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

- Using yfinance to Extract Stock Info
- Using yfinance to Extract Historical Share Price Data
- Using yfinance to Extract Historical Dividends Data
- Exercise

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) (1.3.5)
Requirement already satisfied: numpy>=1.15 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.
1.67) (1.21.6)
Requirement already satisfied: requests>=2.20 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance=
=0.1.67) (2.29.0)
Collecting multitasking>=0.0.7 (from yfinance==0.1.67)
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: lxml>=4.5.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance==0.
1.67) (4.9.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from p
andas>=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>=0.2
4->yfinance==0.1.67) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.20->yfinance==0.1.67) (3.1.0)
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) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from re
quests>=2.20->yfinance==0.1.67) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from reque
sts>=2.20->yfinance==0.1.67) (2022.12.7)
Requirement already satisfied: six>=1.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from python-dateutil
>=2.7.3->pandas>=0.24->yfinance==0.1.67) (1.16.0)
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.11 yfinance-0.1.67
```

In [9]:
```
import yfinance as yf
import pandas as pd
```

In [20]:
```
pip install yfinance --upgrade --no-cache-dir
```

```
Requirement already satisfied: yfinance in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (0.1.67)
Collecting yfinance
  Downloading yfinance-0.2.18-py2.py3-none-any.whl (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.3/60.3 kB 89.3 MB/s eta 0:00:00
Requirement already satisfied: pandas>=1.3.0 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance)
(1.3.5)
Requirement already satisfied: numpy>=1.16.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance)
(1.21.6)
Requirement already satisfied: requests>=2.26 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance)
(2.29.0)
Requirement already satisfied: multitasking>=0.0.7 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfin
ance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance)
(4.9.2)
Collecting appdirs>=1.4.4 (from yfinance)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Requirement already satisfied: pytz>=2022.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfinance)
(2023.3)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.3.8-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 165.1 MB/s eta 0:00:00
Requirement already satisfied: cryptography>=3.3.2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from yfin
ance) (38.0.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from y
finance) (4.11.1)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 kB 144.6 MB/s eta 0:00:00
Requirement already satisfied: soupsieve>1.2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from beautifuls
oup4>=4.11.1->yfinance) (2.3.2.post1)
Requirement already satisfied: cffi>=1.12 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from cryptography>
=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib>=1.1->
yfinance) (1.16.0)
Requirement already satisfied: webencodings in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib>=
1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from p
andas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
requests>=2.26->yfinance) (3.1.0)
```

```
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from requests>=
2.26->yfinance) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from re
quests>=2.26->yfinance) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from reque
sts>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: pycparser in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from cffi>=1.12->cr
yptography>=3.3.2->yfinance) (2.21)
Installing collected packages: appdirs, html5lib, frozendict, yfinance
  Attempting uninstall: yfinance
    Found existing installation: yfinance 0.1.67
    Uninstalling yfinance-0.1.67:
      Successfully uninstalled yfinance-0.1.67
Successfully installed appdirs-1.4.4 frozendict-2.3.8 html5lib-1.1 yfinance-0.2.18
Note: you may need to restart the kernel to use updated packages.
```

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 [21]:
```
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.

Stock Info

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

In [22]:
```
apple_info=apple.info
apple_info
```

Out[22]:
```
{'regularMarketPrice': None, 'preMarketPrice': None, 'logo_url': ''}
```

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

In [23]:
```
apple_info['country']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_83/1006765094.py in <module>
----> 1 apple_info['country']

KeyError: 'country'
```

