total_volume': 6948500.0,
  'limits': [{'name': 'LIMIT-STOCK', 'units': 1.0}],
  'interest_rate': 0.0,
  'is_tradeable': True,
  'is_shortable': True,
  'start_period': 1,
  'stop_period': 1,
  'description': 'Shares of AC',
  'unit_multiplier': 1,
  'display_unit': 'share',
  'start_price': 20.0,
   'min_price': 1.0,
   'max_price': 1000.0,
   'quoted_decimals': 2,
   'trading_fee': 0.0015,
   'limit_order_rebate': 0.0026,
   'min_trade_size': 0,
   'max_trade_size': 50000,
   'required_tickers': None,
   'underlying_tickers': None,
   'bond_coupon': 0.0,
   'interest_payments_per_period': 0,
   'base_security': '',
   'fixing_ticker': None,
   'api_orders_per_second': 100,
   'execution_delay_ms': 0,
   'interest_rate_ticker': None,
   'otc_price_range': 0.0}]
```

```
 return abs(book[0]['position']) + abs(book[1]['position']) + abs(book[2]['position'])
```

```
   book[0]['position']
Out[127]: 0.0

book[0]['position']
Out[128]: 0.0

book[0]['position']
Out[129]: 0.0
```

```
def main():   ⇐ main function
     position = get_position()
```

Call back and knows the valuables in different other functions

Transfer the information from get_position()

to our main function, so that we could use the information in our main function

Video Eight: get_time_sales: get all the orders traded price information and traded quantities on the book. Track the closest price changes and list the traded quantity on the tick price (bid/ask)

```
def get_time_sales(ticker):
   payload = {'ticker': 'AC', 'limit': 1}
   resp = s.get ('http://localhost:9999/v1/securities/tas', params = payload)
   if resp.ok:
      book = resp.json()
```

Run these lines of codes to get the information and the dictionary

```
payload = {'ticker': 'AC', 'limit': 1}

resp = s.get ('http://localhost:9999/v1/securities/tas', params = payload)

book = resp.json()
```

the timestamped sequence of trades. In the context of your API response, **tick appears to be a numerical identifier for the sequence of trades** within a given period. A higher tick number likely means a more recent trade.

From your data:

```yaml
{'id': 12739, 'period': 1, 'tick': 260, 'price': 20.88, 'quantity': 1400.0},
{'id': 12708, 'period': 1, 'tick': 259, 'price': 20.71, 'quantity': 200.0},
{'id': 12133, 'period': 1, 'tick': 248, 'price': 20.86, 'quantity': 1300.0},
```

- **Tick 260** is more recent than **Tick 259** and **Tick 248**.
- Each tick represents a different event in time.

Why Does `{'ticker': 'AC', 'limit': 1}` Give the Most Recent Tick?

1. **Limit** = **1** tells the API to return only the **latest trade event**.
2. The API likely orders trades in descending order by **tick number** (newest first).
3. Since the most recent trade has the highest tick number, the API naturally returns that.

