Skills Network

Extracting Stock Data Using a Web Scraping

Not all stock data is available via the API in this assignment; you will use web-scraping to obtain financial data. You will be quizzed on your results.
You will extract and share historical data from a web page using the BeautifulSoup library.

Table of Contents

Estimated Time Needed: **30 min**

```
!pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4) (2.6)
```

```
!pip install pandas
!pip install requests
!pip install html5lib
!pip install lxml
!pip install plotly
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2024.7.4)
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->bs4) (2.6)
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: html5lib in /usr/local/lib/python3.10/dist-packages (1.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-packages (from html5lib) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from html5lib) (0.5.1)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (4.9.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.15.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly) (24.1)
```

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

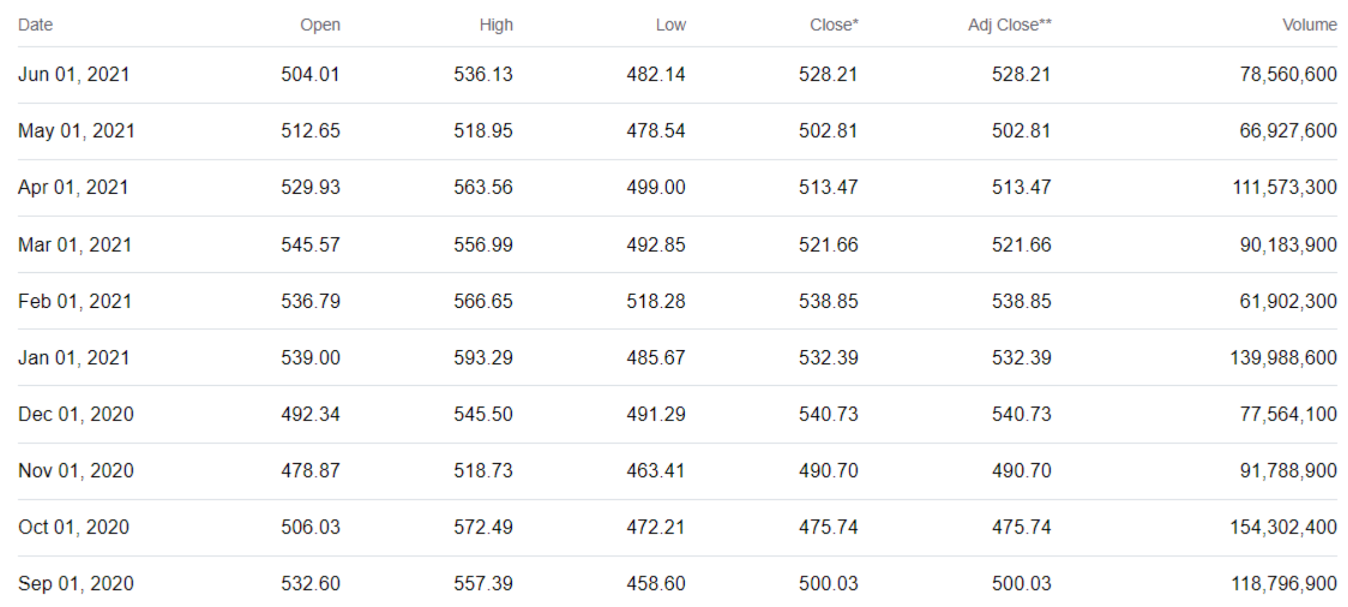
In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

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

Using Webscraping to Extract Stock Data Example

We will extract Netflix stock data https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/netflix_data_webpage.html.

In this example, we are using yahoo finance website and looking to extract Netflix data.



Date	Open	High	Low	Close*	Adj Close**	Volume
Jun 01, 2021	504.01	536.13	482.14	528.21	528.21	78,560,600
May 01, 2021	512.65	518.95	478.54	502.81	502.81	66,927,600
Apr 01, 2021	529.93	563.56	499.00	513.47	513.47	111,573,300
Mar 01, 2021	545.57	556.99	492.85	521.66	521.66	90,183,900
Feb 01, 2021	536.79	566.65	518.28	538.85	538.85	61,902,300
Jan 01, 2021	539.00	593.29	485.67	532.39	532.39	139,988,600
Dec 01, 2020	492.34	545.50	491.29	540.73	540.73	77,564,100
Nov 01, 2020	478.87	518.73	463.41	490.70	490.70	91,788,900
Oct 01, 2020	506.03	572.49	472.21	475.74	475.74	154,302,400
Sep 01, 2020	532.60	557.39	458.60	500.03	500.03	118,796,900

