

Chapter 5: Collect and Analyze Financial Data

In this chapter, you'll learn how to obtain real-time financial information from Yahoo Finance using the firm's ticker symbol. You'll then learn to scrape the web to search for the ticker symbol of the company, based on the company name. This way, you can retrieve the live financial information based on company name.

In the second project, you'll obtain recent daily stock price information and create price plots or candlestick charts.

In the third project, you'll learn to use recent daily stock prices to calculate returns, run regressions, and perform detailed analyses. Once done, the program will tell you the alpha (abnormal return) and beta (market risk) of the stock you are interested in.

Project 1: Python, What's the Stock Price of Bank of America?

In this project, you'll learn to obtain the real-time stock price information (updates every few seconds during trading hours) using Python.

You'll first learn how to use the [*yahoo_fin*](#) module to obtain real-time price information based on the ticker symbol of the stock. You'll then learn to scrape the web to get the ticker symbol based on the company name, so that when you enter the name of a firm into the program, Python will tell you the ticker symbol of the firm's stock.

Obtain the Latest Stock Price from Yahoo

You can obtain the latest stock price information from Yahoo Finance using the [*yahoo_fin*](#) module. The [*yahoo_fin*](#) module is not in the Python standard library, so you need to pip install it first.

Open your Anaconda prompt (in Windows) or a terminal (in Mac or Linux), activate the virtual environment, and run the following command (note that there is an underscore in the middle of the module name):

```
pip install yahoo_fin
```

Once done, open [*LivePrice.py*](#) in your Spyder editor (the program is in the zip file on Canvas), and the code is shown in Listing 5-1.

```
# import the stock_info submodule from the yahoo_fin module
from yahoo_fin import stock_info as si #❶

# start an infinite loop
while True: #❷
    # obtain ticker symbol from you
    ticker = input("What's the ticker symbol of the stock you are looking
for?\n") #❸
    # if you want to stop, type in "done"
    if ticker == "done": #❹
        break
    # otherwise, type in a stock ticker symbol
    else:
        try:
            # obtain live stock price from Yahoo
            price = si.get_live_price(ticker)

            # print out the stock price
            print(f"the stock price for {ticker} is {price}")
        except:
            print("invalid ticker symbol")
```

Listing 5-1: The program to retrieve real-time stock price

We first import the `stock_info` class from the `yahoo_fin` module ❶. We then put the program in an infinite loop ❷ to continuously take your written input about stock ticker symbols ❸. Whenever you want to stop the program, you can type in “done” ❹. Otherwise, the program will obtain the latest stock price information from Yahoo Finance for you. Finally, the program prints out the stock price information for you to see.

Below is the output from an exchange with the program, with user input in bold:

```
What's the ticker symbol of the stock you are looking for?
MSFT
the stock price for MSFT is 183.25

What's the ticker symbol of the stock you are looking for?
```

AAPL

the stock price for AAPL is 317.94000244140625

What's the ticker symbol of the stock you are looking for?

done

As you can see, we typed in the ticker symbols for Microsoft and Apple, **MSFT** and **AAPL**, respectively, and the program returned their latest prices.

Here I used **try** and **except** to handle errors in case you enter an invalid ticker symbol. This way, the program won't crash if you do enter a wrong ticker symbol. For more information about exception handling, see, e.g., <https://docs.python.org/3/tutorial/errors.html>

Notice that the price of the Apple stock has too many digits after the decimal, and you'll learn how to keep only two digits after decimal in all stock prices.

BEGIN BOX

Try it Out

5-1. Run the program *LivePrice.py* and find out the stock prices for Amazon and Tesla (ticker symbols AMZN and TSLA). Then go to the website <https://finance.yahoo.com> to check if the prices from the website are close to your results.

END BOX

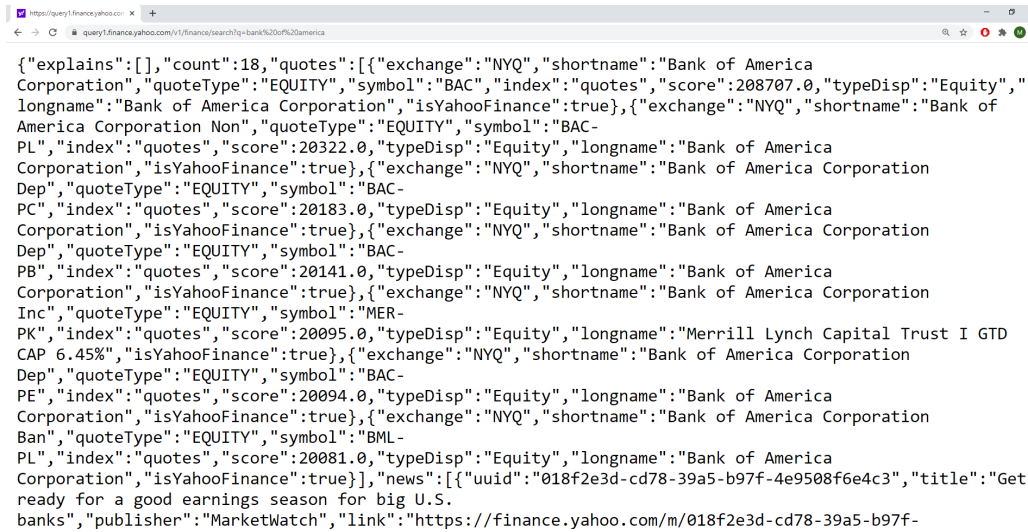
You may wonder, what if I don't know the ticker symbols of the stocks that I am interested in? Can Python find it out for us?

