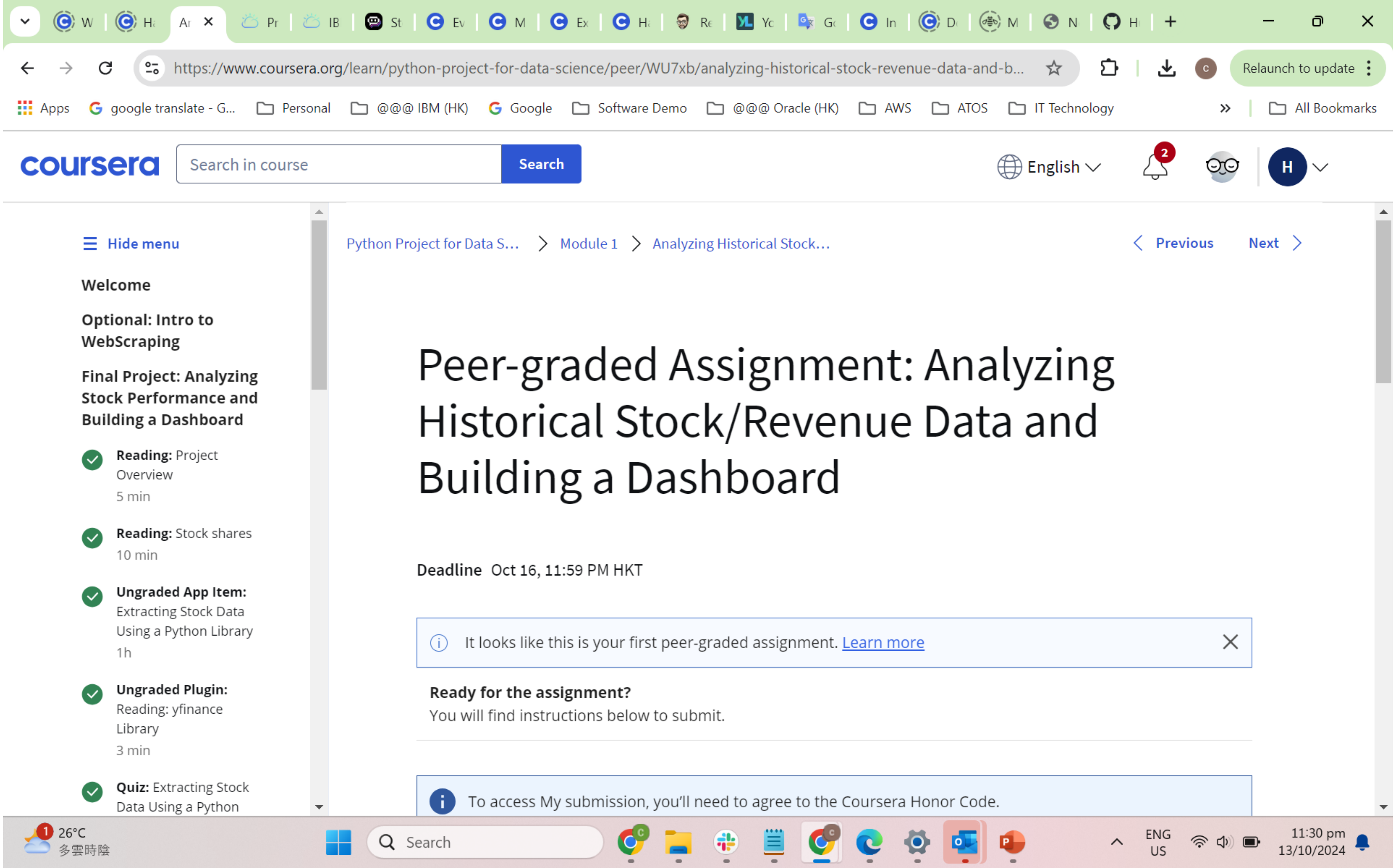

https://www.coursera.org/learn/python-project-for-data-science/peer/WU7xb/analyzing-historical-stock-revenue-data-and-b...

Apps | google translate - G... | Personal | @@@ IBM (HK) | Google | Software Demo | @@@ Oracle (HK) | AWS | ATOS | IT Technology | All Bookmarks

coursera | Search in course | Search | English

Hide menu

Welcome

Optional: Intro to WebScraping

Final Project: Analyzing Stock Performance and Building a Dashboard

- ✓ **Reading:** Project Overview 5 min
- ✓ **Reading:** Stock shares 10 min
- ✓ **Ungraded App Item:** Extracting Stock Data Using a Python Library 1h
- ✓ **Ungraded Plugin:** Reading: yfinance Library 3 min
- ✓ **Quiz:** Extracting Stock Data Using a Python

Python Project for Data S... > Module 1 > Analyzing Historical Stock...

< Previous | Next >

Peer-graded Assignment: Analyzing Historical Stock/Revenue Data and Building a Dashboard

Deadline Oct 16, 11:59 PM HKT

It looks like this is your first peer-graded assignment. Learn more

Ready for the assignment?
You will find instructions below to submit.

To access My submission, you'll need to agree to the Coursera Honor Code.

26°C 多雲時陰 | Search | ENG US | 11:30 pm 13/10/2024

Markdown 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.

(My submission)

