

Final Assignment_Web scraping_completed-Copy1

May 20, 2022

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

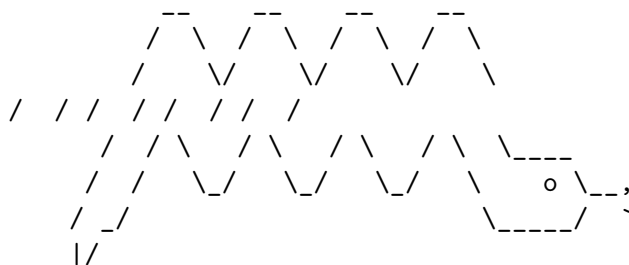
Downloading the Webpage Using Requests Library

Parsing Webpage HTML Using BeautifulSoup

Extracting Data and Building DataFrame

Estimated Time Needed: 30 min

```
[5]: #!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
```



mamba (0.22.1) supported by @QuantStack

GitHub: <https://github.com/mamba-org/mamba>
Twitter: <https://twitter.com/QuantStack>

Looking for: ['bs4==4.10.0']

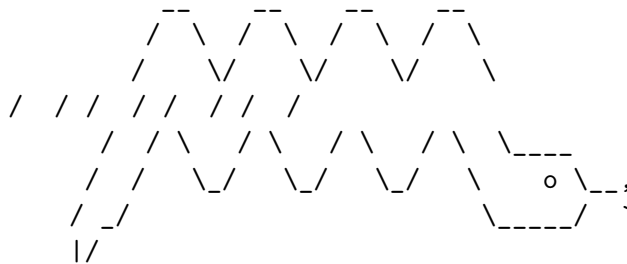
```
[+] 0.0s
pkgs/main/linux-64                                0.0 B
/   ???.?MB @   ???.?MB/s  0.0s[+] 0.1s
pkgs/main/linux-64                                0.0 B
/   ???.?MB @   ???.?MB/s  0.1s
pkgs/main/noarch                                  0.0 B /   ???.?MB
@   ???.?MB/s  0.1s
pkgs/r/linux-64                                   0.0 B /   ???.?MB
@   ???.?MB/s  0.1s
pkgs/r/noarch                                     0.0 B /   ???.?MB
@   ???.?MB/s  0.1s
pkgs/main/linux-64
No change
pkgs/main/noarch                                No change
pkgs/r/noarch                                   No change
pkgs/r/linux-64                                No change

Pinned packages:
- python 3.7.*
```

Transaction

Prefix: /home/jupyterlab/conda/envs/python

All requested packages already installed



mamba (0.22.1) supported by @QuantStack

GitHub: <https://github.com/mamba-org/mamba>

Twitter: <https://twitter.com/QuantStack>

Looking for: ['html5lib==1.1']

pkgs/main/linux-64	Using cache
pkgs/main/noarch	Using cache
pkgs/r/linux-64	Using cache
pkgs/r/noarch	Using cache

Pinned packages:

- python 3.7.*

Transaction

Prefix: /home/jupyterlab/conda/envs/python

All requested packages already installed

Requirement already satisfied: lxml==4.6.4 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (4.6.4)

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

0.1 Using Webscraping to Extract Stock Data Example

First we must use the `request` library to download the webpage, and extract the text. 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.

```
[7]: url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↳ IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/
↳ netflix_data_webpage.html"

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

Next we must parse the text into html using beautiful_soup

```
[8]: soup = BeautifulSoup(data, 'html5lib')
```

Now we can turn the html table into a pandas dataframe

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

We can now print out the dataframe

```
[ ]: netflix_data.head()
```

We can also use the pandas read_html function using the url

```
[ ]: 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))
```

Beacause 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()
```

0.2 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/>

IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/amazon_data_webpage.html. Save the text of the response as a variable named `html_data`.

```
[9]: url2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↳IBMDDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/
↳amazon_data_webpage.html"
```

```
[12]: data2 = requests.get(url2).text
```

Parse the html data using `beautiful_soup`.

```
[13]: soup2 = BeautifulSoup(data2, 'html5lib')
```

Question 1 What is the content of the title attribute:

```
[15]: tag_object=soup2.title
print("tag object:",tag_object)
```

tag object: <title>Amazon.com, Inc. (AMZN) Stock Historical Prices & Data - Yahoo Finance</title>

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.

```
[16]: amazon_data = pd.DataFrame(columns=["Date", "Open", "High", "Low", "Close",
↳"Volume"])

for row in soup2.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

    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.

```
[17]: amazon_data.head()
```

```
[17]:
```

	Date	Open	High	Low	Close	Volume	Adj Close
0	Jan 01, 2021	3,270.00	3,363.89	3,086.00	3,206.20	71,528,900	3,206.20
1	Dec 01, 2020	3,188.50	3,350.65	3,072.82	3,256.93	77,556,200	3,256.93

```

2 Nov 01, 2020 3,061.74 3,366.80 2,950.12 3,168.04 90,810,500 3,168.04
3 Oct 01, 2020 3,208.00 3,496.24 3,019.00 3,036.15 116,226,100 3,036.15
4 Sep 01, 2020 3,489.58 3,552.25 2,871.00 3,148.73 115,899,300 3,148.73

```

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

```
[19]: amazon_data.columns
```

```
[19]: 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?

```
[20]: amazon_data.tail()
```

```
[20]:
      Date      Open      High      Low      Close      Volume  Adj Close
56 May 01, 2016 663.92 724.23 656.00 722.79 90,614,500 722.79
57 Apr 01, 2016 590.49 669.98 585.25 659.59 78,464,200 659.59
58 Mar 01, 2016 556.29 603.24 538.58 593.64 94,009,500 593.64
59 Feb 01, 2016 578.15 581.80 474.00 552.52 124,144,800 552.52
60 Jan 01, 2016 656.29 657.72 547.18 587.00 130,200,900 587.00

```

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

0.3 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
-------------------	---------	------------	--------------------

2021-06-09	1.2	Lakshmi Holla	Added URL in question 3
------------	-----	---------------	-------------------------

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.