The answer is yes, and this is when the web scraping skills you learned in previous chapters become handy.

Search for the Ticker Symbol Based on the Company Name

Many times, we know only the company name, and not the ticker symbol. This script will find the ticker symbol when you enter the name of the company.

We need to first find a website that can reliably provide a company's ticker symbol. We'll use Yahoo! Finance, and query the site using the URL <https://query1.finance.yahoo.com/v1/finance/search?q=> followed by the name of the company you want to query. For example, if you put *Bank of America* at the end, as a set of Python-friendly data results, as shown in Figure 5-1.



```
{
  "explains": [],
  "count": 18,
  "quotes": [
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation",
      "quoteType": "EQUITY",
      "symbol": "BAC",
      "index": "quotes",
      "score": 208707.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Non-Preferred",
      "quoteType": "EQUITY",
      "symbol": "BAC-PL",
      "index": "quotes",
      "score": 20322.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Deposit",
      "quoteType": "EQUITY",
      "symbol": "BAC-PC",
      "index": "quotes",
      "score": 20183.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Deposit",
      "quoteType": "EQUITY",
      "symbol": "BAC-PB",
      "index": "quotes",
      "score": 20141.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Inc",
      "quoteType": "EQUITY",
      "symbol": "MER-PK",
      "index": "quotes",
      "score": 20095.0,
      "typeDisp": "Equity",
      "longname": "Merrill Lynch Capital Trust I GTD CAP 6.45%",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Deposit",
      "quoteType": "EQUITY",
      "symbol": "BAC-PE",
      "index": "quotes",
      "score": 20094.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    },
    {
      "exchange": "NYSE",
      "shortname": "Bank of America Corporation Ban",
      "quoteType": "EQUITY",
      "symbol": "BML-PL",
      "index": "quotes",
      "score": 20081.0,
      "typeDisp": "Equity",
      "longname": "Bank of America Corporation",
      "isYahooFinance": true
    }
  ],
  "news": [
    {
      "uuid": "018f2e3d-cd78-39a5-b97f-4e9508f6e4c3",
      "title": "Get ready for a good earnings season for big U.S. banks",
      "publisher": "MarketWatch",
      "link": "https://finance.yahoo.com/m/018f2e3d-cd78-39a5-b97f-"
    }
  ]
}
```

Figure 5-1 Results when you search for the ticker symbol for Bank of America

This data is formatted in *JSON*, short for *JavaScript Object Notation*. This file format is used for browser-server communication that uses human-readable text to store and transmit data objects. JSON was derived from JavaScript, but is now a language-independent data format that's used by many programming languages, include Python.

To make the JSON data easier to read, we'll use the online JSON data formatter at <https://jsonformatter.curiousconcept.com/>. Open the URL and you'll see a screen similar to Figure 5-2.