Fig:- Table that we need to extract

On the following web page we have a table with columns name (Date, Open, High, Low, close, adj close volume) out of which we must extract following columns

- Date
- Open
- High
- Low
- Close
- Volume

Steps for extracting the data

1. Send an HTTP request to the web page using the requests library.
2. Parse the HTML content of the web page using BeautifulSoup.
3. Identify the HTML tags that contain the data you want to extract.
4. Use BeautifulSoup methods to extract the data from the HTML tags.
5. Print the extracted data

Step 1: Send an HTTP request to the web page

You will use the request library for sending an HTTP request to the web page.

```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsN
```

The requests.get() method takes a URL as its first argument, which specifies the location of the resource to be retrieved. In this case, the value of the url variable is passed as the argument to the requests.get() method, because you will store a web page URL in a url variable.

You use the .text method for extracting the HTML content as a string in order to make it readable.

```
data  = requests.get(url).text
print(data)
```

```
window.DARLA_CONFIG.onBeforePosMsg = function(msg, posId) {var maxWidth = 970, maxHeight = 600;var newWidth, newHeight, pos;if (window._DarlaEvents && msg === "exp-push" && posId &
window.DARLA_CONFIG.onFinishParse = function(eventName, response) {try {if (eventName !== "AUTO") {var positionlist = response.ps();var foundExclusive = false;for (var i = 0; i < p
};
window.DARLA_CONFIG.onStartPrefetchRequest = function(eventName) {window._perfMark('DARLA_PFSTART');};
window.DARLA_CONFIG.onFinishPrefetchRequest = function(eventName, status) {window._perfMark('DARLA_PFEND');try {window._DarlaEvents.emit('finishprefetch', {status: status,eventName
};
window.DARLA_CONFIG.onPosMsg = function(cmd, pos, msg) {try {if (window._DarlaEvents && cmd === "cmsg") {var posmsg = {pos: pos,msg: msg};window._DarlaEvents.emit("splashmsg", posm
};
(function () {var _onloadEvt = function _onloadEvtHandler() {window._loadEvt = true;if (window._darlaSuccessEvt) {window._fireAdPerfBeacon(window._darlaSuccessEvt);}};if (window.ad
window._DarlaBootNeeded = true;window.$sf = window.sf = {};
window.$sf.host = {onReady: function (autorender, deferrender, firstRenderPos, deferRenderDelay) {window._perfMark('DARLA_ONREADY');window._perfMeasure('DARLA_ONREADY');window.sfre
};
window.sf_host = window.$sf.host;
document.onreadystatechange = function () {if (document.readyState == "interactive") {window._perfMark('DOM_INTERACTIVE');}};</script>
</div><script>
(function (root) {
/* -- Data -- */
root.App || (root.App = {});
```

```
    <script src="https://s.yimg.com/aaq/wf/wf-tooltip-1.1.2-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-beacon-1.3.2-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-caas-1.14.10-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-darla-1.0.26-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-loader-1.7.67-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-sticky-1.0.9-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/wf/wf-template-1.4.1-modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/hp-viewer/desktop_1.9.271.modern.js"></script>
    <script src="https://s.aolcdn.com/membership/omp-static/omp-widgets/2.0.0/switch-widget.prod.js"></script>
    <script src="https://s.yimg.com/uc/finance/dd-site/js/main.bf2a40b0585686531eaf.modern.js" defer></script>
    <script src="https://s.yimg.com/aaq/pv/perf-vitals_2.0.0.js" async></script>
    <script>
            (function () {
                var w = window.wafer || {};
                typeof w.ready === 'function' && w.ready(function () {
                    typeof w.on === 'function' && w.on('tab:selected', function (e) {
```

Step 2: Parse the HTML content

What is parsing?

In simple words, parsing refers to the process of analyzing a string of text or a data structure, usually following a set of rules or grammar, to understand its structure and meaning. Parsing involves breaking down a piece of text or data into its individual components or elements, and then analyzing those components to extract the desired information or to understand their relationships and meanings.

Next you will take the raw HTML content of a web page or a string of HTML code which needs to be parsed and transformed into a structured, hierarchical format that can be more easily analyzed and manipulated in Python. This can be done using a Python library called **Beautiful Soup**.

