File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ labs / project /

Name	Last Modified
Final Assig...	4 days ago
Final Assig...	6 minutes ago
Final_Assig...	4 days ago
WebScrapi...	4 days ago

Final Assignment.ipynb

Code

Open in... Python 3 (ipykernel)

Skills Network

Extracting and Visualizing Stock Data

Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

Table of Contents

Estimated Time Needed: **30 min**

Note:- If you are working Locally using anaconda, please uncomment the following code and execute it.

[1]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

[6]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.43-py2.py3-none-any.whl.metadata (11 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading pandas-2.2.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (19 kB)
Collecting numpy>=1.16.5 (from yfinance)
  Downloading numpy-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
     ---------------------------------------- 60.9/60.9 kB 9.8 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.31 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2.31.0)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Collecting lxml>=4.9.1 (from yfinance)
  Downloading lxml-5.3.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (3.8 kB)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/lib/python3.11/site-packages (from yfinance) (4.2.1)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/lib/python3.11/site-packages (from yfinance) (2024.1)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.4-py311-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.6.tar.gz (3.0 MB)
     ---------------------------------------- 3.0/3.0 MB 78.8 MB/s eta 0:00:00:00:01
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.11/site-packages (from yfinance) (4.12.3)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.11/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.11/site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2.9.0)
Collecting tzdata>=2022.7 (from pandas>=1.3.0->yfinance)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2024.6.2)
Downloading yfinance-0.2.43-py2.py3-none-any.whl (84 kB)
   ---------------------------------------- 84.6/84.6 kB 12.1 MB/s eta 0:00:00
Downloading frozendict-2.4.4-py311-none-any.whl (16 kB)
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
   ---------------------------------------- 112.2/112.2 kB 21.1 MB/s eta 0:00:00
Downloading lxml-5.3.0-cp311-cp311-manylinux_2_28_x86_64.whl (5.0 MB)
   ---------------------------------------- 5.0/5.0 MB 120.7 MB/s eta 0:00:00:00:01
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading numpy-2.1.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.3 MB)
   ---------------------------------------- 16.3/16.3 MB 101.9 MB/s eta 0:00:00:00:0100:01
Downloading pandas-2.2.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.0 MB)
   ---------------------------------------- 13.0/13.0 MB 106.1 MB/s eta 0:00:00:00:0100:01
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
   ---------------------------------------- 345.4/345.4 kB 46.4 MB/s eta 0:00:00
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) ... done
  Created wheel for peewee: filename=peewee-3.17.6-py3-none-any.whl size=138891 sha256=a9edd26e9fd416cae53d14591cc24e6982b50301d6c7ae0d2e33469c52a04fae
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/1c/09/7e/9f659fde248ecdc1722a142c1d744271aad3914a0afc191058
Successfully built peewee
Installing collected packages: peewee, multitasking, tzdata, numpy, lxml, html5lib, frozendict, pandas, yfinance
Successfully installed frozendict-2.4.4 html5lib-1.1 lxml-5.3.0 multitasking-0.0.11 numpy-2.1.0 pandas-2.2.2 peewee-3.17.6 tzdata-2024.1 yfinance-0.2.43
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.11/site-packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-packages (from beautifulsoup4->bs4) (2.5)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
Requirement already satisfied: nbformat in /opt/conda/lib/python3.11/site-packages (5.10.4)
Requirement already satisfied: fastjsonschema>2.15 in /opt/conda/lib/python3.11/site-packages (from nbformat) (2.19.1)
Requirement already satisfied: jsonschema>=2.6 in /opt/conda/lib/python3.11/site-packages (from nbformat) (4.22.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /opt/conda/lib/python3.11/site-packages (from nbformat) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/lib/python3.11/site-packages (from nbformat) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: platformdirs>=2.5 in /opt/conda/lib/python3.11/site-packages (from jupyter-core!=5.0.*,>=4.12->nbformat) (4.2.1)
```

[10]:
```
import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

[11]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Define Graphing Function

In this section, we define the function `make_graph`. **You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

[12]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing = .3)
    stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date), y=stock_data_specific.Close.astype("float"), name="Share Price"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date), y=revenue_data_specific.Revenue.astype("float"), name="Revenue"), row=2, col=1)
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
```

```
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard.