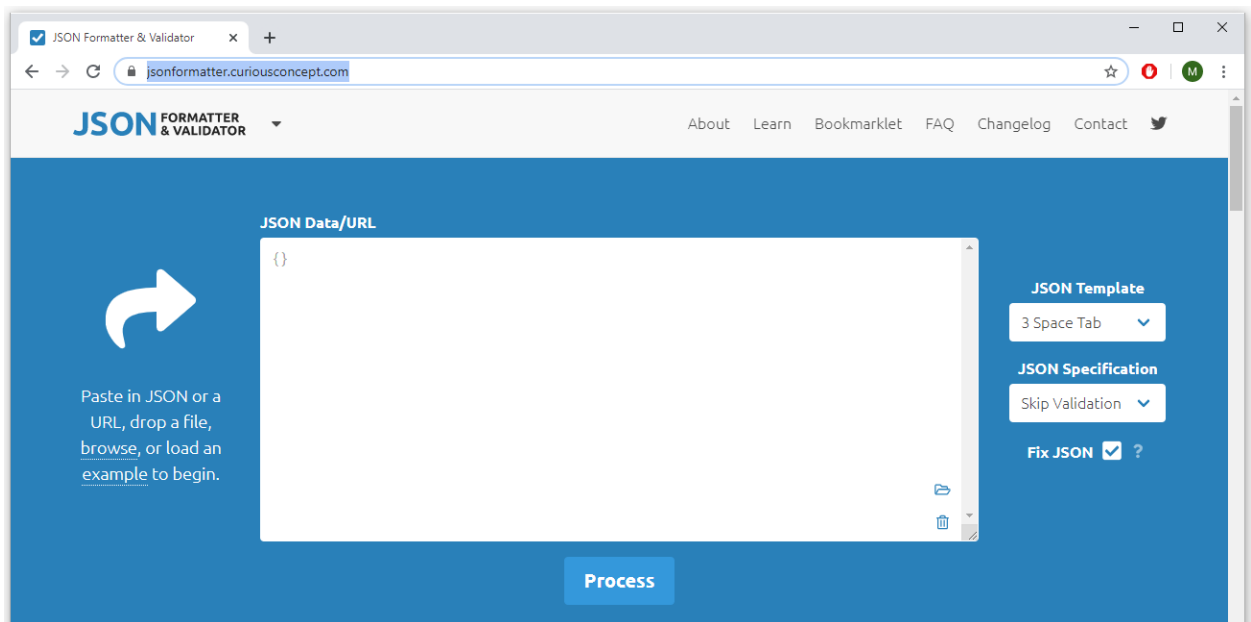


Figure 5-2: A website to format JSON data

Paste the data from Figure 5-1 into the designated space and click the **Process** button. The formatter will convert the data into a much more readable format, shown below

```
{
  1 "explains": [
    ],
    "count": 18,
    2 "quotes": [
      3 {
        "exchange": "NYSE",
        "shortname": "Bank of America Corporation",
        "quoteType": "EQUITY",
        4 "symbol": "BAC",
        "index": "quotes",
        "score": 208707.0,
        "typeDisp": "Equity",
        "longname": "Bank of America Corporation",
        "isYahooFinance": true
      },
      {
        "exchange": "NYSE",
        "shortname": "Bank of America Corporation Non",
        "quoteType": "EQUITY",
        "symbol": "BAC-PL",
        "index": "quotes",
        "score": 20322.0,
        "typeDisp": "Equity",
        "longname": "Bank of America Corporation",
        "isYahooFinance": true
      },
      {
        "exchange": "NYSE",
        "shortname": "Bank of America Corporation Dep",
        "quoteType": "EQUITY",
        "symbol": "BAC-PC",
        "index": "quotes",
        "score": 20183.0,
        "typeDisp": "Equity",
        "longname": "Bank of America Corporation",
        "isYahooFinance": true
      }
    ],
    --snip--
  }
}
```

The dataset is a large dictionary of several elements with the key values `explains` 1, `count`, `quotes` 2, and so on. The ticker symbol information we need is buried in the `quotes` key, which is a list of several dictionaries. The first dictionary contains the keys `exchange`, `shortname`, `quoteType`—and importantly, `symbol` 3, which contains the value `BAC`, the ticker symbol we need 4.

Next, we use a Python script to extract the ticker symbol based on the preceding pattern.

The script `get_ticker_symbol.py` below accomplishes that.

```

import requests

# Start an infinite loop
1 while True:
    # Obtain company name from you
    2 firm = input("Which company's ticker are you looking for?\n")
    # If you want to stop, type in "done"
    3 if firm == "done":
        break
    # Otherwise, type in a company name
    4 else:
        5 try:
            # Extract the source code from the website
            url =
'https://query1.finance.yahoo.com/v1/finance/search?q='+firm
            response = requests.get(url)
            # Read the JSON data
            response_json = response.json()
            # Obtain the value corresponding to "quotes"
            6 quotes = response_json['quotes']
            # Get the ticker symbol
            7 ticker = quotes[0]['symbol']
            # Print out the ticker
            print(f"The ticker symbol for {firm} is {ticker}.")
        8 except:
            print("Sorry, not a valid entry!")
            continue

```

We import the *requests* module, which allows Python to send HyperText Transfer Protocol (HTTP) requests. At 1, we start an infinite loop that asks for your written input 2 in each iteration. To exit the loop, enter *done* 3. Otherwise, you enter in the company name 4. We use exception handling to prevent a crash 5.

We go into the JSON data and extract the list corresponding to the key *quotes* 6. We then go to the first element and look for the value corresponding to the key *symbol* 7. The script prints out the ticker symbol at the IPython console. If there are no results, the script will print *Sorry, not a valid entry!* 8.

Run the script a few times and search for several companies to check that it works. The following output is one interaction with the program:

```

Which company's ticker are you looking for?
ford motor
The ticker symbol for ford motor is F.

Which company's ticker are you looking for?
walt disney company
The ticker symbol for walt disney company is DIS.

Which company's ticker are you looking for?
apple
The ticker symbol for apple is AAPL.

