

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light greenish-blue. They are positioned diagonally, with the blue one partially covering the green one.

Operating Systems and Stock Market Trading

Max Georgeson



Learning Outcomes

- Understand how operating systems aid in modern, online stock market trading
- Understand, at a very basic level, how to use Alpaca's and TD Ameritrade's trading APIs



Online Trading

- What it is and how it works
 - A way to trade shares (and more) of a company online
 - Robinhood, Schwab, TD Ameritrade, ETrade, etc.
- Algo/bot trading
 - Easy patterns that are boring
 - More calculated patterns
 - Removal of human element



SAAS vs. PAAS

- Software as a Service
 - Offering access to centrally hosted software
- Platform as a Service
 - Environments with development tools
- Many trading companies offer SAAS Models
 - Smaller tech companies
 - Alpaca, Interactive Brokers, Robinhood, Quantconnect, Webull
 - Many major banks
 - Merrill, Charles Schwab, ETrade
- Others offer a hybrid or both
 - TD Ameritrade, Alpaca Broker



APIs

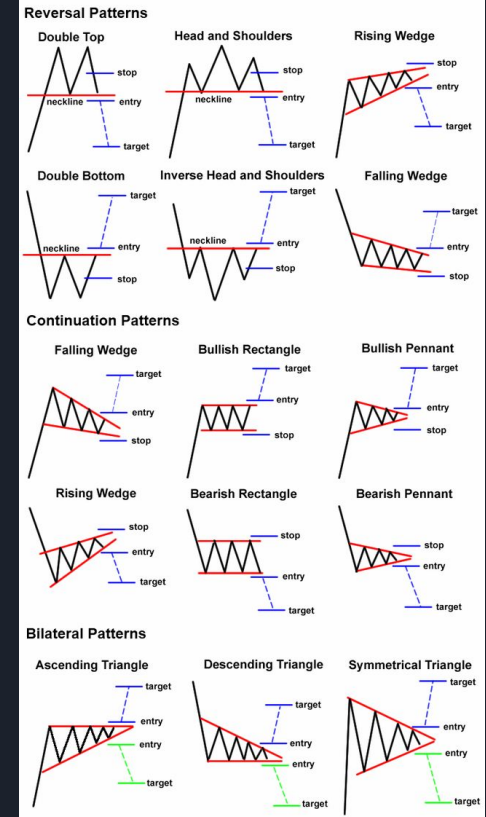
- Either way, you need an API to interact with many of them
- An API is an Application Programming Interface
 - Software interface between two systems or computers
 - Between your browser/app and the brokerage's system
 - Allows you to make requests regarding actions you need without having to understand the intricacies of their system
 - Can just send an order without knowing how exactly it executes
 - Can request data without knowing how to actually fetch it
 - Formats outputs and responses to work on your computer
 - Connect using an API "key"
 - A way to identify and authenticate users

API Benefits

- API benefits
 - Can use to buy or sell manually or with bots
 - Can pull real-time and historic data
 - Process locally however you'd like
 - Can submit trades based on market conditions
 - Chart Patterns
 - Doji Candles



<https://www.dailyfx.com/education/candlestick-patterns/types-of-doji.html>



https://www.reddit.com/r/Daytrading/comments/6dfdhy/here_are_some_chart_patterns_to_keep_in_the_back/

Alpaca Introduction

- “Alpaca is a technology company headquartered in California Silicon Valley that builds an API-first stock brokerage platform”
- Focused on trading with APIs rather than apps
- Commission free
- Offers real time trading, paper trading, sandbox environment and example algorithms