> **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

[14]:
```
import yfinance as yf
import pandas as pd

# Create ticker object for Tesla
tesla_ticker = yf.Ticker("TSLA")

# Extract Tesla stock data
tesla_data = tesla_ticker.history(period="max")

# Reset the index and save the changes
tesla_data.reset_index(inplace=True)

# Display the first five rows of the tesla_data dataframe
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

Using the ticker object and the function `history` extract stock information and save it in a `dataframe` named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[15]:
```
import yfinance as yf
import pandas as pd

# Create ticker object for Tesla
tesla_ticker = yf.Ticker("TSLA")

# Extract stock information and save it in a dataframe
tesla_data = tesla_ticker.history(period="max")

# Display the first five rows to verify
print(tesla_data.head())
```

```
                                Open      High       Low     Close     Volume  \
Date
2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667  281494500
2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667  257806500
2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000  123282000
2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000   77097000
2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000  103003500

                           Dividends  Stock Splits
Date
2010-06-29 00:00:00-04:00        0.0           0.0
2010-06-30 00:00:00-04:00        0.0           0.0
2010-07-01 00:00:00-04:00        0.0           0.0
2010-07-02 00:00:00-04:00        0.0           0.0
2010-07-06 00:00:00-04:00        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

[16]:
```
import yfinance as yf
import pandas as pd

# Create ticker object for Tesla
tesla_ticker = yf.Ticker("TSLA")

# Extract stock information and save it in a dataframe
tesla_data = tesla_ticker.history(period="max")

# Reset the index and save the changes
tesla_data.reset_index(inplace=True)

# Display the first five rows of the tesla_data dataframe
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

[17]:
```
import requests

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Send a GET request to the URL
response = requests.get(url)

# Save the text of the response as a variable named html_data
html_data = response.text

# Optionally, print the first few characters to verify
print(html_data[:500])  # Print the first 500 characters to check the content
```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
              <link rel="canonical" href="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue" />
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content`.

[18]:
```
import requests
from bs4 import BeautifulSoup

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Send a GET request to the URL
response = requests.get(url)

# Save the text of the response as a variable named html_data
html_data = response.text

# Parse the HTML data using BeautifulSoup with the 'html.parser' parser
soup = BeautifulSoup(html_data, 'html.parser')  # or use 'html5lib'

# Optionally, print the parsed data to verify
print(soup.prettify()[:500])  # Print the first 500 characters of the prettified HTML to check the content
```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js">
 <!--<![endif]-->
 <head>
  <meta charset="utf-8"/>
  <meta content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"/>
  <link href="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue" rel="canonical"/>
  <title>
   Te
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

▶ Step-by-step instructions

▶ Click here if you need help locating the table

[24]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Send a GET request to the URL
response = requests.get(url)

# Parse the HTML data using BeautifulSoup
soup = BeautifulSoup(response.text, 'html.parser')

# Find all tables in the HTML
tables = soup.find_all('table')

# Loop through tables to find the relevant one
for table in tables:
    if "Tesla Quarterly Revenue" in table.text:
        tesla_table = table
        break

# Initialize an empty DataFrame with columns 'Date' and 'Revenue'
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

# Find all rows in the selected table
rows = tesla_table.find_all('tr')

# Create a list to store rows temporarily
data = []

for row in rows[1:]:  # Skip the header row
    cols = row.find_all('td')
    date = cols[0].text.strip()  # Extract the date
    revenue = cols[1].text.strip()  # Extract the revenue

    # Clean the revenue data
    revenue = revenue.replace('$', '').replace(',', '')

    # Append the date and revenue to the data list
    data.append({"Date": date, "Revenue": revenue})

# Convert the list of dictionaries to a DataFrame
tesla_revenue = pd.DataFrame(data)

# Display the first few rows of the DataFrame
print(tesla_revenue.head())
```

```
         Date Revenue
0  2022-09-30   21454
1  2022-06-30   16934
2  2022-03-31   18756
3  2021-12-31   17719
4  2021-09-30   13757
```

Execute the following line to remove the comma and dollar sign from the `Revenue` column.

[28]:
```
# Remove the comma and dollar sign from the 'Revenue' column using regex
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$', '', regex=True)

# Display the first few rows of the DataFrame to verify
print(tesla_revenue.head())
```

```
         Date Revenue
0  2022-09-30   21454
1  2022-06-30   16934
2  2022-03-31   18756
3  2021-12-31   17719
4  2021-09-30   13757
```

Execute the following lines to remove an null or empty strings in the Revenue column.