```

```
Which company's ticker are you looking for?  
done
```

As you can see, the script works for companies with one-word names, like Apple, as well as longer company names, such as Walt Disney Company.

As you can see, the program works for companies with one word (such as Apple) as well as companies with more than two words (such as J. P. Morgan Chase). However, you need to use Ford Motor instead of Ford if you search for the ticker symbol for the Ford Motor Company because many companies have the word “Ford” in its name.

BEGIN BOX

Try it Out

5-2. Use the program *GetTickerSymbol.py* to find the ticker symbols for General Motors and Procter & Gamble.

END BOX

Project 2: Data Visualization in Finance

In project 1, you learned how to obtain real-time stock price from Yahoo Finance for a company. However, many times, you want to analyze recent price movements and observe patterns before making investment decisions.

One of the most efficient ways to do that is through data visualization. It’s difficult to detect patterns and trends amongst a large group of numbers. In contrast, data visualization takes in data and places them into a visual context such as plots and charts and makes it easy for human brains to understand.

In this project, you’ll learn how to obtain recent daily stock price information from Yahoo Finance. You’ll then plot a graph to see the price movements over time to detect patterns and trends. You’ll also learn to create candlestick charts so that you can see intraday stock movement patterns.

Create Stock Price Plots

You’ll learn to extract recent daily stock price information from Yahoo Finance using the *pandas_datareader* module. You’ll then use the *matplotlib* module to create plots for stock prices over the last six months.

Below, you’ll first learn how to extract data, and then you’ll learn how to create plots.

First, you need to install a few third-party modules. Go to your Anaconda prompt (in Windows) or a terminal (in Mac or Linux) and activate the virtual environment, then run the following lines of code one by one in the command line:

```
conda install pandas
conda install matplotlib
pip install pandas_datareader
```

Follow the instructions to finish the installations. The *pandas_datareader* module extracts online data from various sources into a *pandas DataFrame*, we therefore installed the *pandas* module above. We'll discuss in Chapter 8 later in this class on what is a *pandas DataFrame*.

Enter the lines of code in Listing 5-4 in your Spyder editor and save the program as *PricePlot.py* in your project folder. Or you can download the program from Canvas.

```
#import needed modules
import matplotlib.pyplot as plt
from pandas_datareader import data as pdr
import matplotlib.dates as mdates

#set the start and end dates
start_date = "2021-02-25" #❶
end_date = "2021-08-25"

#choose stock ticker symbol
ticker = "TSLA" #❷

#get stock price
stock = pdr.get_data_yahoo(ticker, start=start_date, end=end_date) #❸
print(stock)

#obtain dates
stock['Date']=stock.index.map(mdates.date2num) #❹

#choose figure size
fig = plt.figure(dpi=128, figsize=(10, 6)) #❺

#format date to place on the x-axis
formatter = mdates.DateFormatter('%m/%d/%Y') #❻
plt.gca().xaxis.set_major_formatter(formatter)

# Plot data.
plt.plot(stock['Date'], stock['Adj Close'], c='blue') #❼
# Format plot.
```

```
plt.title("The Stock Price of Tesla", fontsize=16) # 8
plt.xlabel('Date', fontsize=10)
fig.autofmt_xdate()
plt.ylabel("Price", fontsize=10)
plt.show() # 9
```

Listing 5-4: The program to create a stock price plot

We first import the needed modules. We then specify the start and end dates of the data you want to extract ❶. The dates should be in the format of YYYY-MM-DD. We use the six-month period from Feb 25, 2021, to Aug 25, 2021. We also provide the ticker symbol of the stock that you are interested in. We use Tesla as our example and the ticker symbol is TSLA ❷.

We use the `get_data_yahoo()` method in the *pandas_datareader* module to extract daily stock price information for Tesla and save the data as a *pandas DataFrame* named *stock* ❸. The dataset looks as follows:

	High	Low	...	Volume	Adj Close
Date			...		
2021-02-25	737.210022	670.580017	...	39023900	682.219971
2021-02-26	706.700012	659.510010	...	41089200	675.500000
2021-03-01	719.000000	685.049988	...	27136200	718.429993
2021-03-02	721.109985	685.000000	...	23732200	686.440002
2021-03-03	700.700012	651.710022	...	30208000	653.200012

2021-08-19	686.549988	667.590027	...	14313500	673.469971
2021-08-20	692.130005	673.700012	...	14781800	680.260010
2021-08-23	712.130005	680.750000	...	20264900	706.299988
2021-08-24	715.219971	702.640015	...	13029900	708.489990
2021-08-25	714.000000	704.070007	...	3431311	713.525818

[127 rows x 6 columns]

As you can see, the dataset uses dates as indexes. The dataset has 127 rows and six columns. The 127 rows represent the 127 trading days during the six-month period. The six columns represent the following information in each trading day: high price, low price, open price, closing price, trading volume, and adjusted closing price. Note that indexes are not considered a separate column of data in the dataset.