```
book
Out[153]:
[{'id': 12739, 'period': 1, 'tick': 260, 'price': 20.88, 'quantity': 1400.0},
 {'id': 12737, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 600.0},
 {'id': 12736, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 500.0},
 {'id': 12735, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 600.0},
 {'id': 12734, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 600.0},
 {'id': 12733, 'period': 1, 'tick': 260, 'price': 20.71, 'quantity': 200.0},
 {'id': 12732, 'period': 1, 'tick': 260, 'price': 20.96, 'quantity': 200.0},
 {'id': 12731, 'period': 1, 'tick': 260, 'price': 20.97, 'quantity': 1200.0},
 {'id': 12730, 'period': 1, 'tick': 260, 'price': 20.97, 'quantity': 400.0},
 {'id': 12729, 'period': 1, 'tick': 260, 'price': 20.98, 'quantity': 1100.0},
 {'id': 12728, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 500.0},
 {'id': 12727, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 400.0},
 {'id': 12726, 'period': 1, 'tick': 260, 'price': 21.01, 'quantity': 100.0},
 {'id': 12722, 'period': 1, 'tick': 260, 'price': 21.01, 'quantity': 1300.0},
 {'id': 12718, 'period': 1, 'tick': 260, 'price': 21.15, 'quantity': 400.0},
 {'id': 12717, 'period': 1, 'tick': 260, 'price': 21.14, 'quantity': 1000.0},
 {'id': 12716, 'period': 1, 'tick': 260, 'price': 21.14, 'quantity': 100.0},
 {'id': 12715, 'period': 1, 'tick': 260, 'price': 21.12, 'quantity': 1200.0},
 {'id': 12712, 'period': 1, 'tick': 260, 'price': 20.96, 'quantity': 1300.0},
 {'id': 12708, 'period': 1, 'tick': 259, 'price': 20.71, 'quantity': 200.0},
 {'id': 12707, 'period': 1, 'tick': 259, 'price': 20.72, 'quantity': 1200.0},
 {'id': 12706, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 1400.0},
 {'id': 12705, 'period': 1, 'tick': 259, 'price': 21.13, 'quantity': 1600.0},
 {'id': 12704, 'period': 1, 'tick': 259, 'price': 20.72, 'quantity': 400.0},
 {'id': 12703, 'period': 1, 'tick': 259, 'price': 20.72, 'quantity': 200.0},
 {'id': 12702, 'period': 1, 'tick': 259, 'price': 21.1, 'quantity': 1100.0},
 {'id': 12699, 'period': 1, 'tick': 259, 'price': 20.72, 'quantity': 700.0},
 {'id': 12698, 'period': 1, 'tick': 259, 'price': 20.73, 'quantity': 100.0},
 {'id': 12697, 'period': 1, 'tick': 259, 'price': 20.74, 'quantity': 200.0},
 {'id': 12696, 'period': 1, 'tick': 259, 'price': 20.92, 'quantity': 200.0},
 {'id': 12695, 'period': 1, 'tick': 259, 'price': 20.92, 'quantity': 1300.0},
 {'id': 12694, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 100.0},
 {'id': 12693, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 1300.0},
 {'id': 12690, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 100.0},
 {'id': 12689, 'period': 1, 'tick': 259, 'price': 20.93, 'quantity': 1400.0},
 {'id': 12683, 'period': 1, 'tick': 259, 'price': 20.74, 'quantity': 1200.0},
 {'id': 12679, 'period': 1, 'tick': 259, 'price': 20.74, 'quantity': 200.0},
 {'id': 12678, 'period': 1, 'tick': 259, 'price': 20.96, 'quantity': 1000.0},
 {'id': 12677, 'period': 1, 'tick': 259, 'price': 20.89, 'quantity': 200.0},
 {'id': 12676, 'period': 1, 'tick': 259, 'price': 20.89, 'quantity': 1200.0},
 {'id': 12673, 'period': 1, 'tick': 259, 'price': 21.01, 'quantity': 100.0},
 {'id': 12670, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 100.0},
 {'id': 12669, 'period': 1, 'tick': 259, 'price': 21.14, 'quantity': 900.0},
 {'id': 12668, 'period': 1, 'tick': 259, 'price': 21.03, 'quantity': 100.0},
 {'id': 12663, 'period': 1, 'tick': 259, 'price': 21.01, 'quantity': 1400.0},
 {'id': 12662, 'period': 1, 'tick': 259, 'price': 21.03, 'quantity': 1100.0}]
```

This function is useful in the market making algo, where your order bigger or smaller than the bid_ask spread

```
  time_sales_book = [item["quantity"] for item in book]
        return time_sales_book
```

Pick the list of prices for ticker AC on the most recent base

```
time_sales_book = [item["quantity"] for item in book]