Alpaca Example

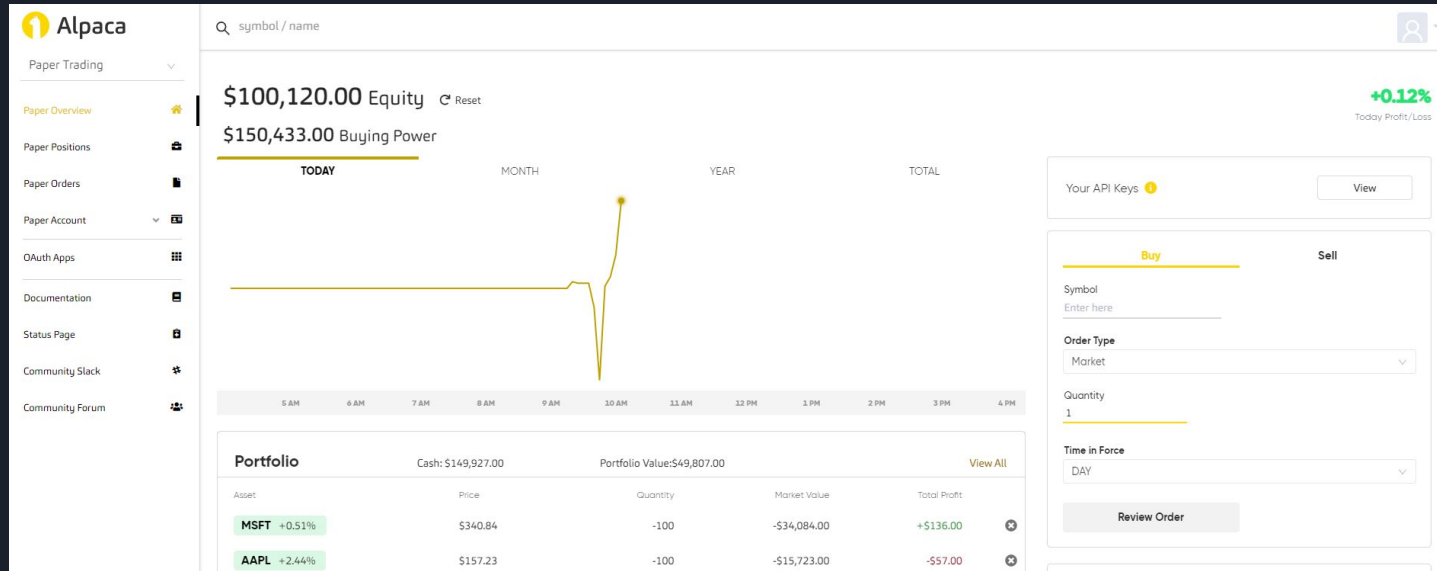
```
1 import requests, json, time
2
3 API_KEY_ID = "
4 SECRET_KEY = "
5
6 BASE_URL = "https://paper-api.alpaca.markets"
7 ACCOUNT_URL = "{}/v2/account".format(BASE_URL)
8 ORDERS_URL = "{}/v2/orders".format(BASE_URL)
9 HEADERS = {'APCA-API-KEY-ID': API_KEY_ID, 'APCA-API-SECRET-KEY': SECRET_KEY}
10
11 def get_account():
12     r = requests.get(ACCOUNT_URL, headers=HEADERS)
13
14     return json.loads(r.content)
15
16
17 def create_order(symbol, qty, side, type, time_in_force):
18     data = {
19         "symbol": symbol,
20         "qty": qty,
21         "side": side,
22         "type": type,
23         "time_in_force": time_in_force
24     }
25
26     r = requests.post(ORDERS_URL, json=data, headers=HEADERS)
27
28     return json.loads(r.content)
29
30
31 create_order("CZR", 100, "buy", "market", "gtc")
32 create_order("RIVN", 100, "buy", "market", "gtc")
33
34 account_info = get_account()
35 print("Shares purchased. Remaining buying power: ", account_info.get("buying_power"), account_info.get("currency"))
36
37 time.sleep(240)
38
39 create_order("CZR", 100, "sell", "market", "gtc")
40 create_order("RIVN", 100, "sell", "market", "gtc")
41
42 account_info = get_account()
43 print("Shares sold. Remaining buying power: ", account_info.get("buying_power"), account_info.get("currency"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Shares purchased. Remaining buying power: 127815 USD
Shares sold. Remaining buying power: 150064
PS D:\Max\Documents\AlgoTrading\AlgoTrading> █
```

- Buys shares, waits 4 minutes, then sells them
- Can also use API to pull real-time market data and trade based on that
 - Doji Candles

Alpaca Results



Order History

Asset	Order	Quantity	Average Cost	Notional	Amount	Status
RIVN	Market SELL 11/18/2021 09:41 AM	100	\$123.06	\$12,306.00	\$12,306.00	Filled
CZR	Market SELL 11/18/2021 09:41 AM	100	\$98.50	\$9,850.00	\$9,850.00	Filled
RIVN	Market BUY 11/18/2021 09:37 AM	100	\$122.83	\$12,283.00	\$12,283.00	Filled
CZR	Market BUY 11/18/2021 09:37 AM	100	\$98.39	\$9,839.00	\$9,839.00	Filled
----	Market SELL	----	-----	-----	-----	-----