We then extract the index of the dataset and set it as the seventh column of the dataset ④. This step is necessary because the dataset doesn't recognize the index as a separate variable, but we need the date information to use as our *x*-axis. We then use the `figure()` function in `matplotlib.pyplot` to specify the size and resolution of the plot and name the generated figure *fig* ⑤. The `dpi=128` argument tells the program that the output will have 128 pixels per inch. The `figure=(10,6)` argument means that the plot will be 10 inches wide and 6 inches tall.

NOTE: DPI stands for (printer) dots per inch from the earlier days. Nowadays, it actually stands for pixels per inch. So DPI is a bit of a misnomer.

We then use the `DateFormatter()` method in `matplotlib.dates` to specify the format of the dates that we want to show on the *x*-axis ⑥. We do the actual plotting using the `plot()` method ⑦. The first two arguments are the variables to use on the *x*- and *y*-axis, respectively. We also use the third argument to specify the color of the plot. In this case, we plot the adjusted closing price against the date and use blue as the color.

NOTE: Adjusted closing price is the closing price adjusted for stock splits and cash dividends. In many cases, it's identical to the closing price.

Starting at ⑧, we put a title on the graph and labels the on the *x*- and *y*-axis. We also use the `autofmt_xdate()` method to show the dates on the *x*-axis diagonally to prevent overlapping.

Finally, the `show()` method is called to display the actual plot ⑨. The output is shown in Figure 5-3.

INSERT FIGURE 5-3 HERE

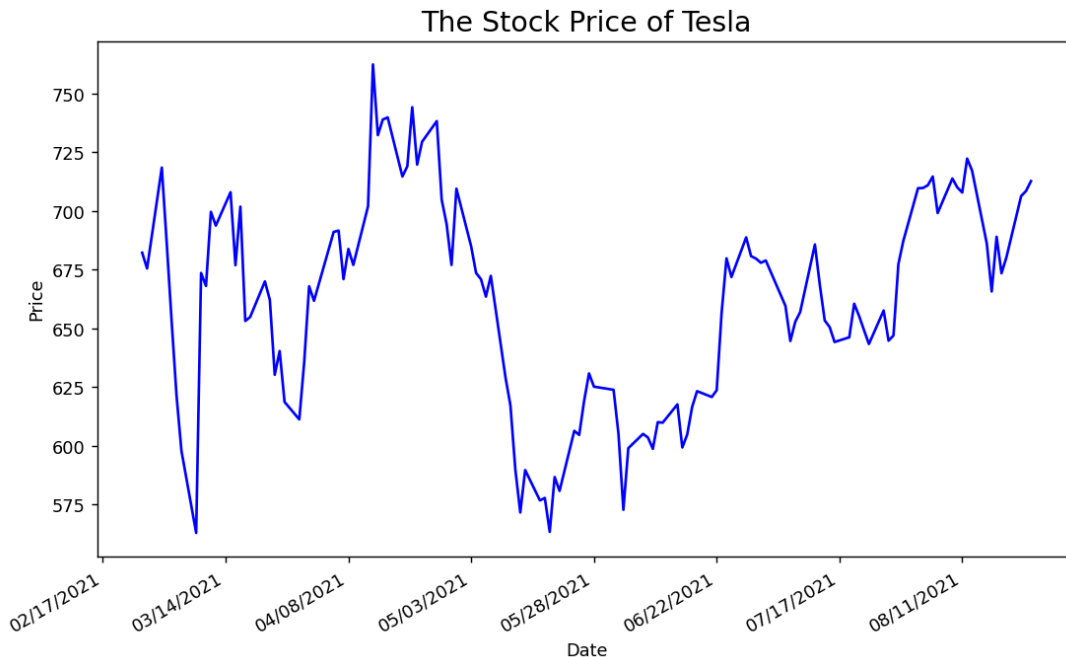


Figure 5-3: Stock price plot for Tesla

We can see the price movement patterns of Tesla in the six months from Feb to Aug 2021.

*NOTE: The default place for plots is in the **plots** pane in the Spyde IDE, in case you cannot locate the generated plot.*

BEGIN BOX

Try it Out

5-4. Run the program *PricePlot.py* and generate a stock price plot for Facebook (ticker symbol FB) for the last six months of 2021.

END BOX

Candlestick Charts

While a price plot gives you information on recent price movements of a stock, it provides only one observation for the stock reach day. Sometimes you are interested in intraday price movements, such as the range of the price fluctuation in a given day, whether the closing price is higher or lower than the opening price, and so on.