time_sales_book
Out[157]:
```

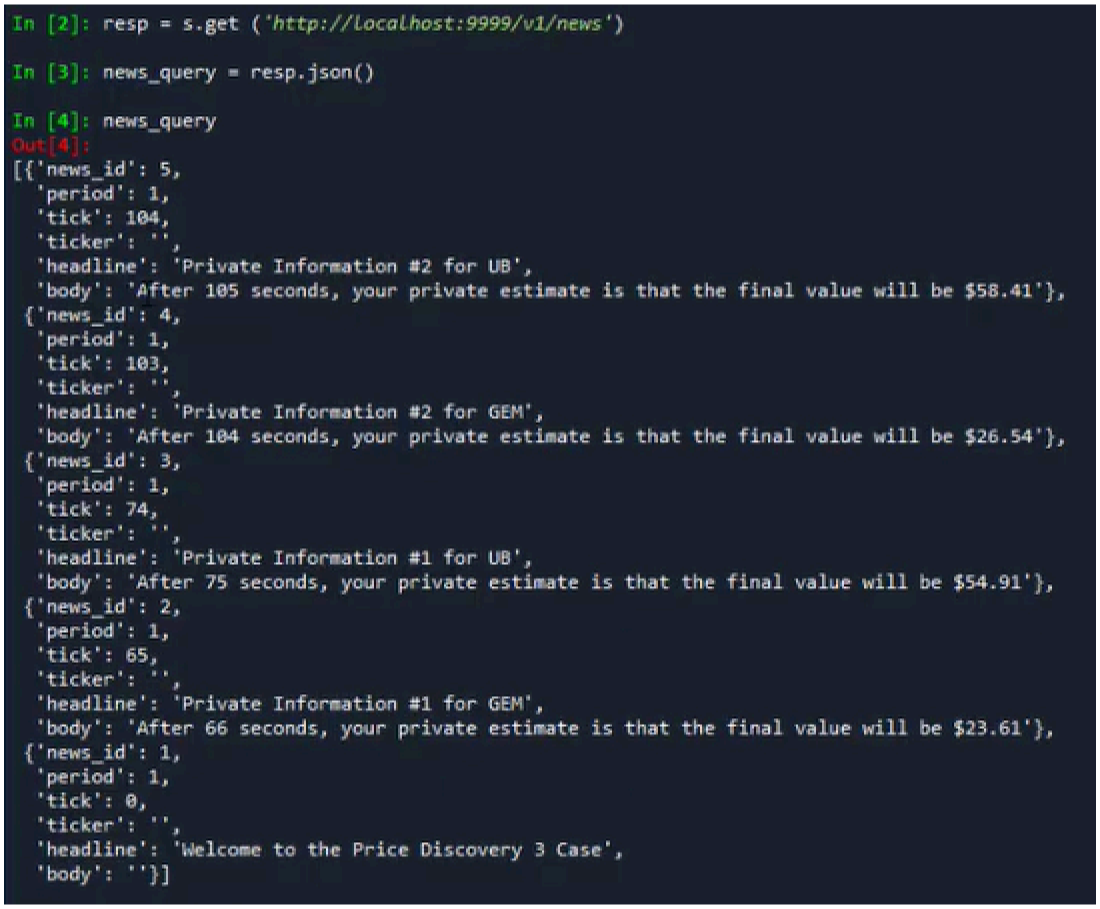
```
[300.0,
 700.0,
 700.0,
 200.0,
 600.0,
 400.0,
 800.0,
 100.0,
 400.0,
 200.0,
 800.0,
 800.0,
 200.0,
 700.0,
 100.0,
 900.0,
 100.0,
 800.0,
 100.0,
 700.0,
 100.0,
 800.0,
 100.0,
 800.0,
 200.0,
 500.0,
 200.0,
 600.0,
 400.0,
 600.0,
 100.0,
 800.0,
 800.0,
 200.0,
 100.0,
 600.0,
 200.0,
 800.0,
 600.0,
 100.0,
 700.0,
 100.0,
 700.0,
 700.0]
```

def main():   ⇐ main function
    time_ and _sales= get_time_sales(‘AC’)
Call back and knows the valuables in different other functions
Transfer the information from get_time_sales()
 to our main function, so that we could use the information in our main function

Video Nine: Get News Query:

```
def get_news():
   resp = s.get ('http://localhost:9999/v1/news')
   if resp.ok:
      news_query = resp.json()
```

Run these lines of codes to get the information and the dictionary
Body: ⟵-the content of the news , this incur private information on the valuation and prices predictions
Headline: ⟵identity if the news, for example “private information”
Tick:<-- the tick number, the most recent time in trading
News_id: ⟵ the number of piece of news, the ranks of news happening in sequence, descending shows the latest



```
In [2]: resp = s.get ('http://localhost:9999/v1/news')

In [3]: news_query = resp.json()