Table of Contents

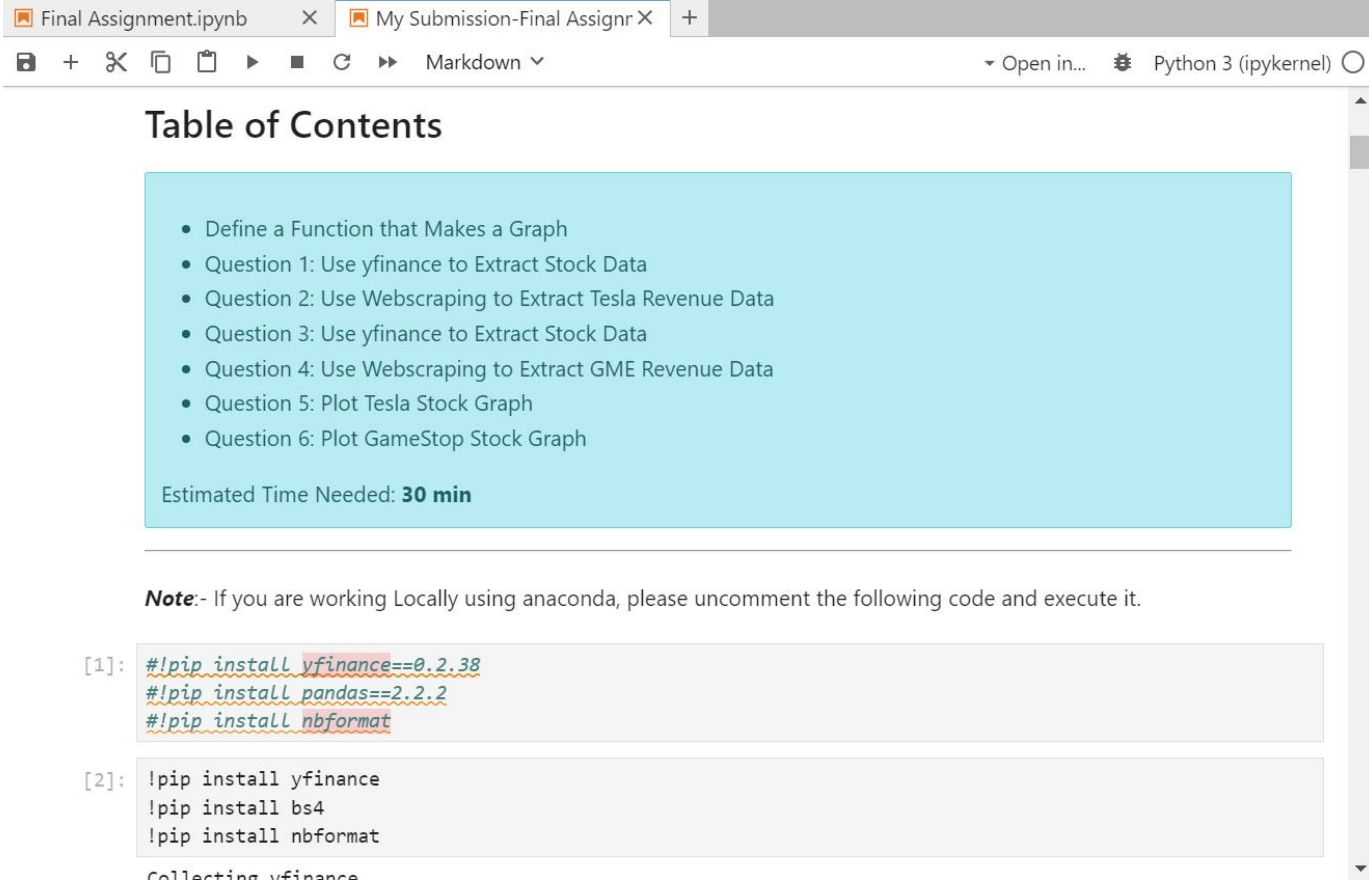


Table of Contents

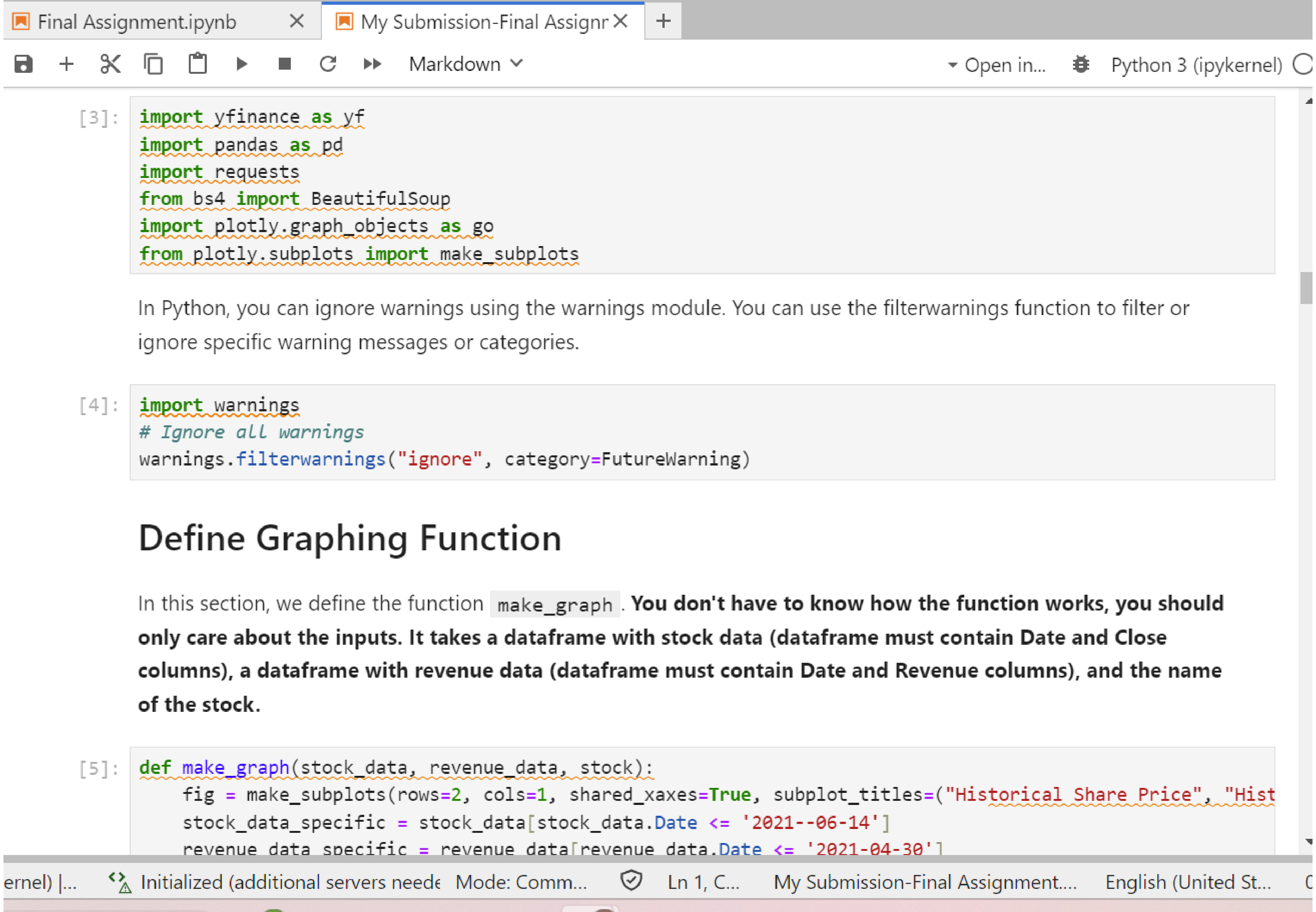
Estimated Time Needed: **30 min**

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
```

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

