Day-0

Today I will dive into **yfinance API**, an open source library that provides access to financial information from Yahoo Finance. It has its own pros and cons. It is free and easy to use but since it is not official it is not guaranteed its future maintainance.

> Source : https://algotrading101.com/learn/yfinance-guide/

In [20]:
```
# pip install yfinance
import yfinance as yf

# other useful libraries
import pandas as pd
import numpy as np
```

It is based on three modules:

- Tickers
- download
- pandas_datareader

In order to gain historical data, there is the **history()** method with the following parameters

In [21]:
```
yf.Tickers.history
```

Out[21]:
```
<function yfinance.tickers.Tickers.history(self, period='1mo', interval='1d', start=None, end=None, prepost=False, ac
tions=True, auto_adjust=True, proxy=None, threads=True, group_by='column', progress=True, **kwargs)>
```

The least obvious parameters are:

- prepost : include pre and post regular market in result
- auto_adjust : Adjust Open/High/Low/Close prices
- actions : download stock dividends and stock splits events

N.B. date format is in British Format (yyyy-mm-dd)

In [36]:
```
amzn = yf.Ticker("aapl")

amzn_historical = amzn.history(start="2018-02-24", end="2021-02-24", interval="5d")
amzn_historical.dropna(inplace=True)
```

In [37]:
```
amzn_historical.sample(5)
```

Out[37]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2019-07-23	51.242569	51.353184	50.954962	51.335976	73420800.0	0.0000	0.0
2018-03-15	43.054043	43.473731	42.950329	43.090221	90975200.0	0.0000	0.0
2019-02-08	41.380897	41.789833	41.241318	41.728615	95280000.0	0.1825	0.0
2020-02-28	63.820297	69.067125	63.599505	67.814331	426884800.0	0.0000	0.0
2019-06-28	48.838498	49.040068	48.437822	48.651680	124442400.0	0.0000	0.0

Then, it is possible to extract other relevant information thanks to the **info()** method. Of course they provide a snapshot of the current situation. So, with a single call we cannot use them for making predictions, but if we collect these informations on a long period, they may become relevant.

In [38]:
```
# amzn.info
```

That's pretty much it! These are the basics of yfinance. Another great library is **ta** (Technical Analysis). It is extremely useful for trying to do some feature engineering on the historical data.

> - https://technical-analysis-library-in-python.readthedocs.io/en/latest/
> - https://towardsdatascience.com/technical-analysis-library-to-financial-datasets-with-pandas-python-4b2b390d3543

The library implements 32 indicators based mostly on the volume, volatility and trend. Actually the techinal alaysis in much more "Trading" oriented, so it will require a proper study, that I will surely do later on. So, let's briefly look at these indicators.

In [39]:
```
# pip install ta
from ta import add_all_ta_features
```

In [40]:
```
df = add_all_ta_features(amzn_historical, open="Open", high="High", low="Low", close="Close", volume="Volume", filln
a=True)
df.columns
```

Out[40]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Dividends', 'Stock Splits',
       'volume_adi', 'volume_obv', 'volume_cmf', 'volume_fi', 'volume_mfi',
       'volume_em', 'volume_sma_em', 'volume_vpt', 'volume_nvi', 'volume_vwap',
       'volatility_atr', 'volatility_bbm', 'volatility_bbh', 'volatility_bbl',
       'volatility_bbw', 'volatility_bbp', 'volatility_bbhi',
       'volatility_bbli', 'volatility_kcc', 'volatility_kch', 'volatility_kcl',
       'volatility_kcw', 'volatility_kcp', 'volatility_kchi',
       'volatility_kcli', 'volatility_dcl', 'volatility_dch', 'volatility_dcm',
       'volatility_dcw', 'volatility_dcp', 'volatility_ui', 'trend_macd',
       'trend_macd_signal', 'trend_macd_diff', 'trend_sma_fast',
       'trend_sma_slow', 'trend_ema_fast', 'trend_ema_slow', 'trend_adx',
       'trend_adx_pos', 'trend_adx_neg', 'trend_vortex_ind_pos',
       'trend_vortex_ind_neg', 'trend_vortex_ind_diff', 'trend_trix',
       'trend_mass_index', 'trend_cci', 'trend_dpo', 'trend_kst',
       'trend_kst_sig', 'trend_kst_diff', 'trend_ichimoku_conv',
       'trend_ichimoku_base', 'trend_ichimoku_a', 'trend_ichimoku_b',
       'trend_visual_ichimoku_a', 'trend_visual_ichimoku_b', 'trend_aroon_up',
       'trend_aroon_down', 'trend_aroon_ind', 'trend_psar_up',
       'trend_psar_down', 'trend_psar_up_indicator',
       'trend_psar_down_indicator', 'trend_stc', 'momentum_rsi',
       'momentum_stoch_rsi', 'momentum_stoch_rsi_k', 'momentum_stoch_rsi_d',
       'momentum_tsi', 'momentum_uo', 'momentum_stoch',
       'momentum_stoch_signal', 'momentum_wr', 'momentum_ao', 'momentum_kama',
       'momentum_roc', 'momentum_ppo', 'momentum_ppo_signal',
       'momentum_ppo_hist', 'others_dr', 'others_dlr', 'others_cr'],
      dtype='object')
```

In [41]:
```
df.sample(5)
```

Out[41]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits	volume_adi	volume_obv	volume_cmf	...	momentum_wr	momentu
2019-01-29	38.097840	38.556234	37.576052	37.715031	166348800.0	0.0	0.0	2.368909e+08	-5.293980e+08	-0.031956	...	-86.125034	-10.0
2020-01-09	76.038715	76.828207	75.781331	76.630219	170108400.0	0.0	0.0	9.411074e+08	9.241112e+08	0.259054	...	-0.920573	14.2
2018-11-15	45.934413	46.807311	45.571110	46.670769	185915200.0	0.0	0.0	1.987587e+08	6.646800e+06	0.021380	...	-89.190406	2.8
2019-07-18	50.146237	50.608370	50.072492	50.554291	74162400.0	0.0	0.0	1.669988e+08	-2.451056e+08	-0.032523	...	-5.400213	4.5
2020-12-24	131.124063	133.260869	130.904390	131.773087	54930100.0	0.0	0.0	1.555994e+09	3.162333e+09	0.084264	...	-5.736207	16.8

5 rows × 90 columns

In [42]:
```
import plotly.offline as pyo
import plotly.graph_objects as go
# Set notebook mode to work in offline
pyo.init_notebook_mode()
```

In [43]:
```
fig = go.Figure()

# Add traces
fig.add_trace(go.Scatter(x=df.index, y=df['Close'], name='Close'))
fig.add_trace(go.Scatter(x=df.index, y=df['Open'], name='Open'))
fig.show()
```