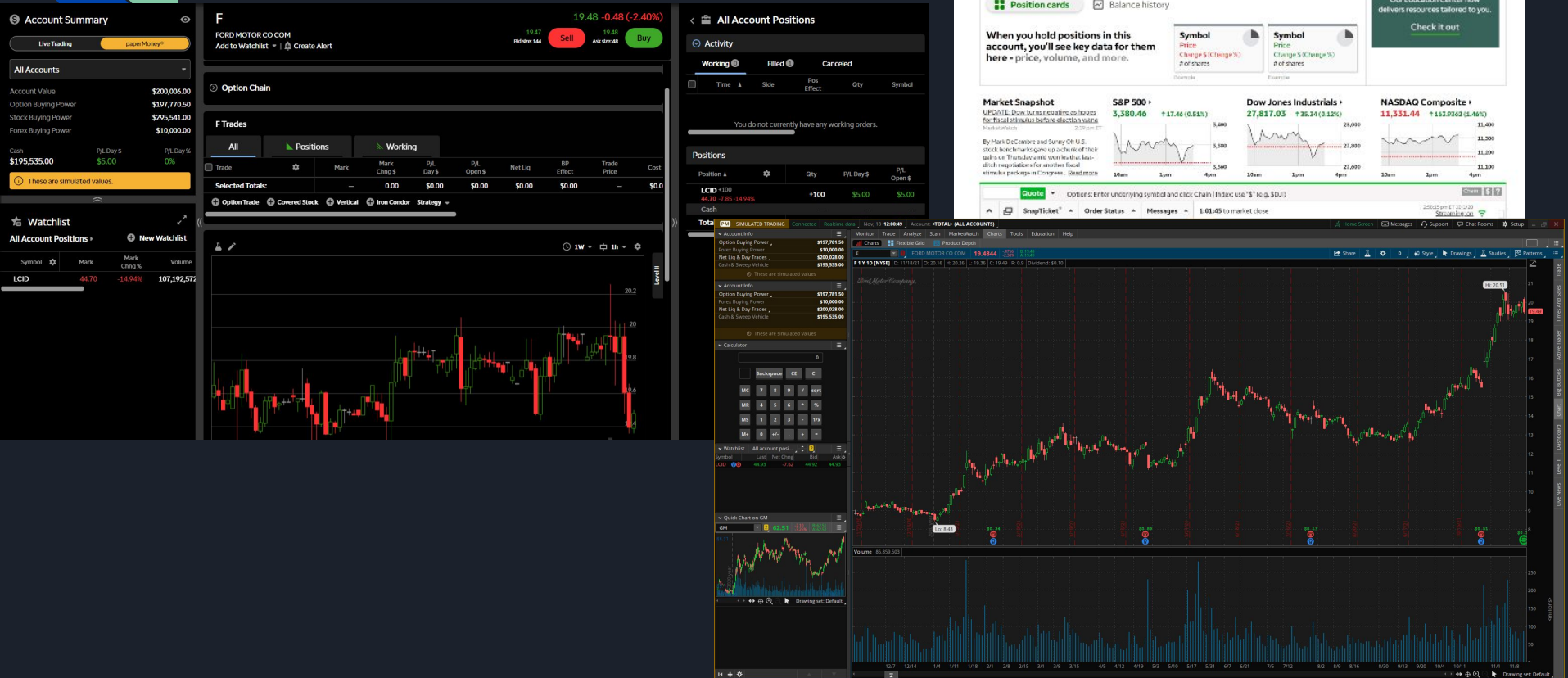


TD Ameritrade Introduction

- Large online broker
 - Stocks, option, futures, forex, crypto, mutual funds, IRAs, etc.
 - Now owned by Charles Schwab
- How It's used
 - Website
 - Think or Swim
 - App vs. Online
 - Developer tools
 - PAAS and API

TD Ameritrade Examples

- Dashboard vs. ToS online vs. ToS website



My Apps

These are your apps! Explore them!

[new](#) [OSDemo](#)