```
Collecting yfinance
```



```
[3]: 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.

```
[4]: 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.**

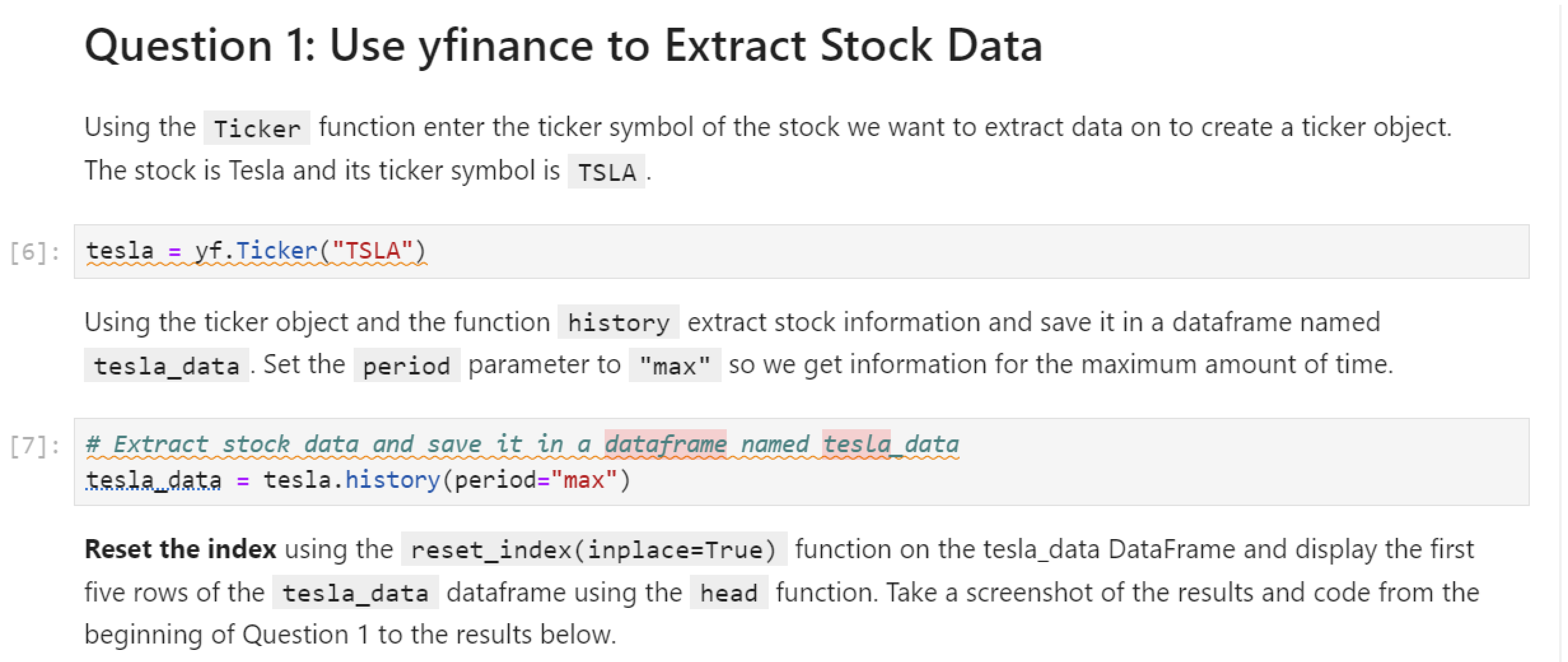
```
[5]: def make_graph(stock_data, revenue_data, stock):
         fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Hist
         stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
         revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
```

ernel) |...
Initialized (additional servers neede
Mode: Comm...
Ln 1, C...
My Submission-Final Assignment....
English (United St...

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", "Hist
       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.astyp
       fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date), y=revenue_data_specific.Revenue
       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

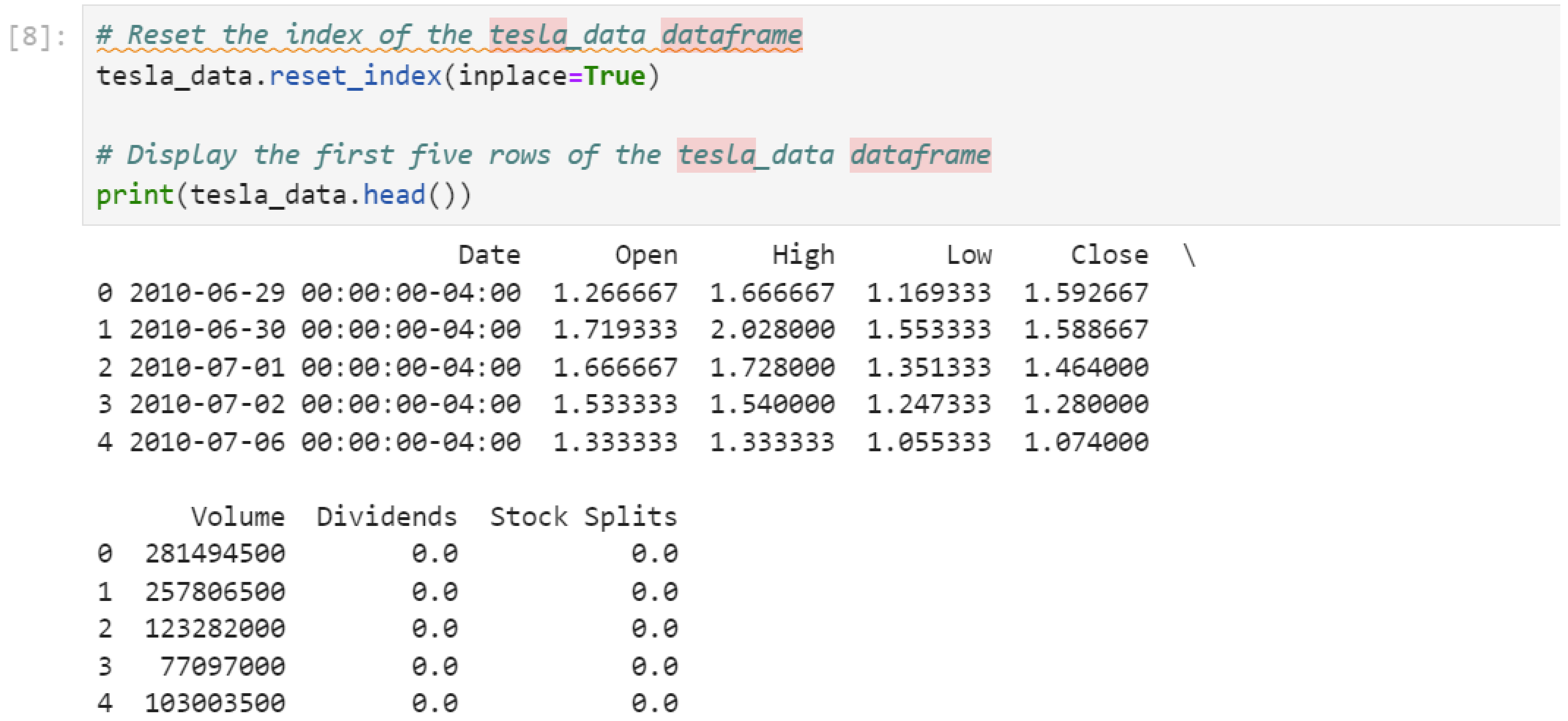
Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

