



## 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

- Define a Function that Makes a Graph
- Question 1: Use yfinance to Extract Stock Data
- Question 2: Use Webscraping to Extract Tesla Revenue Data
- Question 3: Use yfinance to Extract Stock Data
- Question 4: Use Webscraping to Extract GME Revenue Data
- Question 5: Plot Tesla Stock Graph
- Question 6: Plot GameStop Stock Graph

Estimated Time Needed: **30 min**

**Note:-** If you are working in IBM Cloud Watson Studio, please replace the command for installing nbformat from `!pip install nbformat==4.2.0` to simply `!pip install nbformat`

```
!pip install yfinance==0.1.67
!mamba install bs4==4.10.0 -y
!pip install nbformat==4.2.0
```

```
Collecting yfinance==0.1.67
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from yfinance==0.1.67) (1.5.3)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.10/dist-packages (from yfinance==0.1.67) (1.23.5)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from yfinance==0.1.67) (2.31.0)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.10/dist-packages (from yfinance==0.1.67) (0.0.1)
Requirement already satisfied: lxml>=4.5.1 in /usr/local/lib/python3.10/dist-packages (from yfinance==0.1.67) (4.9.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->yfinance==0.1.67) (2.8.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->yfinance==0.1.67) (2020.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->yfinance==0.1.67) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->yfinance==0.1.67) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->yfinance==0.1.67) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->yfinance==0.1.67) (2023.7.22)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.24->yfinance==0.1.67) (1.16.0)
Installing collected packages: yfinance
  Attempting uninstall: yfinance
    Found existing installation: yfinance 0.2.33
    Uninstalling yfinance-0.2.33:
      Successfully uninstalled yfinance-0.2.33
Successfully installed yfinance-0.1.67
/bin/bash: line 1: mamba: command not found
Collecting nbformat==4.2.0
  Downloading nbformat-4.2.0-py2.py3-none-any.whl (153 kB)
    153.3/153.3 kB 2.2 MB/s eta 0:00:00
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (4.17.0)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (5.5.1)
Requirement already satisfied: traitlets>=4.1 in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (5.7.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.32.0)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.10.6)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core->nbformat==4.2.0) (3.10.0)
Installing collected packages: nbformat
  Attempting uninstall: nbformat
    Found existing installation: nbformat 5.9.2
    Uninstalling nbformat-5.9.2:
```

```
Successfully uninstalled nbformat-5.9.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
jupyter-server 1.24.0 requires nbformat>=5.2.0, but you have nbformat 4.2.0 which is incompatible.
nbclient 0.9.0 requires nbformat>=5.1, but you have nbformat 4.2.0 which is incompatible.
nbconvert 6.5.4 requires nbformat>=5.1, but you have nbformat 4.2.0 which is incompatible.
Successfully installed nbformat-4.2.0
```

```
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.

```
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.

```
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"), vert
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, infer_datetime_format=True), y=stock_data_specific.Close
fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_datetime_format=True), y=revenue_data_specific.F
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()
```

## ✓ 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`.

```
import yfinance as yf

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

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.

```
tesla = yf.Ticker("TSLA")

tesla_data = tesla.history(period='max')
```

```

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-21-920e648eda6e> in <cell line: 1>()
----> 1 tesla_data = tesla.history(period='max')

/usr/local/lib/python3.10/dist-packages/yfinance/base.py in history(self, period, interval, start, end, prepost, actions,
auto_adjust, back_adjust, proxy, rounding, tz, timeout, **kwargs)
    277
    278     # index eod/intraday
--> 279     df.index = df.index.tz_localize("UTC").tz_convert(
    280         data["chart"]["result"][0]["meta"]["exchangeTimezoneName"])
    281

```

**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.