In [4]: news_query
Out[4]:
[{'news_id': 5,
  'period': 1,
  'tick': 104,
  'ticker': '',
  'headline': 'Private Information #2 for UB',
  'body': 'After 105 seconds, your private estimate is that the final value will be $58.41'},
 {'news_id': 4,
  'period': 1,
  'tick': 103,
  'ticker': '',
  'headline': 'Private Information #2 for GEM',
  'body': 'After 104 seconds, your private estimate is that the final value will be $26.54'},
 {'news_id': 3,
  'period': 1,
  'tick': 74,
  'ticker': '',
  'headline': 'Private Information #1 for UB',
  'body': 'After 75 seconds, your private estimate is that the final value will be $54.91'},
 {'news_id': 2,
  'period': 1,
  'tick': 65,
  'ticker': '',
  'headline': 'Private Information #1 for GEM',
  'body': 'After 66 seconds, your private estimate is that the final value will be $23.61'},
 {'news_id': 1,
  'period': 1,
  'tick': 0,
  'ticker': '',
  'headline': 'Welcome to the Price Discovery 3 Case',
  'body': ''}]
```

Select the most newest one news from the dictionary , run

```
news_query[0]
news_query[0]
Out[162]:
{'news_id': 5,
 'period': 1,
 'tick': 104,
 'ticker': '',
 'headline': 'Welcome to the Alogithmic Market Making: Extensions case',
 'body': ''}
```

To Capture the market news dollar characters from the most newest one news from the dictionary, run:

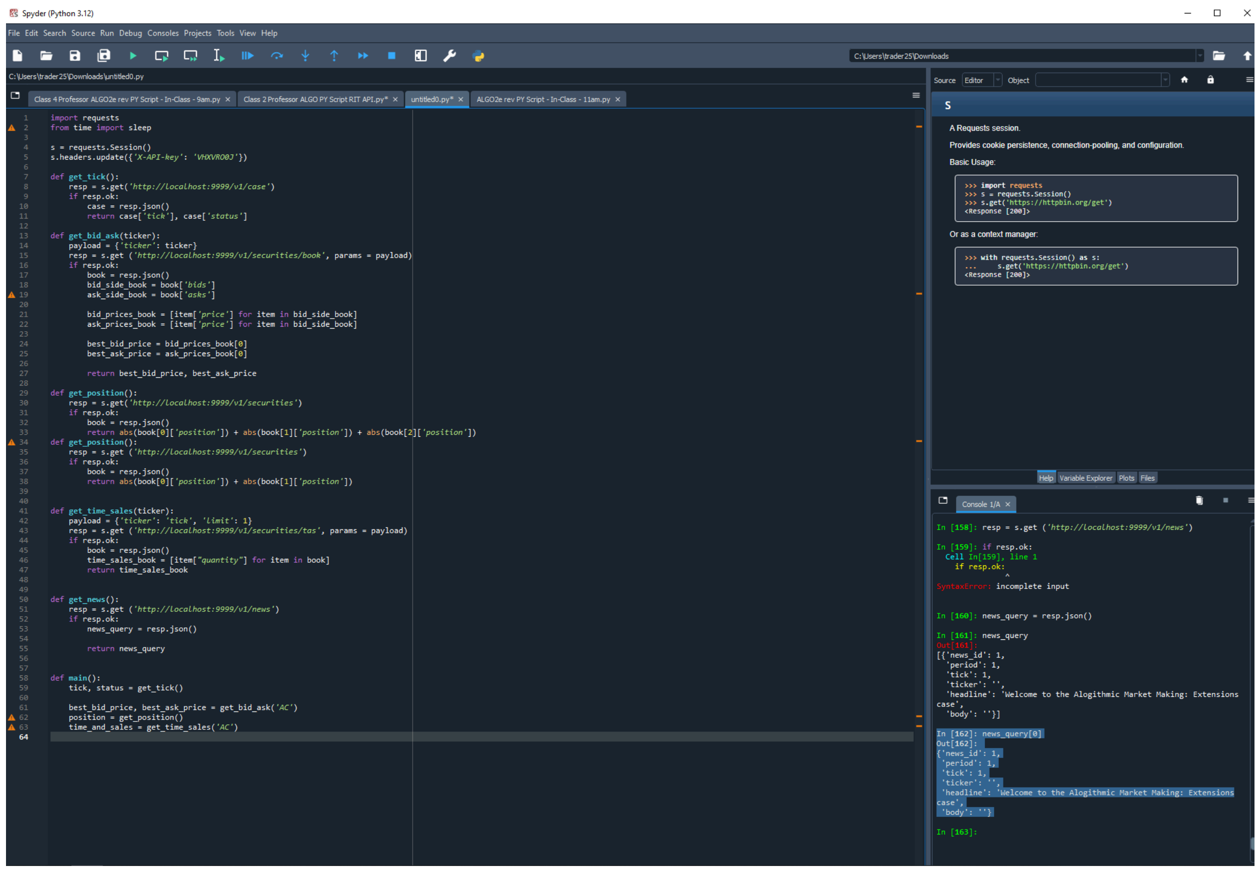
news_query[0][‘body’],find(‘$’)
73 ← the 73th character in the most lasted news sentences

To display the dollar amount in the news information, run:

news_query[0][‘body’][73:79]
58.41 ← this is useful, as I just parse the news to find the dollar amount using python, I use analyst information ‘58.41’ as a variable and use the variable in my algo’s decision making.

Video Evelen: Cancel Market Orders: just use cancel order function in the s.delect () under the main functions, Cancel order is important in ALOGO2e for the role as a Market Maker when trading inventories in the wrong position.

To be continued…
make an algo decision making with a new function defined in the support function for check the price trend whether is up or down , in Main function IF statement if up, then cancel buy orders,



Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\trader25\Downloads\untitled0.py
Class 4 Professor ALGO2e rev PY Script - In-Class - 9am.py
Class 2 Professor ALGO PY Script RIT API.py*
untitled0.py*
ALGO2e rev PY Script - In-Class - 11am.py
C:\Users\trader25\Downloads
Source Editor Object
Help Variable Explorer Plots Files
Console 1/A