```
[6]: tesla = 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.

```
[7]: # Extract stock data and save it in a dataframe named tesla_data
     tesla_data = tesla.history(period="max")
```

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.



```
[8]: # Reset the index of the tesla_data dataframe
     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`.

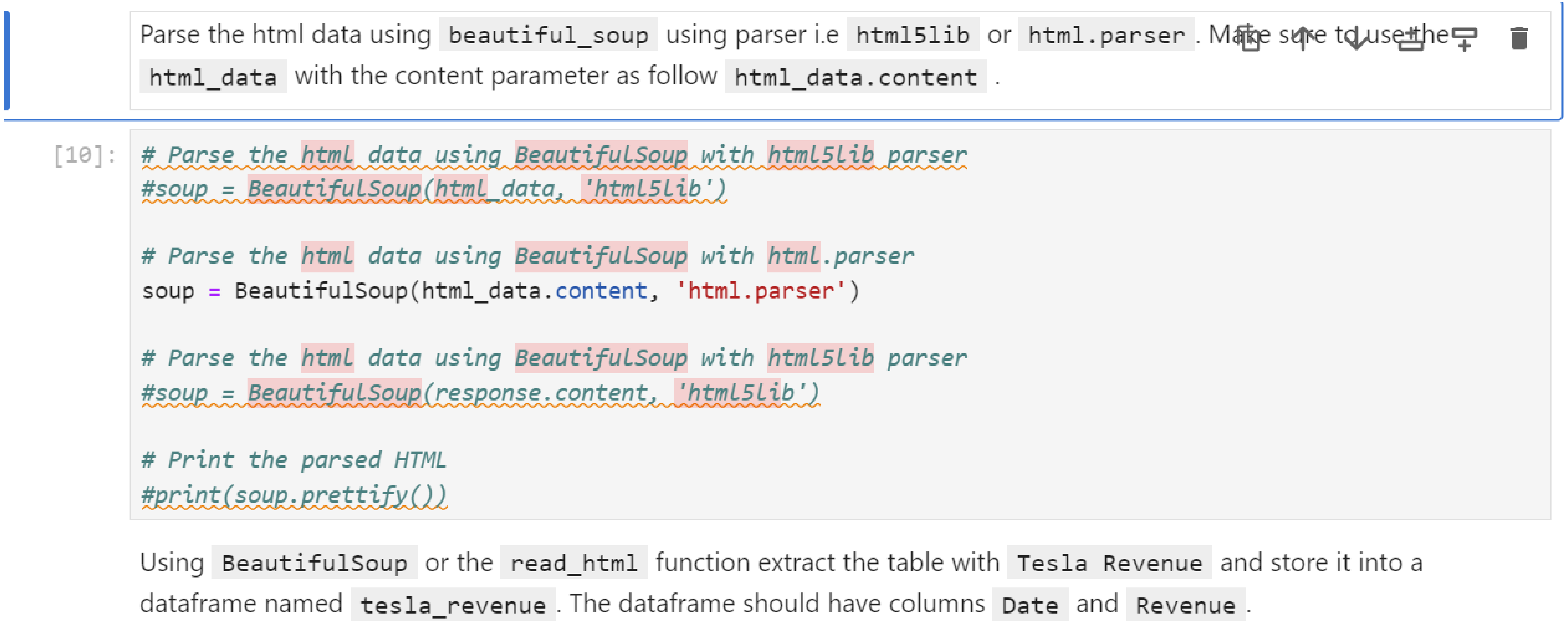
```
import requests

# Send a GET request to the webpage
#response = requests.get('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperS
html_data = requests.get('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperS


# Check if the request was successful
if html_data.status_code == 200:

    print("Webpage downloaded successfully.")
else:
    print("Failed to download the webpage. Status code: ", response.status_code)
```

```
Webpage downloaded successfully.
```



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` .

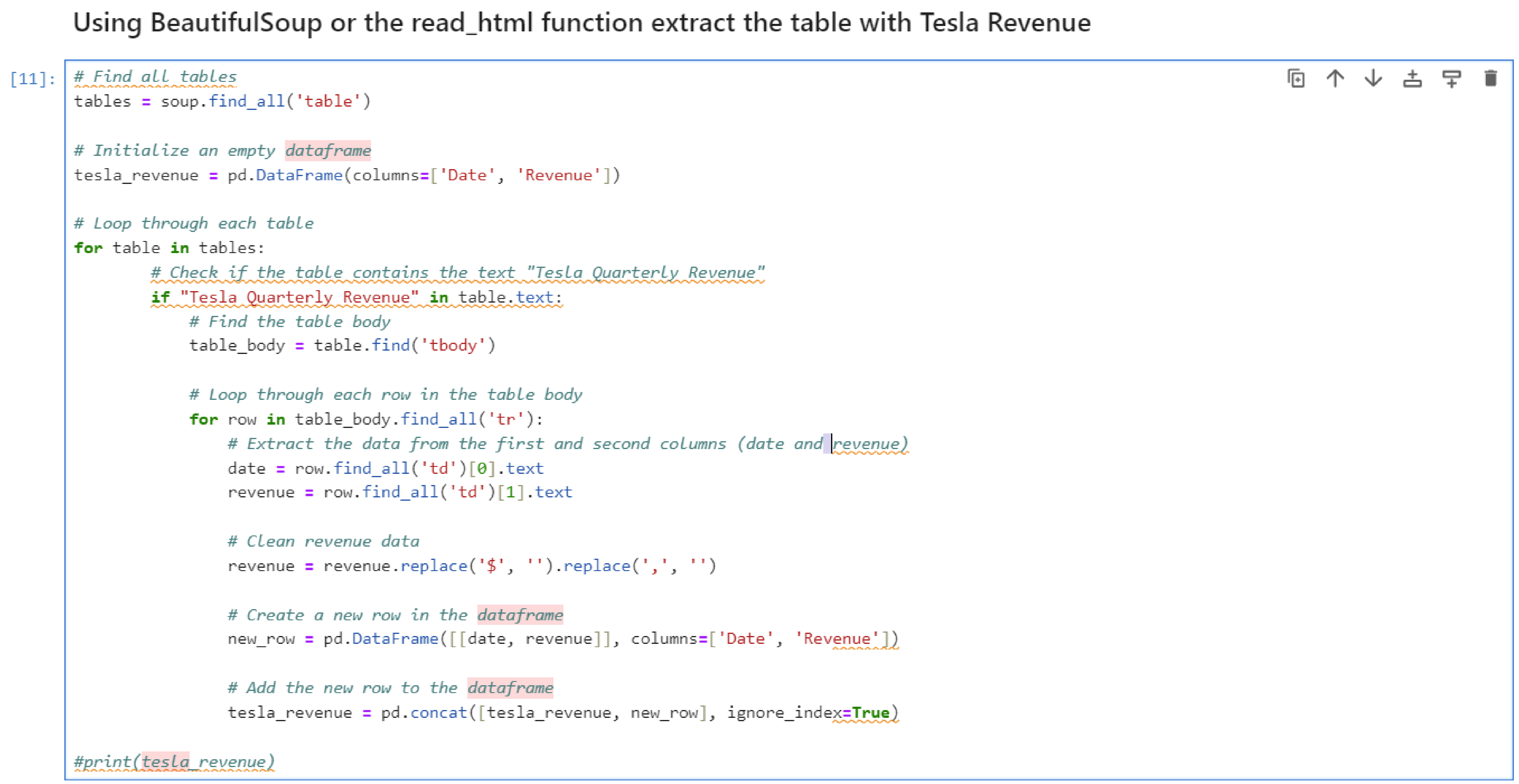
```
[10]: # Parse the html data using BeautifulSoup with html5lib parser
      #soup = BeautifulSoup(html_data, 'html5lib')

