1 - Installing bs4 module

In [2]:
```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\users\mayan\appdata\local\programs\python\pytho
n38-32\lib\site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in c:\users\mayan\appdata\local\programs\p
ython\python38-32\lib\site-packages (from bs4) (4.9.3)
Requirement already satisfied: soupsieve>1.2; python_version >= "3.0" in c:\users\mayan
\appdata\local\programs\python\python38-32\lib\site-packages (from beautifulsoup4->bs4)
(2.0.1)
```

In [3]:
```
import bs4
```

2 - Installing requests module

In [4]:
```
!pip install requests
```

```
Requirement already satisfied: requests in c:\users\mayan\appdata\local\programs\python
\python38-32\lib\site-packages (2.24.0)
Requirement already satisfied: idna<3,>=2.5 in c:\users\mayan\appdata\local\programs\pyt
hon\python38-32\lib\site-packages (from requests) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\mayan\appdata\local\program
s\python\python38-32\lib\site-packages (from requests) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mayan\appdata\local\progra
ms\python\python38-32\lib\site-packages (from requests) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\mayan
\appdata\local\programs\python\python38-32\lib\site-packages (from requests) (1.25.11)
```

In [5]:
```
import requests
```

In [18]:
```
# send a request and receive the information from https://www.google.com
response = requests.get("https://www.google.com")

# printing the response
print(response.content)
```

In [11]:
```
print(response.headers)
```

```
{'Date': 'Fri, 04 Dec 2020 14:43:26 GMT', 'Expires': '-1', 'Cache-Control': 'private, ma
x-age=0', 'Content-Type': 'text/html; charset=ISO-8859-1', 'P3P': 'CP="This is not a P3P
policy! See g.co/p3phelp for more info."', 'Content-Encoding': 'gzip', 'Server': 'gws',
'X-XSS-Protection': '0', 'X-Frame-Options': 'SAMEORIGIN', 'Set-Cookie': '1P_JAR=2020-12-
04-14; expires=Sun, 03-Jan-2021 14:43:26 GMT; path=/; domain=.google.com; Secure, NID=20
4=YR4aYsZGVOuEFMDeyzr_PwPGRdmaDExgmpl7IIHXzdyV4--SuRVeXyJbitl3yr06aZGR3S1-7nI-VL8iJZPMJK
0B9Xza5SaRev5Hx_ihOlFTlXAzz5Uf4kA3t71xAiMCtVVTDq-t_VgdggSPjZlO8o9QUTG4T3fhFRt9HwRxMGE; e
xpires=Sat, 05-Jun-2021 14:43:26 GMT; path=/; domain=.google.com; HttpOnly', 'Alt-Svc':
'h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046
=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"', 'Tra
nsfer-Encoding': 'chunked'}
```

In [12]:
```
for key, value in response.headers.items():
    print(key, '\t\t', value)
```

```
Date            Fri, 04 Dec 2020 14:43:26 GMT
Expires                 -1
Cache-Control           private, max-age=0
Content-Type            text/html; charset=ISO-8859-1
P3P             CP="This is not a P3P policy! See g.co/p3phelp for more info."
Content-Encoding                gzip
```

```
Server                  gws
X-XSS-Protection                0
X-Frame-Options                 SAMEORIGIN
Set-Cookie              1P_JAR=2020-12-04-14; expires=Sun, 03-Jan-2021 14:43:26 GMT; pa
th=/; domain=.google.com; Secure, NID=204=YR4aYsZGVOuEFMDeyzr_PwPGRdmaDExgmpl7IIHXzdyV4-
-SuRVeXyJbitl3yr06aZGR3S1-7nI-VL8iJZPMJK0B9Xza5SaRev5Hx_ihOlFTlXAzz5Uf4kA3t71xAiMCtVVTDq
-t_VgdggSPjZlO8o9QUTG4T3fhFRt9HwRxMGE; expires=Sat, 05-Jun-2021 14:43:26 GMT; path=/; do
main=.google.com; HttpOnly
Alt-Svc                 h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":4
43"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=25
92000; v="46,43"
Transfer-Encoding               chunked
```