[31]:
```
# Remove rows with null values in the 'Revenue' column
tesla_revenue.dropna(subset=['Revenue'], inplace=True)

# Remove rows where 'Revenue' is an empty string
tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]

# Display the first few rows of the DataFrame to verify
print(tesla_revenue.head())
```

```
         Date Revenue
0  2022-09-30   21454
1  2022-06-30   16934
2  2022-03-31   18756
3  2021-12-31   17719
4  2021-09-30   13757
```

Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[32]:
```
# Display the last 5 rows of the DataFrame
print(tesla_revenue.tail())
```

```
         Date Revenue
48  2010-09-30      31
49  2010-06-30      28
50  2010-03-31      21
52  2009-09-30      46
53  2009-06-30      27
```

Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

[33]:
```
import yfinance as yf

# Create ticker object for GameStop
```

```
gamestop_ticker = yf.Ticker("GME")

# Extract GameStop stock data
gamestop_data = gamestop_ticker.history(period="max")

# Reset the index
gamestop_data.reset_index(inplace=True)

# Display the first five rows
print(gamestop_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615921  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

[34]:
```
import yfinance as yf
import pandas as pd

# Create ticker object for GameStop
gme_ticker = yf.Ticker("GME")

# Extract GameStop stock data with maximum period
gme_data = gme_ticker.history(period="max")

# Reset the index
gme_data.reset_index(inplace=True)

# Display the first five rows of the DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666417  1.666417  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662209  1.603296  1.662209   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

[35]:
```
import yfinance as yf
import pandas as pd

# Create ticker object for GameStop
gme_ticker = yf.Ticker("GME")

# Extract GameStop stock data with maximum period
gme_data = gme_ticker.history(period="max")

# Reset the index
gme_data.reset_index(inplace=True)

# Display the first five rows of the DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716073  1.670625  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2`.

[37]:
```
import requests

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send a GET request to the URL
response = requests.get(url)

# Save the text of the response to a variable named html_data_2
html_data_2 = response.text

# Print the first 500 characters of html_data_2 to verify
print(html_data_2[:500])
```

```
<!DOCTYPE html>
<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue -->
<html class=" js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage borderradius boxsh
adow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransforms3d csstransitions fontface g
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

[40]:
```
from bs4 import BeautifulSoup
import requests

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send a GET request to the URL
response = requests.get(url)

# Save the text of the response to a variable named html_data_2
html_data_2 = response.text

# Parse the HTML data using BeautifulSoup with html.parser
soup = BeautifulSoup(html_data_2, 'html.parser')

# Print the first 500 characters of the parsed data to verify
print(soup.prettify()[:500])
```

```
<!DOCTYPE html>
<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue -->
<html class="js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage borderradius boxsha
dow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransforms3d csstransitions fontface ge
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

> **Note: Use the method similar to what you did in question 2.**

▶ Click here if you need help locating the table

[41]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL of the webpage to be downloaded
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send a GET request to the URL
response = requests.get(url)

# Save the text of the response to a variable named html_data_2
html_data_2 = response.text

# Parse the HTML data using BeautifulSoup with html.parser
soup = BeautifulSoup(html_data_2, 'html.parser')

# Find all tables in the HTML
tables = soup.find_all('table')

# Loop through tables to find the relevant one
for table in tables:
    if "GameStop Revenue" in table.text:
        gme_table = table
        break

# Initialize an empty DataFrame with columns 'Date' and 'Revenue'
gme_revenue = pd.DataFrame(columns=["Date", "Revenue"])

# Find all rows in the selected table
rows = gme_table.find_all('tr')

# Create a list to store rows temporarily
data = []

for row in rows[1:]:  # Skip the header row
    cols = row.find_all('td')
    date = cols[0].text.strip()  # Extract the date
    revenue = cols[1].text.strip()  # Extract the revenue

    # Clean the revenue data
    revenue = revenue.replace('$', '').replace(',', '')

    # Append the date and revenue to the data list
    data.append({"Date": date, "Revenue": revenue})

# Convert the list of dictionaries to a DataFrame
gme_revenue = pd.DataFrame(data)

# Display the first few rows of the DataFrame to verify
print(gme_revenue.head())
```

```
                                Date Revenue
0  GameStop Revenue 2006-2020 | GME
1                        Macrotrends
2                             Source
```

Display the last five rows of the `gme_revenue` dataframe using the `tail` function. Take a screenshot of the results.