```python
import requests
from time import sleep

s = requests.Session()
s.headers.update({'X-API-key': 'VHXVRO0J'})

def get_tick():
    resp = s.get('http://localhost:9999/v1/case')
    if resp.ok:
        case = resp.json()
        return case['tick'], case['status']

def get_bid_ask(ticker):
    payload = {'ticker': ticker}
    resp = s.get ('http://localhost:9999/v1/securities/book', params = payload)
    if resp.ok:
        book = resp.json()
        bid_side_book = book['bids']
        ask_side_book = book['asks']

        bid_prices_book = [item['price'] for item in bid_side_book]
        ask_prices_book = [item['price'] for item in bid_side_book]

        best_bid_price = bid_prices_book[0]
        best_ask_price = ask_prices_book[0]

        return best_bid_price, best_ask_price

def get_position():
    resp = s.get('http://localhost:9999/v1/securities')
    if resp.ok:
        book = resp.json()
        return abs(book[0]['position']) + abs(book[1]['position']) + abs(book[2]['position'])
def get_position():
    resp = s.get ('http://localhost:9999/v1/securities')
    if resp.ok:
        book = resp.json()
        return abs(book[0]['position']) + abs(book[1]['position'])


def get_time_sales(ticker):
    payload = {'ticker': 'tick', 'limit': 1}
    resp = s.get ('http://localhost:9999/v1/securities/tas', params = payload)
    if resp.ok:
        book = resp.json()
        time_sales_book = [item["quantity"] for item in book]
        return time_sales_book


def get_news():
    resp = s.get ('http://localhost:9999/v1/news')
    if resp.ok:
        news_query = resp.json()

        return news_query


def main():
    tick, status = get_tick()

    best_bid_price, best_ask_price = get_bid_ask('AC')
    position = get_position()
    time_and_sales = get_time_sales('AC')

```

S

A Requests session.

Provides cookie persistence, connection-pooling, and configuration.

Basic Usage:

```
>>> import requests
>>> s = requests.Session()
>>> s.get('https://httpbin.org/get')
<Response [200]>
```

Or as a context manager:

```
>>> with requests.Session() as s:
...     s.get('https://httpbin.org/get')
<Response [200]>
```

```
In [158]: resp = s.get ('http://localhost:9999/v1/news')

In [159]: if resp.ok:
  Cell In[159], line 1
    if resp.ok:
               ^
SyntaxError: incomplete input


In [160]: news_query = resp.json()

In [161]: news_query
Out[161]:
[{'news_id': 1,
  'period': 1,
  'tick': 1,
  'ticker': '',
  'headline': 'Welcome to the Alogithmic Market Making: Extensions
case',
  'body': ''}]

In [162]: news_query[0]
Out[162]:
{'news_id': 1,
 'period': 1,
 'tick': 1,
 'ticker': '',
 'headline': 'Welcome to the Alogithmic Market Making: Extensions
case',
 'body': ''}

In [163]:
```