We can use candlestick charts to do that. You'll be able to visualize four pieces of information each day for a stock: daily high, daily low, opening price, and closing price. The following program generates the candlestick chart for the Amazon stock in the month of August

2021. We usually don't recommend using stock prices of more than a month because the chart may become too crowded and it's hard to detect the pattern.

First, you need to install the *mpl_finance* module. The *mpl_finance* module is not in the Python standard library, so you need to pip install it.

Open your Anaconda prompt (in Windows) or a terminal (in Mac or Linux), activate the virtual environment *chatting*, and run the following command:

```
pip install mpl_finance
```

Once done, open your Spyder editor, and save the following lines of code in Listing 5-5 as *CandleStick.py* in your project folder.

```
#import needed modules
import matplotlib.pyplot as plt
from pandas_datareader import data as pdr
import matplotlib.dates as mdates
from mpl_finance import candlestick_ohlc

#set the start and end date
start_date = "2021-08-01"
end_date = "2021-08-25"
#choose stock ticker symbol
ticker = "AMZN"
#get stock price
stock = pdr.get_data_yahoo(ticker, start=start_date, end=end_date)
#obtain dates
stock['Date']=stock.index.map(mdates.date2num)
#choose the four daily prices: open, high, low, and close
df_ohlc = stock[['Date','Open', 'High', 'Low', 'Close']] #❶
#choose figure size
figure, fig = plt.subplots(dpi=128, figsize = (8,4))
#format dates
formatter = mdates.DateFormatter('%m/%d/%Y')
#choose x-axis
fig.xaxis.set_major_formatter(formatter)
fig.xaxis_date()
plt.setp(fig.get_xticklabels(), rotation = 10) #❷
#create the candlestick chart
```

```

candlestick_ohlc(fig,
                  df_ohlc.values,
                  width=0.8,
                  colorup='green',
                  colordown='red')    # ❸

#put text in the chart that green color means close > open
plt.figtext(0.3,0.2, 'Green: Close > Open')    # ❹

#put text in the chart that red color means close < open
plt.figtext(0.3,0.15, 'Red: Close < Open')

#put chart title and axis labels
plt.title('Candlesticks Chart for Amazon Stock')    # ❺
plt.ylabel('Pirce')
plt.xlabel('Date')
plt.show()

```

Listing 5-5: The program to create candlestick chart

We first import all needed modules. In particular, we import the `candlestick_ohlc()` function from the `mpl_finance` module to create the candlestick chart.

The program retrieves the Amazon daily stock price information in the month of August 2021. At ❶, we select the four daily prices we'll use: opening price, closing price, daily high, and daily low.

The `setp()` function in the `matplotlib` module is used to set object properties. The two arguments in the function at ❷ ask the *x*-axis label to rotate 10 degrees, so as not to overlap. At ❸, we use the `candlestick_ohlc()` function to generate the candlestick chart. The first argument in the function specifies where to place the chart. The second argument is the data to use in the chart. The third argument is the width of the candle body relative to the distance between two observations.

The candlestick chart also uses color to denote whether the opening price is higher or lower than the day's closing price. We use green to indicate that the closing price is higher than the opening price. Otherwise, the color is red. We also put text in the chart to let readers know the meaning of the green and red colors ❹. Finally, we give the chart a title and label the two axes ❺.

The candlestick chart for Amazon stock price is shown in Figure 5-4.