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

      # Parse the html data using BeautifulSoup with html5lib parser
      #soup = BeautifulSoup(response.content, 'html5lib')

      # Print the parsed HTML
      #print(soup.prettify())
```

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`.



Using BeautifulSoup or the read_html function extract the table with Tesla Revenue

[11]:
```python
# Find all tables
tables = soup.find_all('table')

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

# Loop through each table
for table in tables:
        # Check if the table contains the text "Tesla Quarterly Revenue"
        if "Tesla Quarterly Revenue" in table.text:
            # Find the table body
            table_body = table.find('tbody')

            # Loop through each row in the table body
            for row in table_body.find_all('tr'):
                # Extract the data from the first and second columns (date and revenue)
                date = row.find_all('td')[0].text
                revenue = row.find_all('td')[1].text

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

                # Create a new row in the dataframe
                new_row = pd.DataFrame([[date, revenue]], columns=['Date', 'Revenue'])

                # Add the new row to the dataframe
                tesla_revenue = pd.concat([tesla_revenue, new_row], ignore_index=True)

#print(tesla_revenue)
```

Alternatively, you can use the read_html function to extract the table:

```
[12]:
 # Read the HTML tables
tables = pd.read_html(html_data.content)

# Select the table at index 1
tesla_revenue = tables[1]

# Rename the columns
tesla_revenue.columns = ['Date', 'Revenue']

# Clean revenue data
#tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace('$', '').str.replace(',', '')

#print(tesla_revenue)
```

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

```
[13]:
# Clean revenue data
# My Code
#tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace('$', '').str.replace(',', '')

# Original Code
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"", regex=True)
#print(tesla_revenue)
```

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

```
[14]:
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.

Alternatively, you can use the read_html function to extract the table:

[12]:
```python
 # Read the HTML tables
tables = pd.read_html(html_data.content)

# Select the table at index 1
tesla_revenue = tables[1]

# Rename the columns
tesla_revenue.columns = ['Date', 'Revenue']

# Clean revenue data
#tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace('$', '').str.replace(',', '')

#print(tesla_revenue)
```

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

[13]:
```python
# Clean revenue data
# My Code
#tesla_revenue['Revenue'] = tesla_revenue['Revenue'].str.replace('$', '').str.replace(',', '')

# Original Code
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"", regex=True)
#print(tesla_revenue)
```

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

[14]:
```python
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.

[14]:
```
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.

[15]:
```
print(tesla_revenue.tail(5))
```

```
          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`.

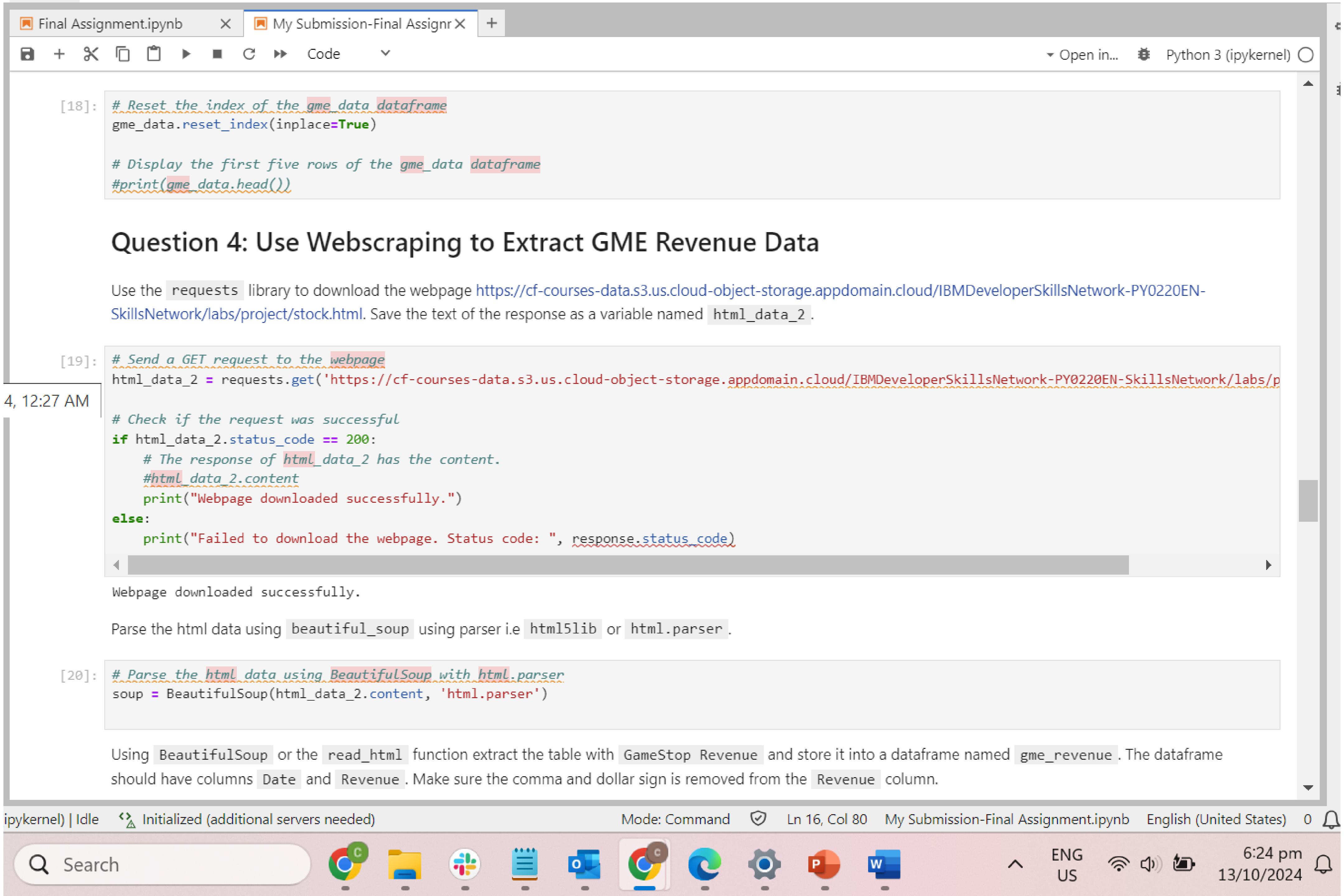
[16]:
```
gamestop = yf.Ticker("GME")
```

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.

[17]:
```
# Extract stock data and save it in a dataframe named gme_data
gme_data = gamestop.history(period="max")

#print(gme_data)
```

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.



[18]:
```
# Reset the index of the gme_data dataframe
gme_data.reset_index(inplace=True)

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

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`.