[42]:
```
# Display the last five rows of the DataFrame
print(gme_revenue.tail())
```

```
                                Date Revenue
0  GameStop Revenue 2006-2020 | GME
1                        Macrotrends
2                             Source
```

Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

▶ Hint

[48]:
```

!pip install matplotlib

import matplotlib.pyplot as plt
print("matplotlib version:", plt.__version__)

import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Define the make_graph function
def make_graph(stock_data, revenue_data, title):
    fig, ax1 = plt.subplots(figsize=(14, 7))

    # Plot stock data
    ax1.plot(stock_data['Date'], stock_data['Close'], 'b-', label='Stock Price')
    ax1.set_xlabel('Date')
    ax1.set_ylabel('Stock Price', color='b')
    ax1.tick_params('y', colors='b')

    # Plot revenue data on a secondary y-axis
    ax2 = ax1.twinx()
    ax2.plot(revenue_data['Date'], revenue_data['Revenue'], 'r-', label='Revenue')
    ax2.set_ylabel('Revenue', color='r')
    ax2.tick_params('y', colors='r')

    # Set the title and show the plot
    plt.title(title)
    plt.show()

# Example data for testing
data = {
    'Date': pd.date_range(start='2020-01-01', periods=10, freq='M'),
    'Close': [200 + i * 10 for i in range(10)]
}
revenue = {
    'Date': pd.date_range(start='2020-01-01', periods=10, freq='M'),
    'Revenue': [100000 + i * 5000 for i in range(10)]
}

tesla_data = pd.DataFrame(data)
tesla_revenue = pd.DataFrame(revenue)

# Test plot
make_graph(tesla_data, tesla_revenue, 'Tesla Test Graph')
```

```
Collecting matplotlib
  Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.53.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (162 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.6/162.6 kB 19.2 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.4 kB)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.11/site-packages (from matplotlib) (2.1.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.11/site-packages (from matplotlib) (24.0)
Collecting pillow>=8 (from matplotlib)
  Downloading pillow-10.4.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (9.2 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.1.4-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.11/site-packages (from matplotlib) (2.9.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 124.9 MB/s eta 0:00:00a 0:00:01
Downloading contourpy-1.3.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.2/323.2 kB 47.8 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.53.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 126.9 MB/s eta 0:00:0000:01
Downloading kiwisolver-1.4.5-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 100.9 MB/s eta 0:00:00
```

```
Downloading pillow-10.4.0-cp311-cp311-manylinux_2_28_x86_64.whl (4.5 MB)
   ---------------------------------------- 4.5/4.5 MB 116.9 MB/s eta 0:00:00:00:01
Downloading pyparsing-3.1.4-py3-none-any.whl (104 kB)
   ---------------------------------------- 104.1/104.1 kB 17.0 MB/s eta 0:00:00
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, contourpy, matplotlib
Successfully installed contourpy-1.3.0 cycler-0.12.1 fonttools-4.53.1 kiwisolver-1.4.5 matplotlib-3.9.2 pillow-10.4.0 pyparsing-3.1.4
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[48], line 4
      1 get_ipython().system('pip install matplotlib')
      3 import matplotlib.pyplot as plt
----> 4 print("matplotlib version:", plt.__version__)
      6 import yfinance as yf
      7 import pandas as pd

AttributeError: module 'matplotlib.pyplot' has no attribute '__version__'
```