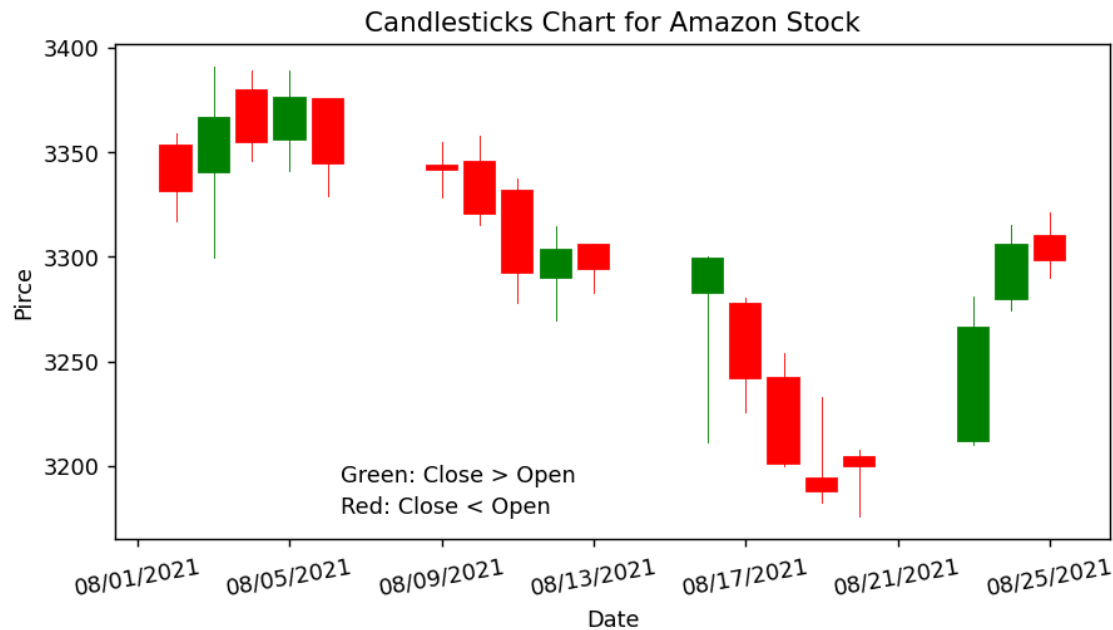


Figure 5-4: A candlestick chart for Amazon daily stock prices

The daily high and daily low are at the end of a thin line for the day (which looks like a candle wick), while the opening and closing prices are at the end of a wide line for the day (which looks like a candle body). Therefore, the graph for each day looks like a candle stick. The chart shows the intraday price movements.

BEGIN BOX

Try it Out

5-5. Run the program [CandleStick.py](#) and generate a candlestick chart for Wells Fargo (ticker symbol WFC) from Dec 1, 2021 to Dec 31, 2021.

END BOX

Project 3: Calculate Stock Alpha and Beta

You learned how to obtain real-time stock price from Yahoo Finance for a company and examine their recent price patterns using plots and charts. However, many times, you want a more detailed analysis on the stock's performance and risk before making investment decisions.

In this project, you'll learn how to obtain recent daily stock price information and perform regression analyses to figure out the recent performance and market risk of the stock.

You'll calculate the abnormal return (alpha) and the market risk (beta) of the stock by running a regression of the stock's return on the market return.

NOTE: For detailed explanations of alpha and beta of stock, see, for example, the relevant articles on Wikipedia: [https://en.wikipedia.org/wiki/Alpha_\(finance\)](https://en.wikipedia.org/wiki/Alpha_(finance)) and [https://en.wikipedia.org/wiki/Beta_\(finance\)](https://en.wikipedia.org/wiki/Beta_(finance)).

Analyze Recent Stock Performance and Risk

You'll extract recent daily stock price information from Yahoo Finance using the *pandas_datareader* module. You'll then use a new module *statsmodels* to perform statistical analyses.

Below, you'll first learn how to install the third party module and extract data. You'll then learn how to run regressions, analyze the data, and report the results.

Extract Daily Stock Price Information and Analyze

First, you need to install the third-party module *statsmodels*. Go to your Anaconda prompt (in Windows) or a terminal (in Mac or Linux) and activate the virtual *chatting* environment, then run the following command line:

```
conda install statsmodels
```

Enter the lines of code in Listing 5-9 in your Spyder editor and save the program as *AlphaBeta.py* in your project folder. Or you can download the file from Canvas.

```
# import needed modules
import statsmodels.api as sm
from pandas_datareader import data as pdr

# set the start and end dates
start_date = "2020-12-01" #❶
end_date = "2021-05-31"
# choose market index (S&P500) and stock ticker symbol
market = "^GSPC" #❷
ticker = "MSFT"
sp = pdr.get_data_yahoo(market, start=start_date, end=end_date) #❸
stock = pdr.get_data_yahoo(ticker, start=start_date, end=end_date)
#calculate returns for sp500 and the stock
```

```

sp['ret_sp'] =(sp['Adj Close']/sp['Adj Close'].shift(1))-1 #4
stock['ret_stock'] =(stock['Adj Close']/stock['Adj Close'].shift(1))-1
# merge the two datasets, keep only returns
df = sp[['ret_sp']].merge(stock[['ret_stock']], left_index=True,
right_index=True) #5
#add risk free rate (assume constant for simplicity)
df['rf'] = 0.00001 #6
# we need a constant to run regressions
df['const'] = 1
df['exret_stock'] = df.ret_stock - df.rf
df['exret_sp'] = df.ret_sp - df.rf
# remove missing values
df.dropna(inplace=True)
# Calculate the stock's alpha and Beta
reg = sm.OLS(endog=df['exret_stock'], exog=df[['const', 'exret_sp']],
missing='drop') #7
results = reg.fit()
print(results.summary())
alpha=round(results.params['const']*100,3) #8
beta=round(results.params['exret_sp'],2)
# print the values of alpha and beta
print(f'the alpha of the stock {ticker} is {alpha} percent')
print(f'the beta of the stock {ticker} is {beta}')

```

Listing 5-9: The program to calculate stock alpha and beta

We first import the needed modules. We then specify the start and end dates of the data you want to extract ❶. The dates should be in the format of YYYY-MM-DD. We use the six-month period from December 1, 2020, to May 31, 2021. We also provide the ticker symbols of the market index (the convention is to use the S&P 500 Index) and Microsoft Corporation (as our example in this case) ❷.