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 [24]:
```
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 [25]:
```
apple_share_price_data.head()
```

Out[25]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1980-12-12	0.099722	0.100155	0.099722	0.099722	469033600	0.0	0.0
1980-12-15	0.094953	0.094953	0.094519	0.094519	175884800	0.0	0.0
1980-12-16	0.088015	0.088015	0.087582	0.087582	105728000	0.0	0.0
1980-12-17	0.089749	0.090183	0.089749	0.089749	86441600	0.0	0.0
1980-12-18	0.092351	0.092785	0.092351	0.092351	73449600	0.0	0.0

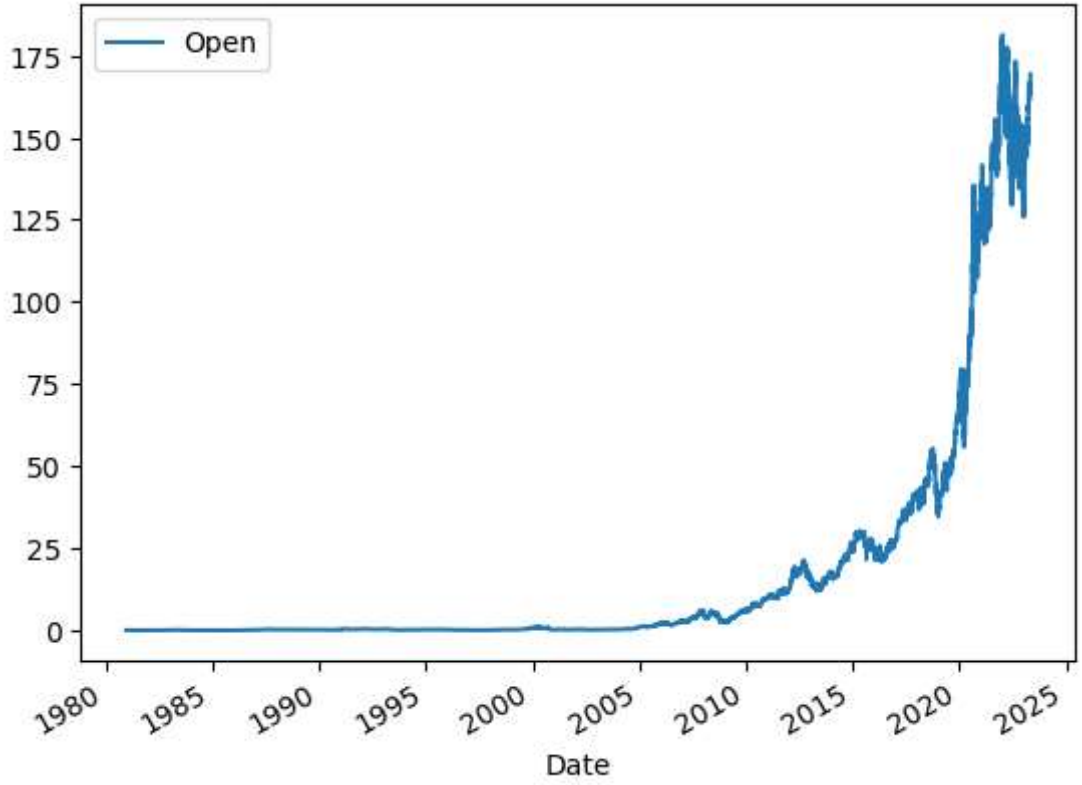
We can reset the index of the DataFrame with the `reset_index` function. We also set the `inplace` paramter to `True` so the change takes place to the DataFrame itself.

In [26]:
```
apple_share_price_data.reset_index(inplace=True)
```

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

In [27]:
```
apple_share_price_data.plot(x="Date", y="Open")
```

Out[27]:
```
<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.