[19]:
```
# Send a GET request to the webpage
html_data_2 = requests.get('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/p

# Check if the request was successful
if html_data_2.status_code == 200:
    # The response of html_data_2 has the content.
    #html_data_2.content
    print("Webpage downloaded successfully.")
else:
    print("Failed to download the webpage. Status code: ", response.status_code)
```

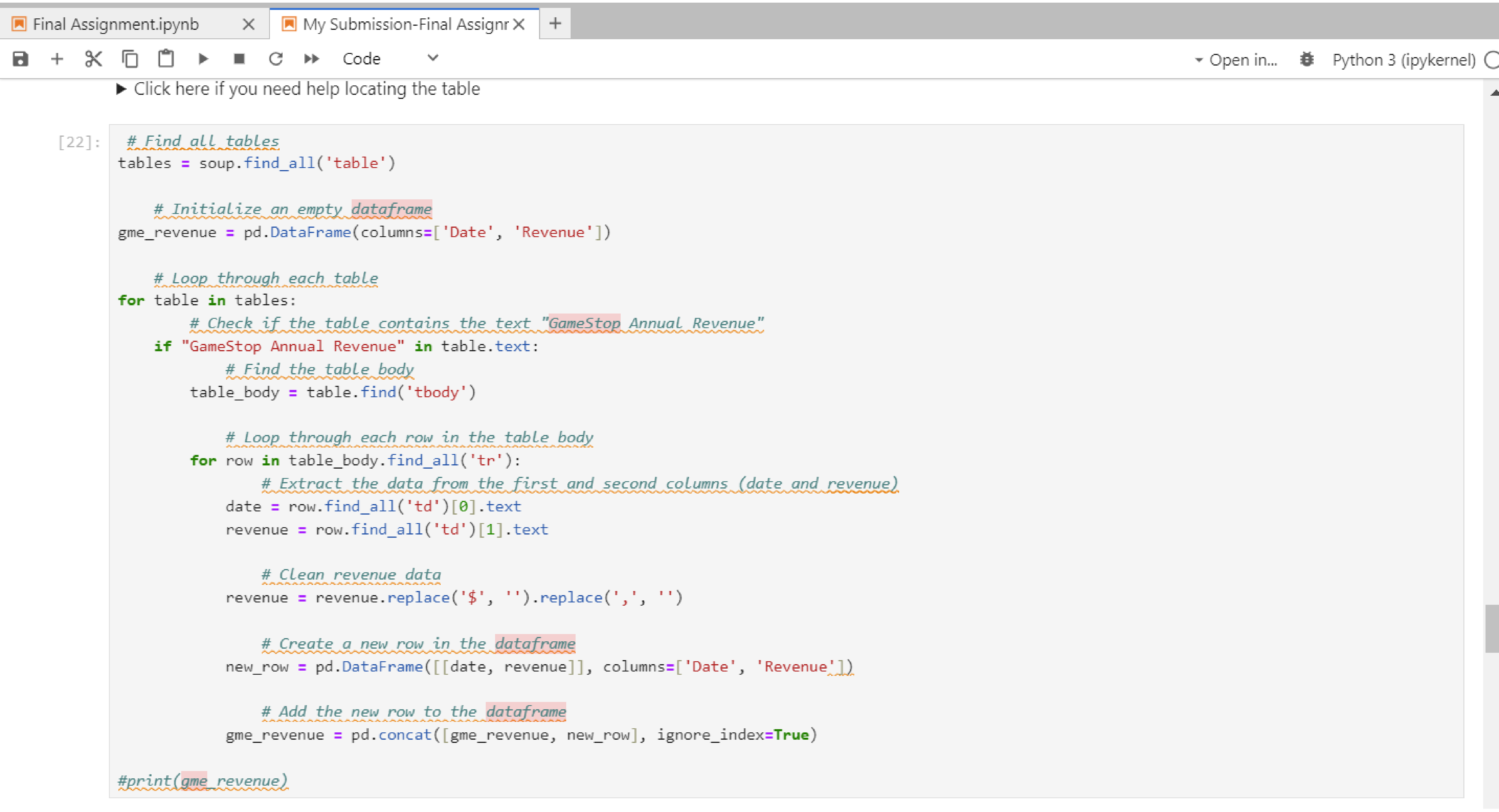
```
Webpage downloaded successfully.
```

4, 12:27 AM

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

[20]:
```
# Parse the html data using BeautifulSoup with html.parser
soup = BeautifulSoup(html_data_2.content, 'html.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.



▶ Click here if you need help locating the table

[22]:
```
 # Find all tables
tables = soup.find_all('table')

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

    # Loop through each table
for table in tables:
        # Check if the table contains the text "GameStop Annual Revenue"
    if "GameStop Annual Revenue" in table.text:
            # Find the table body
        table_body = table.find('tbody')

            # Loop through each row in the table body
        for row in table_body.find_all('tr'):
                # Extract the data from the first and second columns (date and revenue)
            date = row.find_all('td')[0].text
            revenue = row.find_all('td')[1].text

