Web Scrapping

What is Web scraping & How does it work.

Installation of Python and packages in Windows.

How to view HTML source code in Google Chrome.

Request Library for python for web scraping

How to parse HTML content using Beautiful Soup Library

How obtain HTML using BeautifulSoup

Requests Module

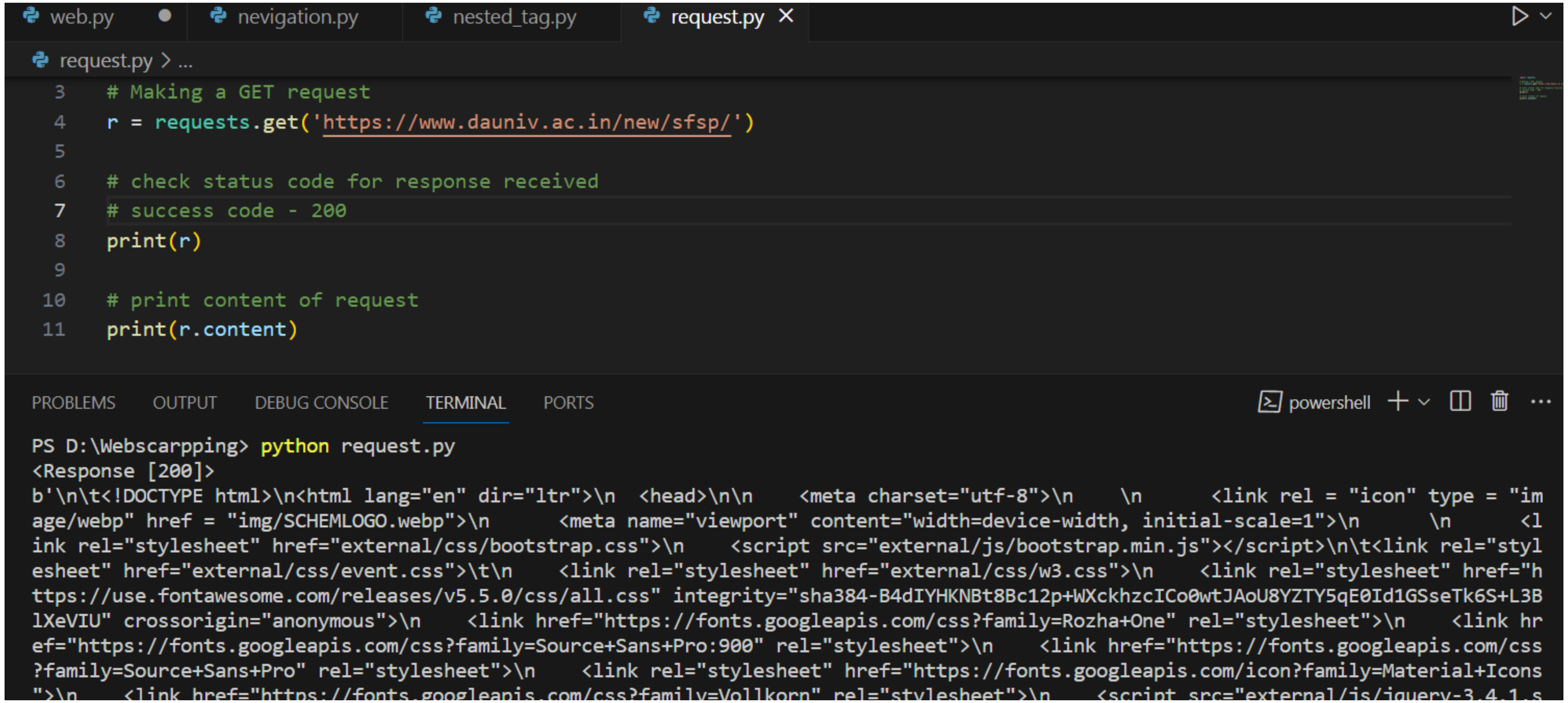
Requests library is used for making HTTP requests to a specific URL and returns the response. Python requests provide inbuilt functionalities for managing both the request and response.

Installation

```
pip install requests
```

Python requests module has several built-in methods to make HTTP requests to specified URI using GET, POST, PUT, PATCH, or HEAD requests. A HTTP request is meant to either retrieve data from a specified URI or to push data to a server. It works as a request-response protocol between a client and a server. Here we will be using the GET request.

GET method is used to retrieve information from the given server using a given URI. The GET method sends the encoded user information appended to the page request.