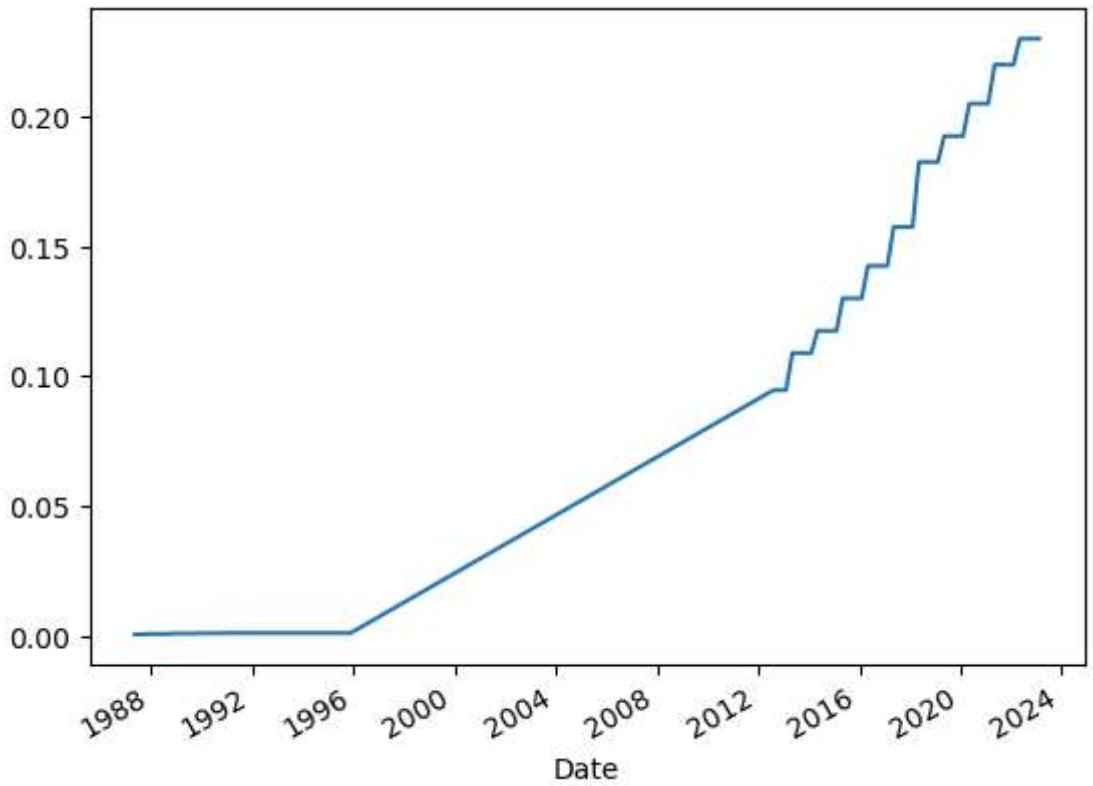
In [28]:
```
apple.dividends
```

Out[28]:
```
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
                 ...
2022-02-04    0.220000
2022-05-06    0.230000
2022-08-05    0.230000
2022-11-04    0.230000
2023-02-10    0.230000
Name: Dividends, Length: 78, dtype: float64
```

We can plot the dividends overtime:

In [29]:
```
apple.dividends.plot()
```

Out[29]:
```
<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 [30]:
```
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 [31]:
```
amd_info = amd.info
amd_info
amd_info['country']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_83/2517634827.py in <module>
      1 amd_info = amd.info
      2 amd_info
----> 3 amd_info['country']

KeyError: 'country'
```

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

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

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_83/2307009968.py in <module>
      1 amd_info = amd.info
      2 amd_info
----> 3 amd_info['sector']

KeyError: 'sector'
```

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 [33]:
```
amd_share_price_data = amd.history(period="max")
amd_share_price_data
```

Out[33]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
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
...	...	...	...	...	...	...	...
2023-04-25	86.900002	87.059998	83.760002	83.800003	52771500	0	0.0
2023-04-26	86.510002	87.349998	85.320000	85.940002	47786600	0	0.0
2023-04-27	86.410004	87.570000	84.089996	87.440002	53087200	0	0.0
2023-04-28	87.019997	89.750000	86.440002	89.370003	50652100	0	0.0
2023-05-01	91.029999	90.999901	89.190002	89.587997	16049959	0	0.0

10873 rows × 7 columns

In [34]:
```
amd_info = amd.info
amd_info
```

Out[34]:
```
{'regularMarketPrice': None, 'preMarketPrice': None, 'logo_url': ''}
```

About the Authors:

Joseph Santarcangelo 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

© IBM Corporation 2020. All rights reserved.

In [ ]: