Collecting data from Yahoo finance using yfinance

This tutorial will demonstrate how to use Python to retrieve financial data from Yahoo Finance. Using this, we may access historical market data as well as financial information about the company (for example, financial ratios).

Installation

```
!pip install yfinance
!pip install yahoofinancials
```

```
Collecting yfinance
  Downloading yfinance-0.1.74-py2.py3-none-any.whl (27 kB)
Requirement already satisfied: numpy>=1.15 in /shared-libs/python3.9/py/lib/py
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: lxml>=4.5.1 in /shared-libs/python3.9/py/lib/py
Requirement already satisfied: requests>=2.26 in /shared-libs/python3.9/py/lil
Requirement already satisfied: pandas>=0.24.0 in /shared-libs/python3.9/py/lil
Requirement already satisfied: pytz>=2017.3 in /shared-libs/python3.9/py/lib/
Requirement already satisfied: python-dateutil>=2.7.3 in /shared-libs/python3.
Requirement already satisfied: idna<4,>=2.5 in /shared-libs/python3.9/py-core/
Requirement already satisfied: charset-normalizer<3,>=2 in /shared-libs/pythor
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /shared-libs/python3.
Requirement already satisfied: certifi>=2017.4.17 in /shared-libs/python3.9/py
Requirement already satisfied: six>=1.5 in /shared-libs/python3.9/py-core/lib/
Installing collected packages: multitasking, yfinance
Successfully installed multitasking-0.0.11 yfinance-0.1.74
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is availab
You should consider upgrading via the '/root/venv/bin/python -m pip install -
Collecting yahoofinancials
  Downloading yahoofinancials-1.6.tar.gz (27 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: beautifulsoup4 in /shared-libs/python3.9/py-co
Requirement already satisfied: pytz in /shared-libs/python3.9/py/lib/python3.
Requirement already satisfied: soupsieve>1.2 in /shared-libs/python3.9/py-core
Building wheels for collected packages: yahoofinancials
  Building wheel for yahoofinancials (setup.py) ... done
  Created wheel for yahoofinancials: filename=yahoofinancials-1.6-py3-none-any
  Stored in directory: /root/.cache/pip/wheels/7d/59/6e/ff5fc25443eef95656f84l
Successfully built yahoofinancials
Installing collected packages: yahoofinancials
Successfully installed yahoofinancials-1.6
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is availab
You should consider upgrading via the '/root/venv/bin/python -m pip install -
```

Analysis

The yfinance package can be imported into Python programs once it has been installed. We must use the company's ticker as an example in our argument.

A security is given a specific set of letters called a ticker or a stock symbol for trading purposes. For instance:

For Amazon, it is “AMZN” For Facebook, it is “FB” For Google, it is “GOOGL” For Microsoft, it is "MSFT"

```
import yfinance as yahooFinance

# Here We are getting Google's financial information
GoogleInfo = yahooFinance.Ticker("GOOGL")
```

whole python dictionary is printed here

```
print(GoogleInfo.info)
```

```
{'zip': '94043', 'sector': 'Communication Services', 'fullTimeEmployees': 174
```

The print statement produces a Python dictionary, which we can analyze and use to get the specific financial data we're looking for from Yahoo Finance. Let's take a few financial critical metrics as an example.

The info dictionary contains all firm information. As a result, we may extract the desired elements from the dictionary by parsing it:

We can retrieve financial key metrics like Company Sector, Price Earnings Ratio, and Company Beta from the above dictionary of items easily. Let us see the below code.

```
# display Company Sector
print("Company Sector : ", GoogleInfo.info['sector'])

# display Price Earnings Ratio
print("Price Earnings Ratio : ", GoogleInfo.info['trailingPE'])

# display Company Beta
print(" Company Beta : ", GoogleInfo.info['beta'])
```

```
Company Sector :  Communication Services
Price Earnings Ratio :  1.6200992
 Company Beta :  1.078487
```

There are a ton of more stuff in the information. By printing the informational keys, we can view all of them:

```
# get all key value pairs that are available
for key, value in GoogleInfo.info.items():
    print(key, ":", value)
```

```
zip : 94043
sector : Communication Services
fullTimeEmployees : 174014
longBusinessSummary : Alphabet Inc. provides various products and platforms
city : Mountain View
phone : 650 253 0000
state : CA
country : United States
companyOfficers : []
website : https://www.abc.xyz
maxAge : 1
address1 : 1600 Amphitheatre Parkway
industry : Internet Content & Information
ebitdaMargins : 0.34834
profitMargins : 0.25892
grossMargins : 0.56744
operatingCashflow : 95001001984
revenueGrowth : 0.126
operatingMargins : 0.29648
ebitda : 96886996992
targetLowPrice : 113
recommendationKey : buy
grossProfits : 146698000000
freeCashflow : 51070373888
targetMedianPrice : 144
currentPrice : 107.85
earningsGrowth : -0.113
currentRatio : 2.809
returnOnAssets : 0.14927
numberOfAnalystOpinions : 46
targetMeanPrice : 144.96
debtToEquity : 11.28
returnOnEquity : 0.29216
targetHighPrice : 187.5
totalCash : 124997001216
totalDebt : 28810000384
totalRevenue : 278139011072
totalCashPerShare : 9.583
financialCurrency : USD
revenuePerShare : 21.03
quickRatio : 2.642
recommendationMean : 1.8
exchange : NMS
shortName : Alphabet Inc.
```

```
longName : Alphabet Inc.
exchangeTimezoneName : America/New_York
exchangeTimezoneShortName : EDT
isEsgPopulated : False
gmtOffSetMilliseconds : -14400000
quoteType : EQUITY
symbol : GOOGL
messageBoardId : finmb_29096
market : us_market
annualHoldingsTurnover : None
enterpriseToRevenue : 4.801
beta3Year : None
enterpriseToEbitda : 13.782
52WeekChange : -0.23653555
morningStarRiskRating : None
```

We can retrieve historical market prices too and display them. Additionally, we can utilize it to get earlier market data.

We will use historical Google stock values over the past few years as our example. It is a relatively easy assignment to complete, as demonstrated below:

```
# covering the past few years.
# max->maximum number of daily prices available
# for Google.
# Valid options are 1d, 5d, 1mo, 3mo, 6mo, 1y, 2y,
# 5y, 10y and ytd.
print(GoogleInfo.history(period="max"))
```

```
                  Open        High         Low       Close     Volume  \
Date
2004-08-19    2.502503    2.604104    2.401401    2.511011  893181924
2004-08-20    2.527778    2.729730    2.515015    2.710460  456686856
2004-08-23    2.771522    2.839840    2.728979    2.737738  365122512
2004-08-24    2.783784    2.792793    2.591842    2.624374  304946748
2004-08-25    2.626627    2.702703    2.599600    2.652653  183772044
...                ...         ...         ...         ...        ...
2022-08-29  109.989998  110.949997  108.800003  109.419998   21191200
2022-08-30  110.169998  110.500000  107.800003  108.940002   27513300
2022-08-31  110.650002  110.849998  108.129997  108.220001   28627000
2022-09-01  108.279999  110.449997  107.360001  109.739998   28360900
2022-09-02  110.589996  110.739998  107.261597  107.849998   23528231

            Dividends  Stock Splits
Date
2004-08-19          0           0.0
2004-08-20          0           0.0
2004-08-23          0           0.0
2004-08-24          0           0.0
2004-08-25          0           0.0
...               ...           ...
2022-08-29          0           0.0
2022-08-30          0           0.0
2022-08-31          0           0.0
2022-09-01          0           0.0
2022-09-02          0           0.0

[4543 rows x 7 columns]
```

We can pass our own start and end dates.

```
import datetime

start = datetime.datetime(2012,5,31)
end = datetime.datetime(2013,1,30)
print(GoogleInfo.history(start=start, end=end))
```

```
                 Open       High        Low      Close     Volume  Dividends
Date
2012-05-31  14.732733  14.764765  14.489489  14.536036  118613268          0
2012-06-01  14.309059  14.330581  14.222973  14.288789  122193684          0
2012-06-04  14.269770  14.526777  14.264515  14.479229   97210692          0
2012-06-05  14.400651  14.467718  14.175926  14.274525   93502404          0
2012-06-06  14.426426  14.563814  14.354605  14.528779   83748168          0
...               ...        ...        ...        ...        ...        ...
2013-01-23  18.418167  18.743744  18.413162  18.556055  236127636          0
2013-01-24  18.549549  18.939690  18.531281  18.874125  135172692          0
2013-01-25  18.788038  18.980982  18.775024  18.860611   88946964          0
2013-01-28  18.812813  18.908909  18.715965  18.787037   65018916          0
2013-01-29  18.687437  18.942694  18.682182  18.860861   69814116          0

            Stock Splits
Date
2012-05-31             0
2012-06-01             0
2012-06-04             0
2012-06-05             0
2012-06-06             0
...                  ...
2013-01-23             0
2013-01-24             0
2013-01-25             0
2013-01-28             0
2013-01-29             0

[166 rows x 7 columns]
```