Approved

TDExampleForOS > OS_Example.py > ...

```

1 import requests
2
3 key = ' '
4
5 def get_price_history(**kwargs):
6
7     url = 'https://api.tdameritrade.com/v1/marketdata/{}/pricehistory'.format(kwargs.get('symbol'))
8
9     params = {}
10    params.update({'apikey': key})
11
12    for arg in kwargs:
13        parameter = {arg: kwargs.get(arg)}
14        params.update(parameter)
15
16    return requests.get(url, params=params).json()
17
18 print(get_price_history(symbol='CZK', period=1, periodType='day', frequencyType='minute'))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
'high': 101.07, 'low': 100.92, 'close': 101.083, 'volume': 2100, 'datetime': 1637161280000}, {'open': 100.98, 'high': 101.03, 'low': 100.88, 'volume': 8000}, {'open': 100.935, 'high': 101.09, 'low': 100.89, 'close': 101.05, 'volume': 3622, 'datetime': 1637162040000}, {'open': 101.065, 'high': 101.12, 'low': 101.01, 'close': 101.08, 'volume': 10000, 'datetime': 1637162180000}, {'open': 101.15, 'high': 101.2, 'low': 100.97, 'close': 100.97, 'volume': 3600, 'datetime': 1637162160000}, {'open': 100.99, 'high': 101.05, 'low': 100.95, 'close': 100.99, 'volume': 2852, 'datetime': 1637162220000}, {'open': 101.03, 'high': 101.05, 'low': 100.97, 'close': 101.05, 'volume': 900, 'datetime': 1637162280000}, {'open': 101.07, 'high': 101.09, 'low': 100.95, 'close': 101.07, 'volume': 3355, 'datetime': 1637162340000}, {'open': 101.08, 'high': 101.05, 'low': 100.94, 'close': 101.0, 'volume': 10000, 'datetime': 1637162400000}, {'open': 101.01, 'high': 101.01, 'close': 101.0428, 'volume': 2448, 'datetime': 1637162460000}, {'open': 101.08, 'high': 101.08, 'low': 100.91, 'close': 101.0, 'volume': 10000, 'datetime': 1637162520000}, {'open': 101.11, 'high': 101.12, 'low': 101.03, 'close': 101.06, 'volume': 745, 'datetime': 1637162580000}, {'open': 101.12, 'high': 101.12, 'close': 101.12, 'volume': 10000, 'datetime': 1637162640000}, {'open': 101.27, 'high': 101.4, 'low': 101.23, 'close': 101.4, 'volume': 1300, 'datetime': 1637162700000}, {'open': 101.4, 'high': 101.4, 'close': 101.4, 'volume': 10000, 'datetime': 1637162760000}, {'open': 101.44, 'high': 101.57, 'low': 101.44, 'close': 101.45, 'volume': 2023, 'datetime': 1637162820000}
```

Based on code from <https://github.com/robswc>

[Price History Documentation](#) / [Get Price History](#)[GET](#) Get Price History[Get price history for a symbol](#)

Resource URL

<https://api.tdameritrade.com/v1/marketdata/{CZR}/pricehistory>

Query Parameters

Name	Values	Description
apikey	<input type="text" value=""/> Select App	Pass your OAuth User ID to make an unauthenticated request for delayed data.
periodType	<input type="text" value="day"/>	The type of period to show. Valid values are <code>day</code> , <code>month</code> , <code>year</code> , or <code>ytd</code> (year to date). Default is <code>day</code> .
period	<input type="text" value="4"/>	The number of periods to show.

```
period : 2
periodType : day
frequency : 1
frequencyType : min
```

Valid periods by periodType (defaults marked with an asterisk):

```
day : 1, 2, 3, 4, 5, 10*
month : 1*, 2, 3, 6
year : 1*, 2, 3, 5, 10, 15, 20
unit : $
```

Resource Summary

Security	OAuth 2.0
Content Type	application/json
Category	Price History

Response Summary

JSON

```
//CandleList:
{
  "candles": [
    {
      "close": 0,
      "datetime": 0,
      "high": 0,
      "low": 0,
      "open": 0,
      "volume": 0
    }
  ],
  "empty": false,
  "symbol": "string"
}
```

Resource Error Codes

HTTP Code	Description
401	Unauthorized
403	Forbidden
404	Not Found

Try it out !!

SEND

RESF

Request:

Response

cURL

HTTP/1.1 200

```
Access-Control-Allow-Headers: origin
Access-Control-Allow-Methods: GET
Access-Control-Allow-Origin: https://developer.tamertamtrade.com
Access-Control-Max-Age: 3628800
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Security-Policy: frame-ancestors 'self'
Content-Type: application/json
Date: Thu, 18 Nov 2021 20:29:06 GMT
Strict-Transport-Security: max-age=31536000; includeSubDomains
Transfer-Encoding: chunked
vary: accept-encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
```

```
{
  "candles": [
    {
      "open": 103,
      "high": 103,
      "low": 103,
      "close": 103,
      "volume": 100,
      "datetime": 1637151840000
    },
    {
      "open": 103.35,
      "high": 103.35,

```



Conclusion

- SAAS, PAAS, and APIs are the backbone of modern investing
- They make the stock market available to everyone from an 18 year old dipping their feet into Robinhood to giant hedge fund running trading algorithms
- You can use websites, apps, and web hosted apps at any level of complexity you desire