```

import yfinance as yf
import pandas as pd

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

# Get historical market data for the maximum available period
tesla_data = ticker.history(period="max")

# Reset the index of tesla_data DataFrame in place
tesla_data.reset_index(inplace=True)

# Displaying the first five rows of the tesla_data DataFrame
print(tesla_data.head())

```

	Date	Open	High	Low	Close	Volume	Dividends	\
0	2010-06-29	1.266667	1.666667	1.169333	1.592667	281494500	0	
1	2010-06-30	1.719333	2.028000	1.553333	1.588667	257806500	0	
2	2010-07-01	1.666667	1.728000	1.351333	1.464000	123282000	0	
3	2010-07-02	1.533333	1.540000	1.247333	1.280000	77097000	0	
4	2010-07-06	1.333333	1.333333	1.055333	1.074000	103003500	0	

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

```

import requests

# URL of the webpage you want to download
html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/'

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

# Check if the request was successful (status code 200 indicates success)
if response.status_code == 200:
    # Print the content of the webpage (HTML, text, etc.)
    print(response.text)
else:
    print('Failed to download the webpage. Status code:', response.status_code)

```



```
$.post('https://api.ipstack.com/check?access_key=14fe63e83d5cfefa0b3d4cec498479ba&output=json&fields=ip,continent_name,count', function(ip_data){

    $(".contribute_user_id").val(ip_data.ip);

});

$( ".donate_buttons" ).click(function() {

    var payment = $(this).attr("value");

    $.post('https://api.ipstack.com/check?access_key=14fe63e83d5cfefa0b3d4cec498479ba&output=json&fields=ip,continent_name', function(ip_data){

        $.post('https://www.macrotrends.net/assets/php/page_view_tracking.php', {ip: ip_data.ip, paid: payment});

    });

});

</script>

-->

<script type="text/javascript">
var clicky_site_ids = clicky_site_ids || [];
clicky_site_ids.push(100827248);
(function() {
    var s = document.createElement('script');
    s.type = 'text/javascript';
    s.async = true;
    s.src = '//static.getclicky.com/js';
    ( document.getElementsByTagName('head')[0] || document.getElementsByTagName('body')[0] ).appendChild( s );
})();
</script>
<noscript><p></p></noscript>

<!-- This site is converting visitors into subscribers and customers with OptinMonster - https://optinmonster.com -->
<!-- <script type="text/javascript" src="https://a.omappapi.com/app/js/api.min.js" data-account="6392" data-user="15772" asy -->
<!-- / OptinMonster -->

</body>

</html>
```

Parse the html data using beautiful\_soup.

```
from bs4 import BeautifulSoup
```

```
html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork'
```

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.

► Click here if you need help locating the table

```
import pandas as pd

# URL of the webpage containing the Tesla Revenue table
html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork'

# Use read_html to extract tables from the webpage
tables = pd.read_html(html_data)

# Assuming the Tesla Revenue table is the first table on the page, retrieve it
tesla_revenue = tables[0]

# Rename columns to Date and Revenue if necessary
tesla_revenue.columns = ['Date', 'Revenue']
```

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

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# URL of the website containing the revenue data
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/'

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

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

# Find all tables on the webpage
tables = soup.find_all('table')

# Check the tables found
for i, table in enumerate(tables):
    df = pd.read_html(str(table))
    if len(df) > 0:
        df = df[0] # Assuming the table of interest is the first one found
        if 'Revenue' in df.columns:
            # Clean the 'Revenue' column by removing commas and dollar signs
            df['Revenue'] = df['Revenue'].str.replace(',', '').str.replace('$', '')

            # Display the cleaned DataFrame
            print(df.head())
            break # Exit loop after processing the table

# If 'Revenue' column is not found in any table
else:
    print("No table with a 'Revenue' column found.")
```

```
0          Sector \
1 Tesla is the market leader in battery-powered ...

0          Industry \
1 Tesla is the market leader in battery-powered ...

0          Market Cap \
1 Tesla is the market leader in battery-powered ...

0          Revenue
1 Tesla is the market leader in battery-powered ...
```

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

```
tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

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


```
last_5_rows = tesla_revenue.tail(5)
print(last_5_rows)
```

	Date	Revenue
8	2013	\$2,013
9	2012	\$413
10	2011	\$204
11	2010	\$117
12	2009	\$112

