Extracting Stock Data Using a Python Library

A company's stock share is a piece of the company more precisely:

A stock (also known as equity) is a security that represents the ownership of a fraction of a corporation. This entitles the owner of the stock to a proportion of the corporation's assets and profits equal to how much stock they own. Units of stock are called "shares." [1]

An investor can buy a stock and sell it later. If the stock price increases, the investor profits, If it decreases,the investor with incur a loss. Determining the stock price is complex; it depends on the number of outstanding shares, the size of the company's future profits, and much more. People trade stocks throughout the day the stock ticker is a report of the price of a certain stock, updated continuously throughout the trading session by the various stock market exchanges.

You are a data scientist working for a hedge fund; it's your job to determine any suspicious stock activity. In this lab you will extract stock data using a Python library. We will use the yfinance library, it allows us to extract data for stocks returning data in a pandas dataframe. You will use the lab to extract.

Table of Contents

Estimated Time Needed: **30 min**

In [1]:
```
!pip install yfinance==0.1.67
#!pip install pandas==1.3.3
```

```
Collecting yfinance==0.1.67
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
yfinance==0.1.67) (1.3.4)
Requirement already satisfied: requests>=2.20 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (fro
m yfinance==0.1.67) (2.26.0)
Requirement already satisfied: lxml>=4.5.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from y
finance==0.1.67) (4.6.4)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.10.tar.gz (8.2 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.15 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from y
finance==0.1.67) (1.21.4)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packa
ges (from pandas>=0.24->yfinance==0.1.67) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pandas>=0.24->yfinance==0.1.67) (2021.3)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages
(from requests>=2.20->yfinance==0.1.67) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packag
es (from requests>=2.20->yfinance==0.1.67) (1.26.7)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (3.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-pa
ckages (from requests>=2.20->yfinance==0.1.67) (2.0.8)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from pyth
on-dateutil>=2.7.3->pandas>=0.24->yfinance==0.1.67) (1.16.0)
Building wheels for collected packages: multitasking
  Building wheel for multitasking (setup.py) ... done
  Created wheel for multitasking: filename=multitasking-0.0.10-py3-none-any.whl size=8500 sha256=8a08999782660c405e2
4b712bb0fea1f4b7e7e876fca85394daa878c1fd28140
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/34/ba/79/c0260c6f1a03f420ec7673eff9981778f293b9107974679e3
6
Successfully built multitasking
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.10 yfinance-0.1.67
```

In [3]:
```
import yfinance as yf
import pandas as pd
```

Using the yfinance Library to Extract Stock Data

Using the `Ticker` module we can create an object that will allow us to access functions to extract data. To do this we need to provide the ticker symbol for the stock, here the company is Apple and the ticker symbol is `AAPL`.

In [4]:
```
apple = yf.Ticker("AAPL")
```

Now we can access functions and variables to extract the type of data we need. You can view them and what they represent here https://aroussi.com/post/python-yahoo-finance (https://aroussi.com/post/python-yahoo-finance?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0220ENSkillsNetwork23455606-2021-01-01).

Stock Info

Using the attribute `info` we can extract information about the stock as a Python dictionary.

In [5]:
```
apple_info=apple.info
apple_info
```

```
ed in 1977 and is headquartered in Cupertino, California.',
 'city': 'Cupertino',
 'phone': '408 996 1010',
 'state': 'CA',
 'country': 'United States',
 'companyOfficers': [],
 'website': 'https://www.apple.com',
 'maxAge': 1,
 'address1': 'One Apple Park Way',
 'industry': 'Consumer Electronics',
 'ebitdaMargins': 0.32867,
 'profitMargins': 0.25882,
 'grossMargins': 0.41779,
 'operatingCashflow': 104037998592,
 'revenueGrowth': 0.288,
 'operatingMargins': 0.29782,
 'ebitda': 120233000960,
 'targetLowPrice': 128.01,
 'recommendationKey': 'buy',
 'grossProfits': 152836000000,
```

We can get the `'country'` using the key country

In [6]:
```
apple_info['country']
```

Out[6]:
```
'United States'
```

Extracting Share Price

A share is the single smallest part of a company's stock that you can buy, the prices of these shares fluctuate over time. Using the `history()` method we can get the share price of the stock over a certain period of time. Using the `period` parameter we can set how far back from the present to get data. The options for `period` are 1 day (1d), 5d, 1 month (1mo) , 3mo, 6mo, 1 year (1y), 2y, 5y, 10y, ytd, and max.