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

                # Create a new row in the dataframe
            new_row = pd.DataFrame([[date, revenue]], columns=['Date', 'Revenue'])

                # Add the new row to the dataframe
            gme_revenue = pd.concat([gme_revenue, new_row], ignore_index=True)

#print(gme_revenue)
```

```
[23]: print(gme_revenue.tail(5))
```

```
    Date Revenue
11  2009    8806
12  2008    7094
13  2007    5319
14  2006    3092
15  2005    1843
```

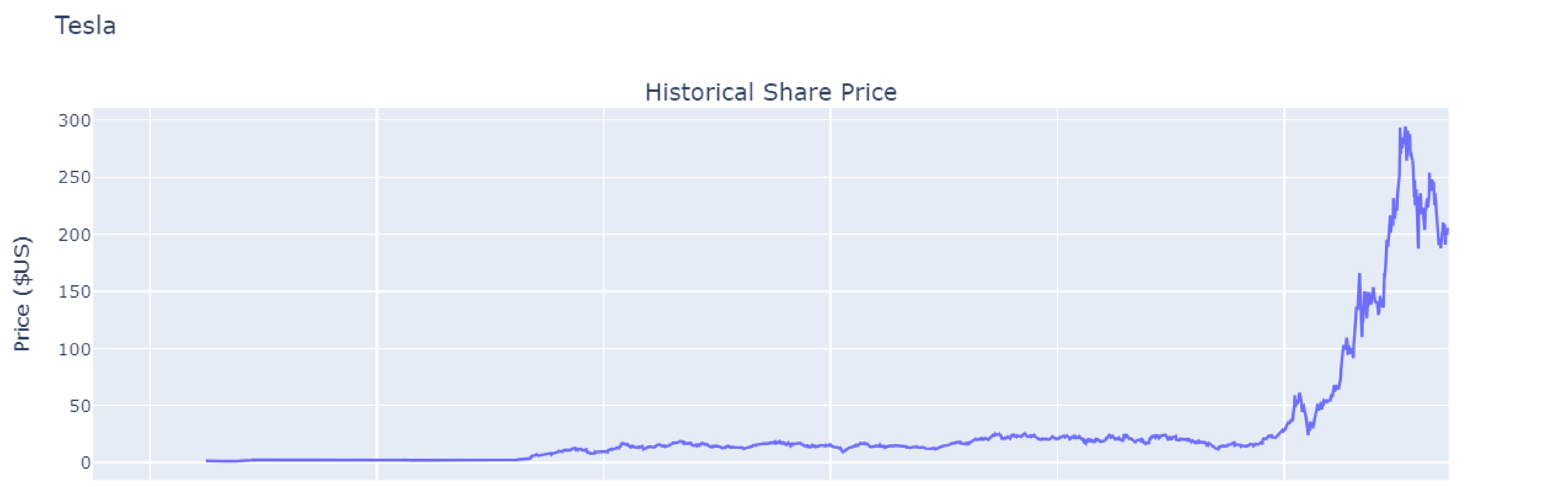
Question 5: Plot Tesla Stock Graph

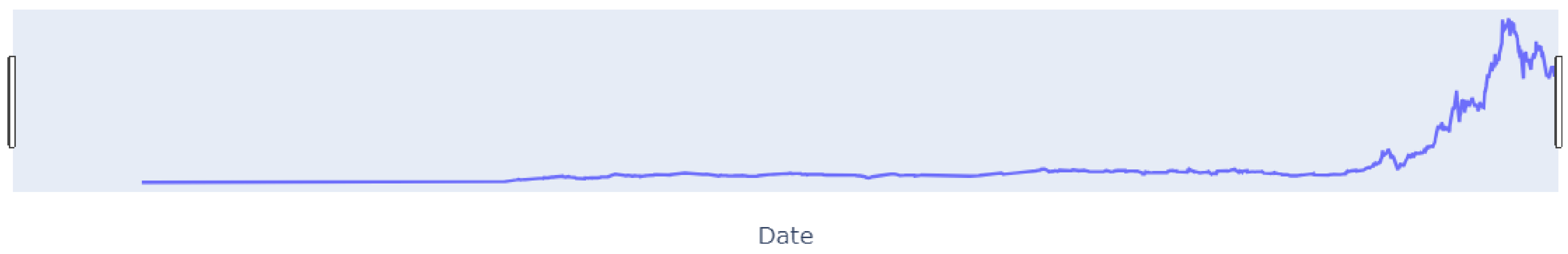
Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

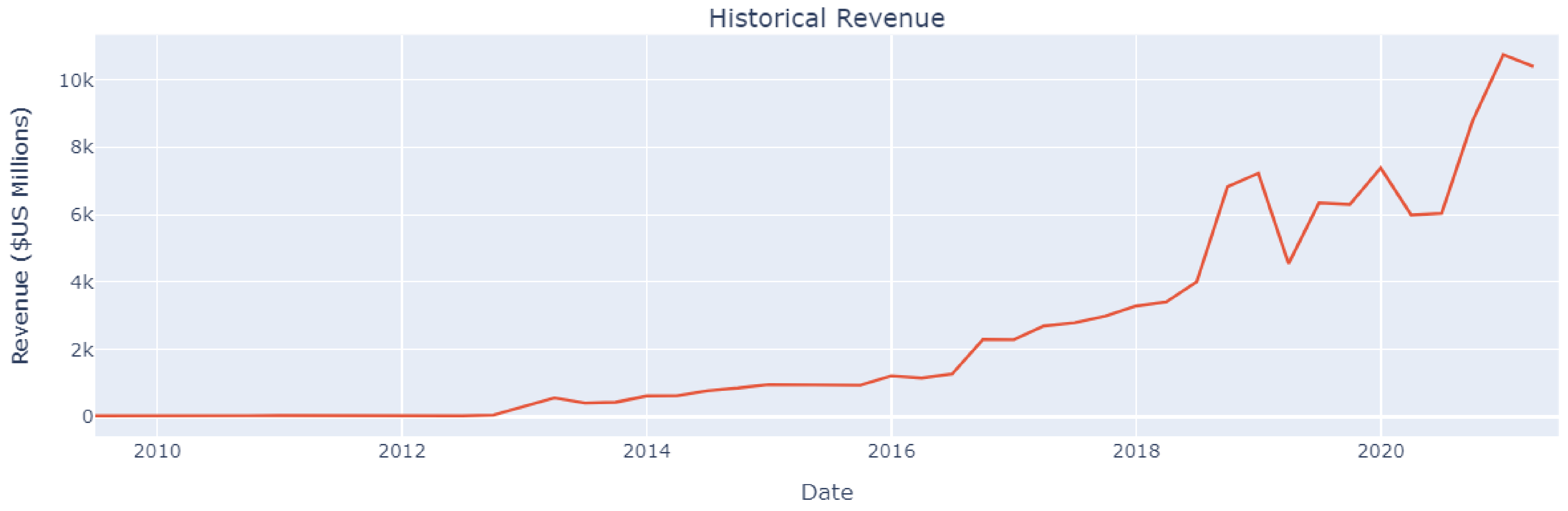
▼ Hint

```
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(tesla_data, tesla_revenue, 'Tesla')`.
```