```
tesla_revenue.tail()
```

	Date	Revenue
8	2013	\$2,013
9	2012	\$413
10	2011	\$204
11	2010	\$117
12	2009	\$112

 **Generate**    Using ...    a slider using jupyter widgets        [Close](#)

Generate is available for a limited time for unsubscribed users. [Upgrade to Colab Pro](#) 

```
html_data = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork"
html_data = requests.get(html_data).text
```

```
beautiful_soup = BeautifulSoup(html_data,"html.parser")
```

```
import pandas as pd

# Assuming you have retrieved the necessary data using beautiful_soup
data = [
    {"Date": col[0].text, "Revenue": col[1].text}
    for row in beautiful_soup.find_all("tbody")[1].find_all("tr")
    for col in [row.find_all('td')]
]

tesla_revenue = pd.DataFrame(data)
```

```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"")
```

```
tesla_revenue = tesla_revenue[tesla_revenue['Revenue'].notna()]
```

```
tesla_revenue.tail()
```

	Date	Revenue
49	2010-06-30	28
50	2010-03-31	21
51	2009-12-31	
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`.

```
import yfinance as yf

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

# Get information about the stock
stock_info = ticker.info

# Display the stock information
print(stock_info)
```

```
{'regularMarketPrice': None, 'preMarketPrice': None, 'logo_url': ''}
```

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.

```
import yfinance as yf
import pandas as pd

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

# Get historical market data for the maximum available period
gme_data = ticker.history(period="max")

# Displaying the retrieved data or further processing
print(gme_data.head()) # Display the first few rows of the DataFrame
```

	Open	High	Low	Close	Volume	Dividends	\
Date							
2002-02-13	1.620128	1.693350	1.603296	1.691666	76216000	0.0	
2002-02-14	1.712707	1.716073	1.670626	1.683250	11021600	0.0	
2002-02-15	1.683250	1.687458	1.658001	1.674834	8389600	0.0	
2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	
2002-02-20	1.615920	1.662210	1.603296	1.662210	6892800	0.0	

Stock Splits	
Date	
2002-02-13	0.0
2002-02-14	0.0
2002-02-15	0.0
2002-02-19	0.0
2002-02-20	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.

```
# Reset the index of the gme_data DataFrame using the reset_index(inplace=True) function
gme_data.reset_index(inplace=True)

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

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13	1.620128	1.693350	1.603296	1.691666	76216000	0.0	0.0
1	2002-02-14	1.712707	1.716073	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
3	2002-02-19	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20	1.615920	1.662210	1.603296	1.662210	6892800	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/IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html>. Save the text of the response as a variable named `html_data`.

```
import requests

# URL of the webpage you want to download
html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html'

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

# Check if the request was successful (status code 200 indicates success)
if response.status_code == 200:
    # Print the content of the webpage (HTML, text, etc.)
    print(response.text)
else:
    print('Failed to download the webpage. Status code:', response.status_code)
```

```
<!DOCTYPE html>
<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/
<html class=" js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange histor
<script type="text/javascript">window.addEventListener('DOMContentLoaded',function(){var v=archive_analytics.values;v.servic
<script type="text/javascript" src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/bundle-playback.js.download" char
<script type="text/javascript" src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/wombat.js.download" charset="utf-
<script type="text/javascript">
  __wm.init("https://web.archive.org/web/");
  __wm.wombat("https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue","20200814131437","https://web.archive.org/","
    "1597410877");
</script>
<link rel="stylesheet" type="text/css" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/banner-styles.css">
<link rel="stylesheet" type="text/css" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/iconochive.css">
<!-- End Wayback Rewrite JS Include -->

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <link rel="canonical" href="https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/ch
  <title>GameStop Revenue 2006-2020 | GME | MacroTrends</title>
  <meta name="description" content="GameStop revenue from 2006 to 2020. Revenue can be defined as the amount o
  <meta name="robots" content="">

  <link rel="shortcut icon" href="https://web.archive.org/web/20200814131437im /https://www.macrotrends.net/as

  <meta name="msvalidate.01" content="1228954C688F5907894001CD8E5E624B">
  <meta name="google-site-verification" content="6MnD_3iDtAP1ZyoGK1YMyVIVck4r5Ws80I9xD3ue4_A">

  <!-- Load in Roboto Font -->
  <link rel="stylesheet" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/css">

  <!-- Bootstrap -->
  <link rel="stylesheet" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/bootstrap.min.css"> <!--
  <link rel="stylesheet" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/bootstrap-theme.min.css">

  <!-- Font Awesome -->
  <link rel="stylesheet" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/font-awesome.min.css"> <

  <!-- JQuery, Bootstrap and Menu Javascript -->
  <script src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/jquery-1.12.4.min.js.download" integrity
  <script src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/bootstrap.min.js.download"></script>

  <!-- Modernizr for cross-browser support -->
  <script type="text/javascript" src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/modernizr-2.6.2-r

  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet" href="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/fuelux.min.css">

  <!-- Latest compiled and minified JavaScript -->
  <script src="/GameStop Revenue 2006-2020 _ GME _ MacroTrends_files/fuelux.min.js.download"></script>

  <!-- Twitter Card data -->
  <meta name="twitter:card" content="summary_large_image">
  <meta name="twitter:site" content="@macrotrends">
  <meta name="twitter:title" content="GameStop Revenue 2006-2020 | GME">
  <meta name="twitter:description" content="GameStop revenue from 2006 to 2020. Revenue can be defined as th

  <!-- Open Graph data -->
```