Parsing the data using the BeautifulSoup library

- Create a new BeautifulSoup object.

  Note: To create a BeautifulSoup object in Python, you need to pass two arguments to its constructor:

1. The HTML or XML content that you want to parse as a string.
2. The name of the parser that you want to use to parse the HTML or XML content. This argument is optional, and if you don't specify a parser, BeautifulSoup will use the default HTML parser included with the library. here in this lab we are using "html5lib" parser.

```
soup = BeautifulSoup(data, 'html.parser')
```

Step 3: Identify the HTML tags

As stated above, the web page consists of a table so, we will scrape the content of the HTML web page and convert the table into a data frame.

You will create an empty data frame using the **pd.DataFrame()** function with the following columns:

- "Date"
- "Open"
- "High"
- "Low"
- "Close"
- "Volume"

```
netflix_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Volume"])
```

Working on HTML table

These are the following tags which are used while creating HTML tables.

- <table>: This tag is a root tag used to define the start and end of the table. All the content of the table is enclosed within these tags.
- <tr>: This tag is used to define a table row. Each row of the table is defined within this tag.
- <td>: This tag is used to define a table cell. Each cell of the table is defined within this tag. You can specify the content of the cell between the opening and closing tags.
- <th>: This tag is used to define a header cell in the table. The header cell is used to describe the contents of a column or row. By default, the text inside a tag is bold and centered.
- <tbody>: This is the main content of the table, which is defined using the tag. It contains one or more rows of elements.

Step 4: Use a BeautifulSoup method for extracting data

We will use **find()** and **find_all()** methods of the BeautifulSoup object to locate the table body and table row respectively in the HTML.

- The *find() method* will return particular tag content.
- The *find_all()* method returns a list of all matching tags in the HTML.

```
# First we isolate the body of the table which contains all the information
# Then we loop through each row and find all the column values for each row
for row in soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
    Open = col[1].text
    high = col[2].text
    low = col[3].text
    close = col[4].text
    adj_close = col[5].text
    volume = col[6].text

    # Finally we append the data of each row to the table
    netflix_data = pd.concat([netflix_data,pd.DataFrame({"Date":[date], "Open":[Open], "High"
```

Step 5: Print the extracted data

We can now print out the data frame using the head() or tail() function.

```
netflix_data.head()
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	Jun 01, 2021	504.01	536.13	482.14	528.21	78,560,600	528.21
1	May 01, 2021	512.65	518.95	478.54	502.81	66,927,600	502.81
2	Apr 01, 2021	529.93	563.56	499.00	513.47	111,573,300	513.47
3	Mar 01, 2021	545.57	556.99	492.85	521.66	90,183,900	521.66
4	Feb 01, 2021	536.79	566.65	518.28	538.85	61,902,300	538.85

Next steps: Generate code with `netflix_data` View recommended plots New interactive sheet

Extracting data using `pandas` library

We can also use the pandas `read_html` function from the pandas library and use the URL for extracting data.

What is read_html in pandas library?

`pd.read_html(url)` is a function provided by the pandas library in Python that is used to extract tables from HTML web pages. It takes in a URL as input and returns a list of all the tables found on the web page.

```
read_html_pandas_data = pd.read_html(url)
```

Or you can convert the BeautifulSoup object to a string.

```
read_html_pandas_data = pd.read_html(str(soup))
```

Because there is only one table on the page, just take the first table in the returned list.

```
netflix_dataframe = read_html_pandas_data[0]

netflix_dataframe.head()
```

	Date	Open	High	Low	Close*	Adj Close**	Volume
0	Jun 01, 2021	504.01	536.13	482.14	528.21	528.21	78560600
1	May 01, 2021	512.65	518.95	478.54	502.81	502.81	66927600
2	Apr 01, 2021	529.93	563.56	499.00	513.47	513.47	111573300
3	Mar 01, 2021	545.57	556.99	492.85	521.66	521.66	90183900
4	Feb 01, 2021	536.79	566.65	518.28	538.85	538.85	61902300

Next steps: Generate code with `netflix_dataframe` View recommended plots New interactive sheet

Exercise: use webscraping to extract stock 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/amazon_data_webpage.html. Save the text of the response as a variable named `html_data`.