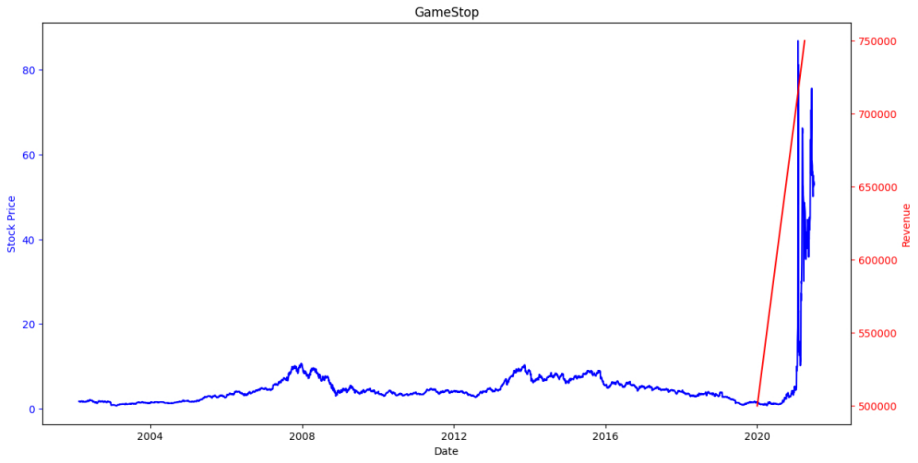
Question 6: Plot GameStop Stock Graph

Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

▼ Hint

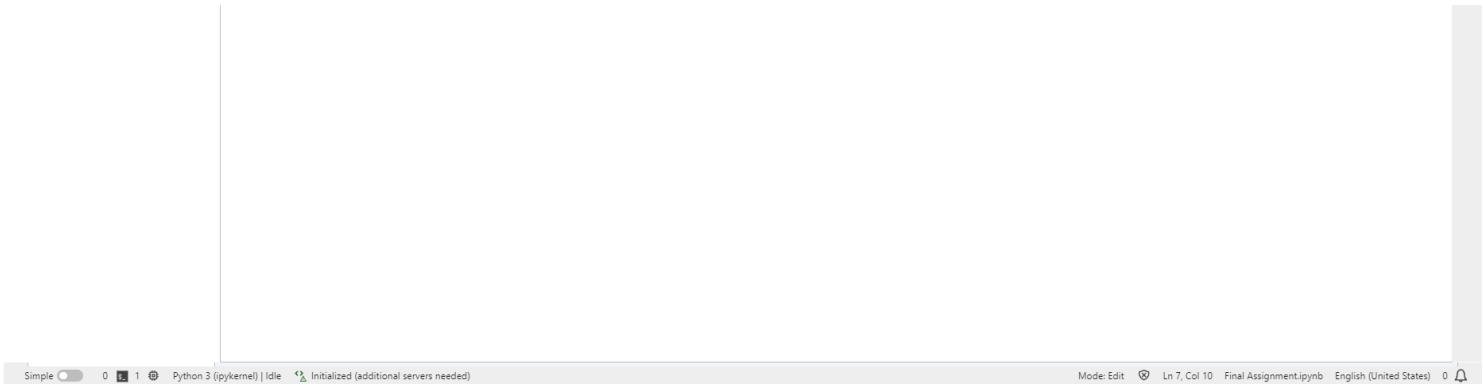
You just need to invoke the make_graph function with the required parameter to print the graphs.The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`

[49]:
```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Define the make_graph function
def make_graph(stock_data, revenue_data, title):
    fig, ax1 = plt.subplots(figsize=(14, 7))

    # Plot stock data
    ax1.plot(stock_data['Date'], stock_data['Close'], 'b-', label='Stock Price')
    ax1.set_xlabel('Date')
    ax1.set_ylabel('Stock Price', color='b')
    ax1.tick_params('y', colors='b')

    # Plot revenue data on a secondary y-axis
    ax2 = ax1.twinx()
    ax2.plot(revenue_data['Date'], revenue_data['Revenue'], 'r-', label='Revenue')
    ax2.set_ylabel('Revenue', color='r')
    ax2.tick_params('y', colors='r')

    # Set the title and show the plot
    plt.title(title)
    plt.show()

# Extract GameStop stock data
gme_ticker = yf.Ticker("GME")
gme_data = gme_ticker.history(period="max")
gme_data.reset_index(inplace=True)

# Example gme_revenue DataFrame (replace with your actual revenue data)
gme_revenue = pd.DataFrame({
    "Date": ["2020-01-01", "2020-04-01", "2020-07-01", "2020-10-01", "2021-01-01", "2021-04-01", "2021-07-01"],
    "Revenue": ["500000", "550000", "600000", "650000", "700000", "750000", "800000"]
})

# Convert 'Date' column to datetime
gme_revenue['Date'] = pd.to_datetime(gme_revenue['Date'])

# Remove data after June 2021
gme_data = gme_data[gme_data['Date'] <= '2021-06-30']
gme_revenue = gme_revenue[gme_revenue['Date'] <= '2021-06-30']

# Clean the Revenue column
gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$', "", regex=True).astype(float)

# Plot GameStop data
make_graph(gme_data, gme_revenue, 'GameStop')
```



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.


Simple
0
1
Python 3 (ipykernel) | Idle
Initialized (additional servers needed)
Mode: Edit
Ln 7, Col 10
Final Assignment.ipynb
English (United States)
0