Parse the html data using `beautiful_soup`.

```
from bs4 import BeautifulSoup

html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork'
# Create a BeautifulSoup object
soup = BeautifulSoup(html_data, 'html.parser')
```

`<ipython-input-42-88faf2005f2d>:5: MarkupResemblesLocatorWarning: The input looks more like a URL than markup. You may wish to use the lxml parser.`

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 using a method similar to what you did in Question 2.

► [Click here if you need help locating the table](#)

```
import pandas as pd
from bs4 import BeautifulSoup
import requests

# Fetch the webpage content
html_data = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork'
response = requests.get(html_data)

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

# using read_html function:
tables = pd.read_html(response.content)
gme_revenue = tables[0] # Assuming the table with GameStop Revenue is at index 0

# Rename columns if necessary
gme_revenue.columns = ['Date', 'Revenue']

# Print the first few rows of the gme_revenue dataframe
print(gme_revenue.head())
```

	Date	Revenue
0	2021	\$53,823
1	2020	\$31,536
2	2019	\$24,578
3	2018	\$21,461
4	2017	\$11,759

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

```
print(gme_revenue.tail())
```

	Date	Revenue
8	2013	\$2,013
9	2012	\$413
10	2011	\$204
11	2010	\$117
12	2009	\$112

```
gme_revenue= pd.DataFrame(columns=["Date","Revenue"])
for row in soup.find_all("tbody")[1].find_all("tr"):
    col= row.find_all('td')
    date= col[0].text
    revenue= col[1].text
    gme_revenue= gme_revenue.append({"Date":date, "Revenue":revenue}, ignore_index = True)
    gme_revenue["Revenue"] = gme_revenue['Revenue'].str.replace(',|\$',"")
```

```
gme_revenue.tail()
```

	Date	Revenue	
49	2010-06-30	28	
50	2010-03-31	21	
51	2009-12-31		
52	2009-09-30	46	
53	2009-06-30	27	

```
import requests
```

```
html_data2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork"
response = requests.get(html_data2)
```

```
if response.status_code == 200:
    html_data = response.text
else:
    print("Failed to retrieve data. Status code:", response.status_code)
```

```
soup = BeautifulSoup(html_data,"html.parser")
```

```
import pandas as pd
```

```
# Assuming 'soup' contains the parsed HTML data
```

```
# Extracting table data and storing in lists
```

```
dates = []
revenues = []
for row in soup.find_all("tbody")[1].find_all("tr"):
    col = row.find_all('td')
    date = col[0].text
    revenue = col[1].text.replace(',','').replace('$','') # Removing commas and dollar signs
    dates.append(date)
    revenues.append(revenue)
```

```
# Creating DataFrame from lists
```

```
gme_revenue = pd.DataFrame({
    "Date": dates,
    "Revenue": revenues
})
```

```
gme_revenue.tail()
```