In [8]:
```
apple_share_price_data = apple.history(period="max")
```

The format that the data is returned in is a Pandas DataFrame. With the `Date` as the index the share `Open` , `High` , `Low` , `Close` , `Volume` , and `Stock Splits` are given for each day.

In [9]:
```
apple_share_price_data.head()
```

Out[9]:

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
1980-12-12	0.100453	0.100890	0.100453	0.100453	469033600	0.0	0.0
1980-12-15	0.095649	0.095649	0.095213	0.095213	175884800	0.0	0.0
1980-12-16	0.088661	0.088661	0.088224	0.088224	105728000	0.0	0.0
1980-12-17	0.090408	0.090845	0.090408	0.090408	86441600	0.0	0.0
1980-12-18	0.093029	0.093466	0.093029	0.093029	73449600	0.0	0.0

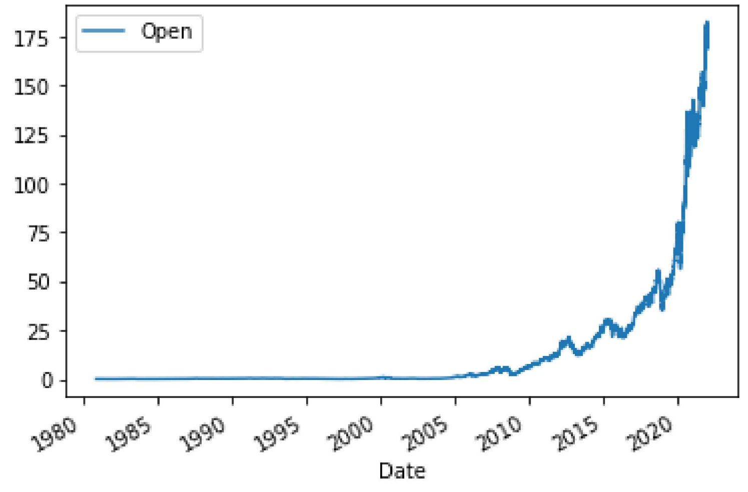
We can reset the index of the DataFrame with the `reset_index` function. We also set the `inplace` paramter to `True` so the change takes place to the DataFrame itself.

In [10]:
```
apple_share_price_data.reset_index(inplace=True)
```

We can plot the `Open` price against the `Date` :

```
In [11]: apple_share_price_data.plot(x="Date", y="Open")
```

```
Out[11]: <AxesSubplot:xlabel='Date'>
```



Extracting Dividends

Dividends are the distribution of a companys profits to shareholders. In this case they are defined as an amount of money returned per share an investor owns. Using the variable `dividends` we can get a dataframe of the data. The period of the data is given by the period defined in the 'history` function.

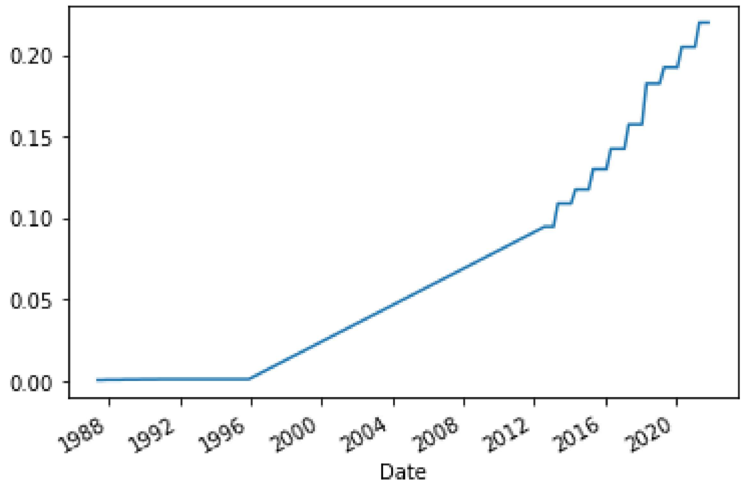
In [12]:
```
apple.dividends
```

Out[12]:
```
Date
1987-05-11    0.000536
1987-08-10    0.000536
1987-11-17    0.000714
1988-02-12    0.000714
1988-05-16    0.000714
                ...
2020-11-06    0.205000
2021-02-05    0.205000
2021-05-07    0.220000
2021-08-06    0.220000
2021-11-05    0.220000
Name: Dividends, Length: 73, dtype: float64
```

We can plot the dividends overtime:

In [13]:
```
apple.dividends.plot()
```

Out[13]:
```
<AxesSubplot:xlabel='Date'>
```



Exercise

Now using the `Ticker` module create an object for AMD (Advanced Micro Devices) with the ticker symbol is `AMD` called; name the object `amd`.