```
import requests

url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsN
response = requests.get(url)

html_data = response.text

print(html_data[:500])
```

```
<!DOCTYPE html><html id="atomic" class="NoJs chrome desktop" lang="en-US"><head prefix="og: http://ogp.me/ns#"><script>window.performance && window.performance.mark && window.performa
```

Parse the html data using `beautiful_soup`.

```
from bs4 import BeautifulSoup

# Parse
soup = BeautifulSoup(html_data, "html.parser")


print(soup.prettify()[:500])  # Print the first 500 characters of the prettified HTML
```

```
<!DOCTYPE html>
<html class="NoJs chrome desktop" id="atomic" lang="en-US">
 <head prefix="og: http://ogp.me/ns#">
  <script>
   window.performance && window.performance.mark && window.performance.mark('PageStart');
  </script>
  <meta charset="utf-8"/>
  <title>
   Amazon.com, Inc. (AMZN) Stock Historical Prices &amp; Data - Yahoo Finance
  </title>
  <meta content="AMZN, Amazon.com, Inc., AMZN historical prices, Amazon.com, Inc. historical prices, historical prices, stocks, quotes, finance" na
```

Question 1: What is the content of the title attribute?

```
title_content = soup.title.string

print(f"Title content: {title_content}")
```

```
Title content: Amazon.com, Inc. (AMZN) Stock Historical Prices & Data - Yahoo Finance
```

Using BeautifulSoup, extract the table with historical share prices and store it into a data frame named `amazon_data`. The data frame should have columns Date, Open, High, Low, Close, Adj Close, and Volume. Fill in each variable with the correct data from the list `col`.

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

```
# Download the webpage and parse it
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsN
response = requests.get(url)
html_data = response.text
soup = BeautifulSoup(html_data, "html.parser")

# Create an empty DataFrame with the specified columns
amazon_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close", "Adj Close", "Vol

# Extract data from each row in the table and append it to the DataFrame
for row in soup.find("tbody").find_all("tr"):
    col = row.find_all("td")
    date = col[0].text.strip()
    open_ = col[1].text.strip().replace(",", "")
    high = col[2].text.strip().replace(",", "")
    low = col[3].text.strip().replace(",", "")
    close = col[4].text.strip().replace(",", "")
    adj_close = col[5].text.strip().replace(",", "")
    volume = col[6].text.strip().replace(",", "")

    # Append the row data to the DataFrame
    amazon_data = pd.concat([amazon_data, pd.DataFrame({"Date":[date], "Open":[open_], "High"

# Optionally, display the first few rows of the DataFrame to verify the result
print(amazon_data.head())
```

```
           Date     Open     High      Low    Close Adj Close     Volume
0  Jan 01, 2021  3270.00  3363.89  3086.00  3206.20   3206.20   71528900
1  Dec 01, 2020  3188.50  3350.65  3072.82  3256.93   3256.93   77556200
2  Nov 01, 2020  3061.74  3366.80  2950.12  3168.04   3168.04   90810500
3  Oct 01, 2020  3208.00  3496.24  3019.00  3036.15   3036.15  116226100
4  Sep 01, 2020  3489.58  3552.25  2871.00  3148.73   3148.73  115899300
```

Print out the first five rows of the `amazon_data` data frame you created.

```
print(amazon_data.head())
```

```
           Date     Open     High      Low    Close Adj Close     Volume
0  Jan 01, 2021  3270.00  3363.89  3086.00  3206.20   3206.20   71528900
1  Dec 01, 2020  3188.50  3350.65  3072.82  3256.93   3256.93   77556200
2  Nov 01, 2020  3061.74  3366.80  2950.12  3168.04   3168.04   90810500
3  Oct 01, 2020  3208.00  3496.24  3019.00  3036.15   3036.15  116226100
4  Sep 01, 2020  3489.58  3552.25  2871.00  3148.73   3148.73  115899300
```

Question 2: What are the names of the columns in the data frame?

```
print(amazon_data.columns)
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

Question 3: What is the `Open` of the last row of the amazon_data data frame?

```
last_row_open = amazon_data.iloc[-1]['Open']

# Print the result
print(f"The Open value of the last row is: {last_row_open}")
```

```
The Open value of the last row is: 656.29
```

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
Akansha yadav

<!--

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
02-05-2023	1.3	Akansha yadav	Updated Lab content under maintenance
2021-06-09	1.2	Lakshmi Holla	Added URL in question 3
2020-11-10	1.1	Malika Singla	Deleted the Optional part

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-08-27	1.0	Malika Singla	Added lab to GitLab