In [13]:
```
# Status of Request
print(response.status_code)
```

```
200
```

3 - Setup User Agent

In [15]:
```
!pip install fake_useragent
```

```
Requirement already satisfied: fake_useragent in c:\users\mayan\appdata\local\programs\p
ython\python38-32\lib\site-packages (0.1.11)
```

In [17]:
```
# import UserAgent from the fake_useragent module
from fake_useragent import UserAgent

# create an instance of the 'UserAgent' class
obj = UserAgent()

# create a dictionary with key 'user-agent' and value 'obj.chrome'
header = {'user-agent': obj.chrome}

# send request by passing 'header' to the 'headers' parameter in 'get' method
r = requests.get('https://google.com', headers=header)

print(r.content)
```

4 - BeautifulSoup: Prettify Content

In [20]:
```
# import modules
import requests

# importing the beautifulsoup module
import bs4

# send a request and receive the information from https://www.google.com
response = requests.get("https://www.google.com")

# creating BeautifulSoup object
soup = bs4.BeautifulSoup(response.content, "html.parser")

# using 'prettify' method to print the content
print(soup.prettify())
```

5 - BeautifulSoup: Accessing HTML Tags

In [28]:
```
# import modules
```

```
import requests

# importing the beautifulsoup module
import bs4

# send a request and receive the information from https://www.google.com
response = requests.get("https://www.google.com")

# creating BeautifulSoup object
soup = bs4.BeautifulSoup(response.content, "html.parser")

# getting 'title' tag from the google BeautifulSoup -> 'soup'
print(soup.title.text)
```

```
Google
```

6 - BeautifulSoup: contents method

In [30]:
```
body = soup.body

# getting all the children of 'body' using 'contents'
content_list = body.contents

# printing all the children using for loop
for tag in content_list:
    if tag != "\n":
        print(tag)
        print("\n")
```

7 - BeautifulSoup: children method

In [32]:
```
body = soup.body

## we can also convert iterator into list using the 'list(iterator)'
for tag in body.children:
    if tag != "\n":
        print(tag)
        print("\n")
```

8 - BeautifulSoup: descendants method

In [44]:
```
body = soup.body

## we can also convert iterator into list using the 'list(iterator)'
for tag in body.descendants:
    if tag != "\n":
        print(tag)
        print("\n")
```

9 - BeautifulSoup: parent method

In [37]:
```
body = soup.body

# getting parent of 'body'
body_parent = body.parent
```

```
# you have to use 'name' method to print the name of the tag
# printing the name of the parent using 'name' method
print(body_parent.name)
```

```
html
```

10 - BeautifulSoup: find_all method

In [42]:
```
# finding all p tags
p_tags = soup.find_all("p")
print(p_tags)
```

```
[<p style="font-size:8pt;color:#70757a">© 2020 - <a href="/intl/en/policies/privacy/">Pr
ivacy</a> - <a href="/intl/en/policies/terms/">Terms</a></p>]
```

11 - BeautifulSoup: find method

In [41]:
```
p_tag = soup.find("p")

print(p_tag)
print(p_tag.text)
```

```
<p style="font-size:8pt;color:#70757a">© 2020 - <a href="/intl/en/policies/privacy/">Pri
vacy</a> - <a href="/intl/en/policies/terms/">Terms</a></p>
© 2020 - Privacy - Terms
```

12 - Writing Data to CSV File

In [45]:
```
# importing bs4, requests, fake_useragent and csv modules
import bs4
import requests
from fake_useragent import UserAgent
import csv

# initializing the UserAgent object
user_agent = UserAgent()
url = "https://www.consumerreports.org/cro/a-to-z-index/products/index.htm"

# getting the reponse from the page using get method of requests module
page = requests.get(url, headers={"user-agent": user_agent.chrome})

# storing the content of the page in a variable
html = page.content

# creating BeautifulSoup object
soup = bs4.BeautifulSoup(html, "html.parser")

# div tags with crux-body-copy class
div_class = "crux-body-copy"

# getting all the divs with class 'crux-body-copy'
div_tags = soup.find_all("div", class_="div_class")

# then we open a csv file in append mode
with open("product_data.csv", "a") as csv_file:
    writer = csv.writer(csv_file)

    # extracting the names and links from the div tags
```

```
    for tag in div_tags:
        name = tag.a.text.strip()
        link = tag.a['href']
        ## now we will write data to the file
        writer.writerow([name, link])
```