In [16]:
```
amd = yf.Ticker("AMD")
```

Question 1 Use the key `'country'` to find the country the stock belongs to, remember it as it will be a quiz question.

In [20]:
```
amd_info = amd.info
amd_info
amd_info['country']
```

Out[20]:
```
'United States'
```

Question 2 Use the key `'sector'` to find the sector the stock belongs to, remember it as it will be a quiz question.

In [21]:
```
amd_info = amd.info
amd_info
amd_info['sector']
```

Out[21]:
```
'Technology'
```

Question 3 Obtain stock data for AMD using the `history` function, set the `period` to max. Find the `Volume` traded on the first day (first row).

In [58]:
```
amd_share_price_data = amd.history(period="max")
amd_share_price_data
```

Out[58]:

	Open	High	Low	Close	Volume	Dividends	Stock Splits
Date							
1980-03-17	0.000000	3.302083	3.125000	3.145833	219600	0	0.0
1980-03-18	0.000000	3.125000	2.937500	3.031250	727200	0	0.0
1980-03-19	0.000000	3.083333	3.020833	3.041667	295200	0	0.0
1980-03-20	0.000000	3.062500	3.010417	3.010417	159600	0	0.0
1980-03-21	0.000000	3.020833	2.906250	2.916667	130800	0	0.0
...	...	...	...	...	...	...	...
2022-01-04	151.009995	152.419998	140.699997	144.419998	80200500	0	0.0
2022-01-05	142.820007	143.759995	135.289993	136.149994	65403200	0	0.0
2022-01-06	136.190002	138.000000	131.770004	136.229996	64802900	0	0.0
2022-01-07	136.279999	137.440002	131.130005	132.000000	58398000	0	0.0
2022-01-10	129.080002	132.419998	125.029999	132.000000	84592100	0	0.0

10546 rows × 7 columns

In [32]:
```
amd_info = amd.info
amd_info
```

Out[32]:
```
{'zip': '95054',
 'sector': 'Technology',
 'fullTimeEmployees': 12600,
 'longBusinessSummary': 'Advanced Micro Devices, Inc. operates as a semiconductor company worldwide. The company o
perates in two segments, Computing and Graphics; and Enterprise, Embedded and Semi-Custom. Its products include x8
6 microprocessors as an accelerated processing unit, chipsets, discrete and integrated graphics processing units
(GPUs), data center and professional GPUs, and development services; and server and embedded processors, and semi-
custom System-on-Chip (SoC) products, development services, and technology for game consoles. The company provides
x86 microprocessors for desktop PCs under the AMD Ryzen, AMD Ryzen PRO, Ryzen, Threadripper, AMD A-Series, AMD FX,
AMD Athlon, AMD Athlon PRO, and AMD Pro A-Series processors brands; microprocessors for notebook and 2-in-1s under
the AMD Ryzen, AMD A-Series, AMD Athlon, AMD Ryzen PRO, AMD Athlon PRO, and AMD Pro A-Series processors brands; mi
croprocessors for servers under the AMD EPYC and AMD Opteron brands; and chipsets under the AMD trademark. It also
offers discrete GPUs for desktop and notebook PCs under the AMD Radeon graphics and AMD Embedded Radeon brands; pr
ofessional graphics products under the AMD Radeon Pro and AMD FirePro graphics brands; and Radeon Instinct and AMD
Instinct accelerators for servers. In addition, the company provides embedded processor solutions under the AMD Op
teron, AMD Athlon, AMD Geode, AMD Ryzen, AMD EPYC, AMD R-Series, and G-Series processors brands; and customer-spec
ific solutions based on AMD CPU, GPU, and multi-media technologies, as well as semi-custom SoC products. It serves
original equipment manufacturers, public cloud service providers, original design manufacturers, system integrator
s, independent distributors, online retailers, and add-in-board manufacturers through its direct sales force, inde
```

About the Authors:

Joseph Santarcangelo (https://www.linkedin.com/in/joseph-s-50398b136/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0220ENSkillsNetwork23455606-2021-01-01) has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Azim Hirjani

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-11-10	1.1	Malika Singla	Deleted the Optional part
2020-08-27	1.0	Malika Singla	Added lab to GitLab