	Date	Revenue	
57	2006-01-31	1667	
58	2005-10-31	534	
59	2005-07-31	416	
60	2005-04-30	475	
61	2005-01-31	709	

## ✓ Question 5: Plot Tesla Stock Graph

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

```
!pip install nbformat==4.2.0
make_graph(html_data, tesla_revenue, 'tesla_data')
```

Requirement already satisfied: nbformat==4.2.0 in /usr/local/lib/python3.10/dist-packages (4.2.0)  
 Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (0.2.0)  
 Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (4.17.0)  
 Requirement already satisfied: jupyter-core in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (5.5.1)  
 Requirement already satisfied: traitlets>=4.1 in /usr/local/lib/python3.10/dist-packages (from nbformat==4.2.0) (5.7.1)  
 Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (23.1.0)  
 Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (2023.12.1)  
 Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.35.1)  
 Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema!=2.5.0,>=2.4->nbformat==4.2.0) (0.18.0)  
 Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core->nbformat==4.2.0) (3.10.0)

AttributeError Traceback (most recent call last)

<ipython-input-55-224c572ce517> in <cell line: 2>()

```
1 get_ipython().system('pip install nbformat==4.2.0')
----> 2 make_graph(html_data, tesla_revenue, 'tesla_data')
```

<ipython-input-4-47a2266b7876> in make\_graph(stock\_data, revenue\_data, stock)

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

AttributeError: 'str' object has no attribute 'Date'

EXPLAIN ERROR

```
# Assuming the make_graph function is defined somewhere in your code or imported from a module
# Import necessary libraries or functions
import matplotlib.pyplot as plt # For Matplotlib plotting

# Prepare tesla_data and tesla_revenue datasets

# Call the make_graph function with Tesla Stock Data and a title
make_graph(html_data, tesla_revenue, 'Tesla')
plt.title('Tesla Stock Data') # Assign a title to the graph using Matplotlib
plt.show() # Display the graph
```

AttributeError Traceback (most recent call last)

<ipython-input-57-1101dca75353> in <cell line: 8>()

```
6
7 # Call the make_graph function with Tesla Stock Data and a title
----> 8 make_graph(html_data, tesla_revenue, 'Tesla')
9 plt.title('Tesla Stock Data') # Assign a title to the graph using Matplotlib
10 plt.show() # Display the graph
```

<ipython-input-4-47a2266b7876> in make\_graph(stock\_data, revenue\_data, stock)

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

AttributeError: 'str' object has no attribute 'Date'

EXPLAIN ERROR

```
tesla_data['Date'] = pd.to_datetime(tesla_data['Date'])
tesla_revenue['Date'] = pd.to_datetime(tesla_revenue['Date'])

tesla_data_specific = tesla_data[tesla_data['Date'] <= pd.to_datetime('2021-06-14')]
revenue_data_specific = tesla_revenue[tesla_revenue['Date'] <= pd.to_datetime('2021-04-30')]
```

## ✓ 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.

```
make_graph(html_data2, gme_revenue, 'GameStop')
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-53-eb55e7dff001> in <cell line: 1>()
----> 1 make_graph(html_data2, gme_revenue, 'GameStop')

<ipython-input-4-47a2266b7876> in make_graph(stock_data, revenue_data, stock)
      1 def make_graph(stock_data, revenue_data, stock):
      2     fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical
Revenue"), vertical_spacing = .3)
----> 3     stock_data_specific = stock_data[stock_data.Date <= '2021-06-14']
      4     revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
      5     fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date, infer_datetime_format=True),
y=stock_data_specific.Close.astype("float"), name="Share Price"), row=1, col=1)

AttributeError: 'str' object has no attribute 'Date'
```

EXPLAIN ERROR

## 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
2022-02-28	1.2	Lakshmi Holla	Changed the URL of GameStop
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.