We use the `get_data_yahoo()` method in the *pandas_datareader* module to extract daily stock price information for the market index and Microsoft and save the data as two *pandas DataFrame* named *sp* and *stock*, respectively ❸.

We then calculate the daily stock returns for both the S&P 500 and Microsoft ❹. The `shift()` method in *pandas* allows one to shift the index by a desired number of periods. We use `shift(1)` to obtain the price information in the previous trading day.

To do the actual calculation, we first merge the two datasets into one ⑤. Because we are dealing with daily stock returns, the risk-free rate is very small and doesn't make a significant difference, so for simplicity we use a small constant value ⑥. We then use the `OLS()` method in the *statsmodels* module to run a regression ⑦. We also print out the regression results. The alpha and beta we want are the regression coefficients on the constant and the excess return on the market, respectively ⑧.

The regression results are shown in Figure 5-7.

INSERT FIGURE 5-7 HERE

OLS Regression Results						
Dep. Variable:	exret_stock	R-squared:	0.468			
Model:	OLS	Adj. R-squared:	0.463			
Method:	Least Squares	F-statistic:	106.3			
Date:	Wed, 25 Aug 2021	Prob (F-statistic):	2.84e-18			
Time:	10:42:17	Log-Likelihood:	387.80			
No. Observations:	123	AIC:	-771.6			
Df Residuals:	121	BIC:	-766.0			
Df Model:	1					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	-3.192e-06	0.001	-0.003	0.997	-0.002	0.002
exret_sp	1.1334	0.110	10.311	0.000	0.916	1.351
Omnibus:	1.720	Durbin-Watson:	1.972			
Prob(Omnibus):	0.423	Jarque-Bera (JB):	1.306			
Skew:	0.052	Prob(JB):	0.521			
Kurtosis:	3.494	Cond. No.	117.			
Notes:						
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.						

Figure 16-7: Regression analysis results for Microsoft.

Finally, we print out the values of the firm's alpha and beta. The output is as follows:

```
the alpha of the stock MSFT is -0.0 percent
the beta of the stock MSFT is 1.13
```

As you can see, the analysis shows that the alpha and beta are 0.0% and 1.1, respectively. This means Microsoft has outperformed similar stocks on the market by 0.0% per day, and the company has a market risk slightly greater than an average firm (which has a beta of 1).

BEGIN BOX

Try it Out

5-6. Use the program *AlphaBeta.py* to obtain the alpha and beta for the stock of British Petroleum (BP) using the most recent six months of data.

END BOX

Summary

In this chapter, you learned how to obtain and analyze financial data.

You first learned how to obtain a firm's stock ticker symbol based on company name. You then use the ticker symbol to retrieve real-time stock price and ask Python to tell you the information.

In the second project, you learned to obtain recent daily stock price information and create price plots and candlestick charts.

In the third project, you learned to conduct a detailed regression analysis. The program retrieves recent daily stock prices and calculates returns. The program then runs regressions and calculate the alpha and beta of the stock you are interested in.

End of Chapter Exercises

- 1: Modify *PricePlot.py* so that: i) The starting date is March 1, 2020; ii) change the plot color to red.
- 2: Modify *CandleStick.py* so that the dates on the x-axis are in the format of 01-23-2023 (instead of 01/23/2023, or January 23, 2023) and rotate 15 degrees (instead of 10 degrees).
3. Combine *LivePrice.py* and *GetTickerSymbol.py* so that the script asks for the company name, and after you enter the company name, it will print out both the ticker symbol and the stock price of the company.
4. We hardcoded the ticker symbol in *CandleStick.py*. Now, modify *CandleStick.py* so that after you run the program, it will ask you three questions one by one: ticker symbol, starting date, and end date. Once you finish entering the three pieces of information, the program will create a candlestick chart accordingly. Note that you have to enter start and end dates in the format of 2022-08-01. After the modification, run the program, and enter the ticker symbol as GS, and set starting and end dates as Jan 1 and Jan 31 of 2023. Show your output. That is, you have to copy and paste the chart, as well as your input and output in the IPython console.

5. Modify *AlphaBeta.py* so that after you run the program, it will ask you three questions one by one: ticker symbol, starting date, and end date. Once you finish entering the three pieces of information, the program will print out the regression results, the alpha, and the beta. After the modification, run the program, and enter the ticker symbol as AMZN, and set starting and end dates as Aug 1, 2022 and Jan 31, 2023, respectively. Show your output. That is, you have to copy and paste your input and output in the IPython console.