```
[24]: make_graph(tesla_data, tesla_revenue, 'Tesla')
```




Date

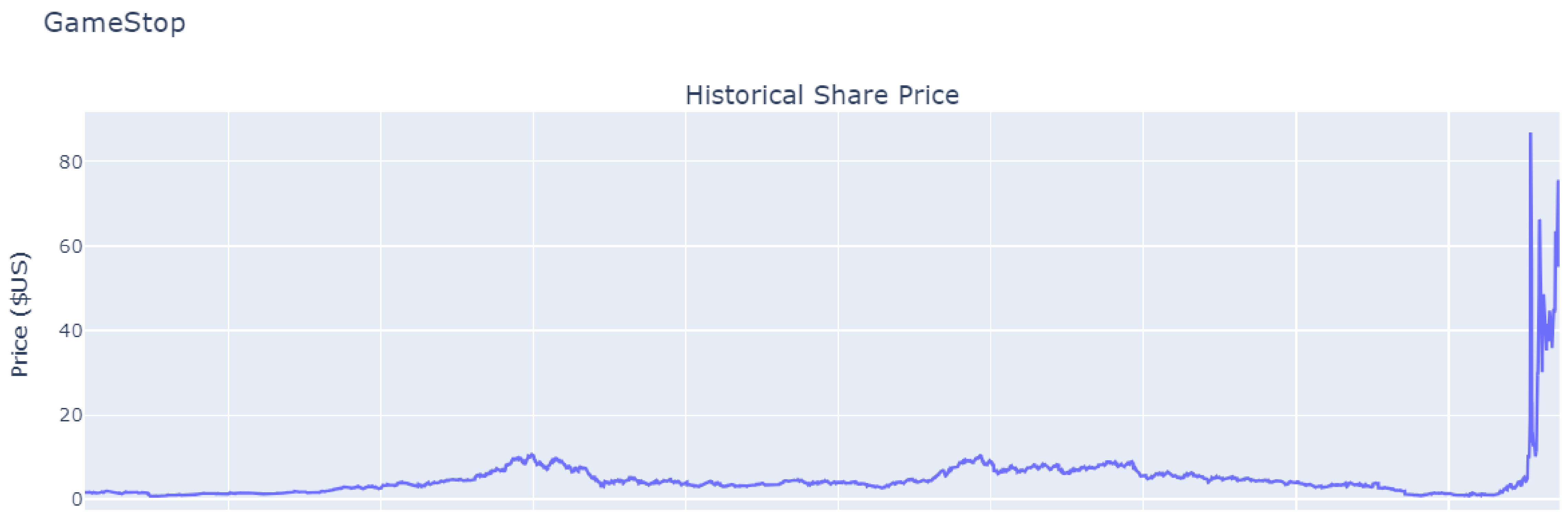

Historical Revenue
Revenue ($US Millions)
10k
8k
6k
4k
2k
0
2010
2012
2014
2016
2018
2020
Date

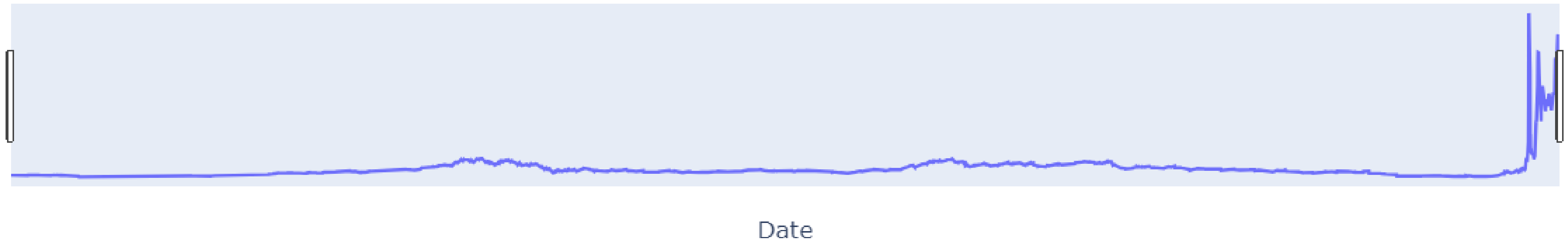
Question 6: Plot GameStop Stock Graph

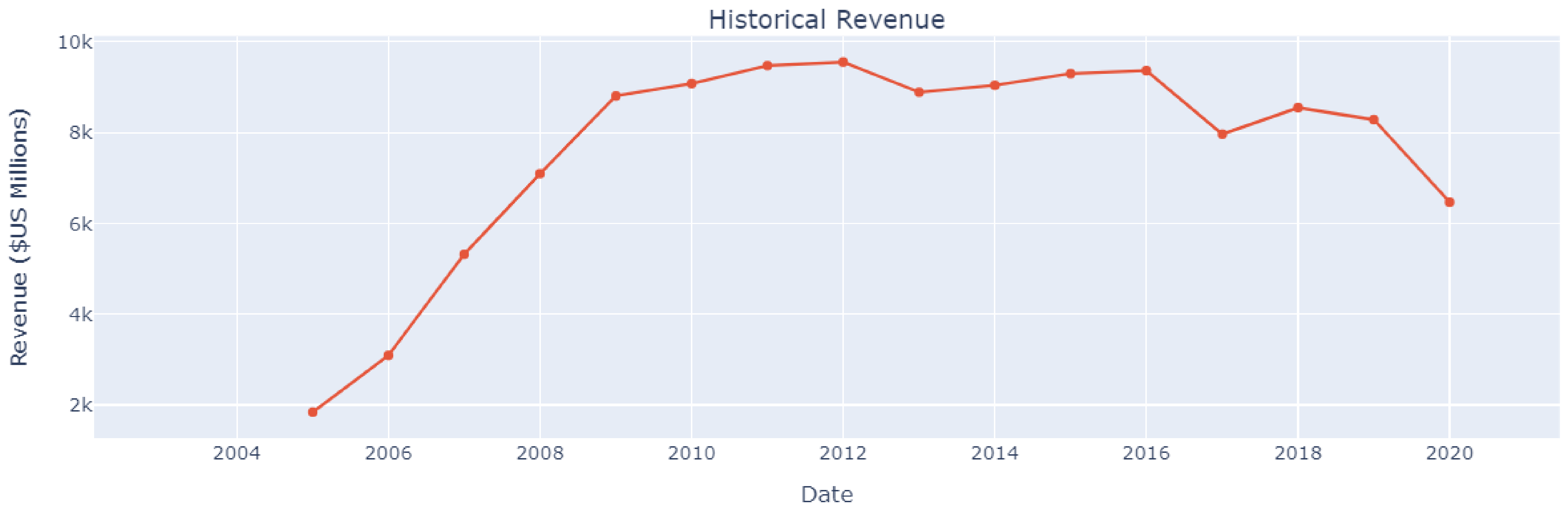
Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')`. Note the graph will only show data upto June 2021.

▶ Hint

```
[25]: make_graph(gme_data, gme_revenue, 'GameStop')
```




Date


Historical Revenue
Revenue ($US Millions)
10k
8k
6k
4k
2k
2004
2006
2008
2010
2012
2014
2016
2018
2020
Date