We can simultaneously download historical prices for many stocks:

The code below Pandas DataFrame including the different price data for the requested stocks. We now select the individual stock by printing df.GOOGL to have the historical market data for Google:

```
df = yahooFinance.download("AMZN GOOGL", start="2019-01-01", end="2020-01-01",group
print(df)
print(df.GOOGL)
```

```
[*********************100%***********************]  2 of 2 completed
                 AMZN
                 Open       High        Low      Close  Adj Close     Volume
Date
2019-01-02  73.260002  77.667999  73.046501  76.956497  76.956497  159662000
2019-01-03  76.000504  76.900002  74.855499  75.014000  75.014000  139512000
2019-01-04  76.500000  79.699997  75.915497  78.769501  78.769501  183652000
2019-01-07  80.115501  81.727997  79.459503  81.475502  81.475502  159864000
2019-01-08  83.234497  83.830498  80.830498  82.829002  82.829002  177628000
...               ...        ...        ...        ...        ...        ...
2019-12-24  89.690498  89.778503  89.378998  89.460503  89.460503   17626000
2019-12-26  90.050499  93.523003  89.974998  93.438499  93.438499  120108000
2019-12-27  94.146004  95.070000  93.300499  93.489998  93.489998  123732000
2019-12-30  93.699997  94.199997  92.030998  92.344498  92.344498   73494000
2019-12-31  92.099998  92.663002  91.611504  92.391998  92.391998   50130000

                GOOGL
                 Open       High        Low      Close  Adj Close    Volume
Date
2019-01-02  51.360001  53.039501  51.264000  52.734001  52.734001  31868000
2019-01-03  52.533501  53.313000  51.118500  51.273499  51.273499  41960000
2019-01-04  52.127998  54.000000  51.842999  53.903500  53.903500  46022000
2019-01-07  54.048500  54.134998  53.132000  53.796001  53.796001  47446000
2019-01-08  54.299999  54.667500  53.417500  54.268501  54.268501  35414000
...               ...        ...        ...        ...        ...       ...
2019-12-24  67.510498  67.600502  67.208504  67.221497  67.221497  13468000
2019-12-26  67.327499  68.160004  67.275497  68.123497  68.123497  23662000
2019-12-27  68.199997  68.352501  67.650002  67.732002  67.732002  23212000
2019-12-30  67.840500  67.849998  66.891998  66.985497  66.985497  19994000
2019-12-31  66.789497  67.032997  66.606499  66.969498  66.969498  19514000

[252 rows x 12 columns]
                 Open       High        Low      Close  Adj Close    Volume
Date
2019-01-02  51.360001  53.039501  51.264000  52.734001  52.734001  31868000
2019-01-03  52.533501  53.313000  51.118500  51.273499  51.273499  41960000
2019-01-04  52.127998  54.000000  51.842999  53.903500  53.903500  46022000
2019-01-07  54.048500  54.134998  53.132000  53.796001  53.796001  47446000
2019-01-08  54.299999  54.667500  53.417500  54.268501  54.268501  35414000
...               ...        ...        ...        ...        ...       ...
2019-12-24  67.510498  67.600502  67.208504  67.221497  67.221497  13468000
2019-12-26  67.327499  68.160004  67.275497  68.123497  68.123497  23662000
2019-12-27  68.199997  68.352501  67.650002  67.732002  67.732002  23212000
2019-12-30  67.840500  67.849998  66.891998  66.985497  66.985497  19994000
2019-12-31  66.789497  67.032997  66.606499  66.969498  66.969498  19514000

[252 rows x 6 columns]
```

Save the data to CSV

```
df.to_csv('data/FinanceData.csv')
```

Congratulations!

Credit: This tutorial is prepared by Ajay Sadananda.