**Extracting Stock Data Using a Web Scraping**

Not all stock data is available via API in this assignment; you will use web-scraping to obtain financial data. You will be quizzed on your results.  
Using beautiful soup we will extract historical share data from a web-page.

**Table of Contents**

1. Extracting data using Beautiful soup
   * Downloading the Webpage Using Requests Library
   * Parsing Webpage HTML Using BeautifulSoup
   * Extracting Data and Building DataFrame
2. Extracting data using pandas
3. Exercise

Estimated Time Needed: **30 min**

*#!pip install pandas==1.3.3*

*#!pip install requests==2.26.0*

**!**mamba install bs4**==**4.10.0 **-**y

**!**mamba install html5lib**==**1.1 **-**y

**!**pip install lxml**==**4.6.4

*#!pip install plotly==5.3.1*

**import** pandas **as** pd

**import** requests

**from** bs4 **import** BeautifulSoup

**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.**

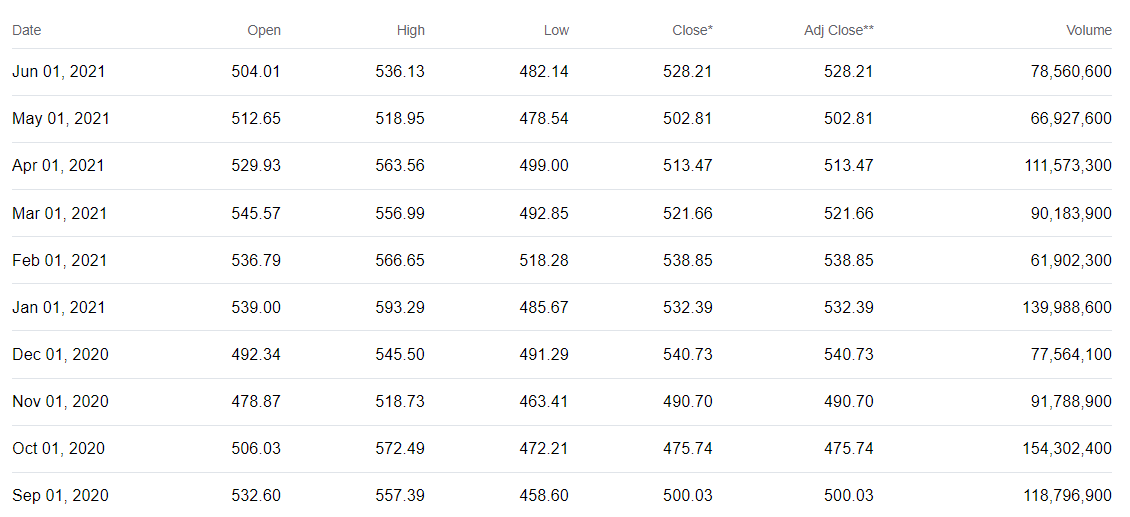


Fig:- Table that we need to extract

On the following webpage 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 to be followed for extracting data**

1. Send an HTTP request to the webpage using the requests library.
2. Parse the HTML content of the webpage 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 webpage**

We are using Request library for sending an HTTP request to the webpage.

url **=** "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/netflix\_data\_webpage.html"

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, as we've stored a webpage URL in a url variable.

we have used .text method for extracting the HTML content as a string in order to make it readable.

data **=** requests.get(url).text

print(data)

**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 we will take the raw HTML content of a webpage 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**.

**How to parse the data using Beautiful soup library?**

* Create a new Beautiful soup object.  
    
  **Note:-**To create a Beautiful Soup 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, Beautiful Soup will use the default HTML parser included with the library. here in this lab we are using "html5lib" parser.

soup **=** BeautifulSoup(data, 'html5lib')

**Step-3 Identify the HTML tags**

As stated above webpage consist of table so, we will be scrapping the content of the HTML webpage and convert the table into a dataframe.

We will creates an empty DataFrame 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> tag: This tag is root tag used to define the start and end of the table. All the content of the table is enclosed within these tags.
* <tr> tag: This tag is used to define a table row. Each row of the table is defined within this tag.
* <td> tag: 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> tag: 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> tag: 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 Beautiful soup 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

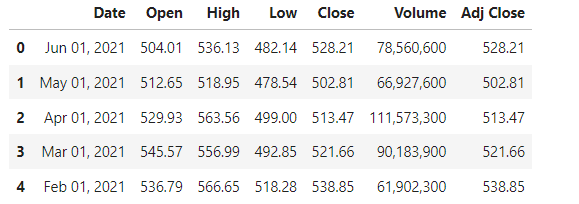
*# Finally we append the data of each row to the table*

netflix\_data **=** netflix\_data.append({"Date":date, "Open":Open, "High":high, "Low":low, "Close":close, "Adj Close":adj\_close, "Volume":volume}, ignore\_index**=True**)

**Step-5 Print the Extracted Data**

We can now print out the DataFrame using head() or tail() function

netflix\_data.head()



**Extracting data using pandas library**

We can also use the pandas read\_html function from 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 webpage.

read\_html\_pandas\_data **=** pd.read\_html(url)

Or we 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, we just take the first table in the list returned

netflix\_dataframe **=** read\_html\_pandas\_data[0]

​

netflix\_dataframe.head()

**Using Webscraping to Extract Stock Data Exercise**

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.

​

Parse the html data using beautiful\_soup.

​

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

​

Using beautiful soup extract the table with historical share prices and store it into a dataframe named amazon\_data. The dataframe should have columns Date, Open, High, Low, Close, Adj Close, and Volume. Fill in each variable with the correct data from the list col.

amazon\_data **=** pd.DataFrame(columns**=**["Date", "Open", "High", "Low", "Close", "Volume"])

​

**for** row **in** soup.find("tbody").find\_all("tr"):

col **=** row.find\_all("td")

date **=** *#ADD\_CODE*

Open **=** *#ADD\_CODE*

high **=** *#ADD\_CODE*

low **=** *#ADD\_CODE*

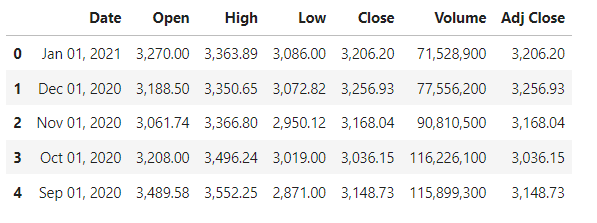
close **=** *#ADD\_CODE*

adj\_close **=** *#ADD\_CODE*

volume **=** *#ADD\_CODE*

amazon\_data **=** amazon\_data.append({"Date":date, "Open":Open, "High":high, "Low":low, "Close":close, "Adj Close":adj\_close, "Volume":volume}, ignore\_index**=True**)

Print out the first five rows of the amazon\_data dataframe you created.



​

**Question 2** What is the name of the columns of the dataframe?

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

​

**Question 3** What is the Open of the last row of the amazon\_data dataframe?