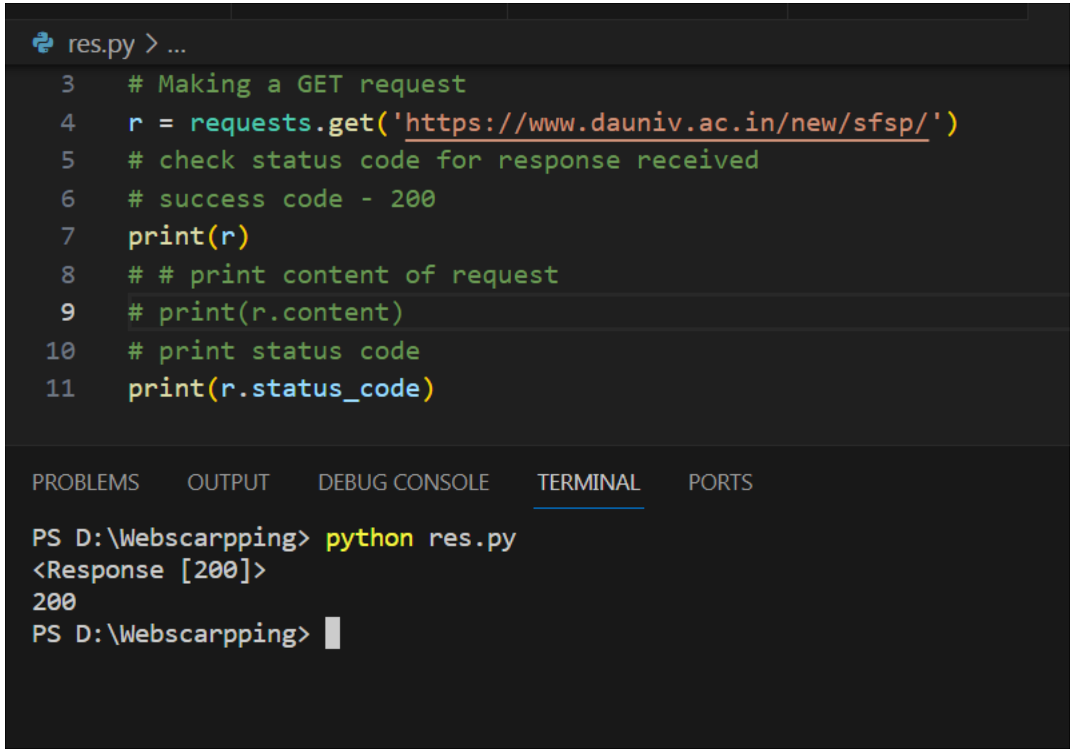


```python
# Making a GET request
r = requests.get('https://www.dauniv.ac.in/new/sfsp/')

# check status code for response received
# success code - 200
print(r)

# print content of request
print(r.content)
```

```
PS D:\Webscarpping> python request.py
<Response [200]>
b'\n\t<!DOCTYPE html>\n<html lang="en" dir="ltr">\n  <head>\n\n    <meta charset="utf-8">\n    \n      <link rel = "icon" type = "im
age/webp" href = "img/SCHEMLOGO.webp">\n      <meta name="viewport" content="width=device-width, initial-scale=1">\n      \n      <l
ink rel="stylesheet" href="external/css/bootstrap.css">\n    <script src="external/js/bootstrap.min.js"></script>\n\t<link rel="styl
esheet" href="external/css/event.css">\t\n    <link rel="stylesheet" href="external/css/w3.css">\n    <link rel="stylesheet" href="h
ttps://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3B
lXeVIU" crossorigin="anonymous">\n    <link href="https://fonts.googleapis.com/css?family=Rozha+One" rel="stylesheet">\n    <link hr
ef="https://fonts.googleapis.com/css?family=Source+Sans+Pro:900" rel="stylesheet">\n    <link href="https://fonts.googleapis.com/css
?family=Source+Sans+Pro" rel="stylesheet">\n    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons
">\n    <link href="https://fonts.googleapis.com/css?family=Vollkorn" rel="stylesheet">\n    <script src="external/js/jquery-3.4.1.s
```

Response object

When one makes a request to a URI, it returns a response. This Response object in terms of python is returned by requests.method(), method being – get, post, put, etc. Response is a powerful object with lots of functions and attributes that assist in normalizing data or creating ideal portions of code. For example, response.status_code returns the status code from the headers itself, and one can check if the request was processed successfully or not.



```python
# Making a GET request
r = requests.get('https://www.dauniv.ac.in/new/sfsp/')
# check status code for response received
# success code - 200
print(r)
# # print content of request
# print(r.content)
# print status code
print(r.status_code)
```

```
PS D:\Webscarpping> python res.py
<Response [200]>
200
PS D:\Webscarpping>
```

BeautifulSoup Library

Installation

```
pip install beautifulsoup4
```

Features of Beautiful Soup

Beautiful Soup is a Python library developed for quick reversal projects like screen-scraping. Three features make it powerful:

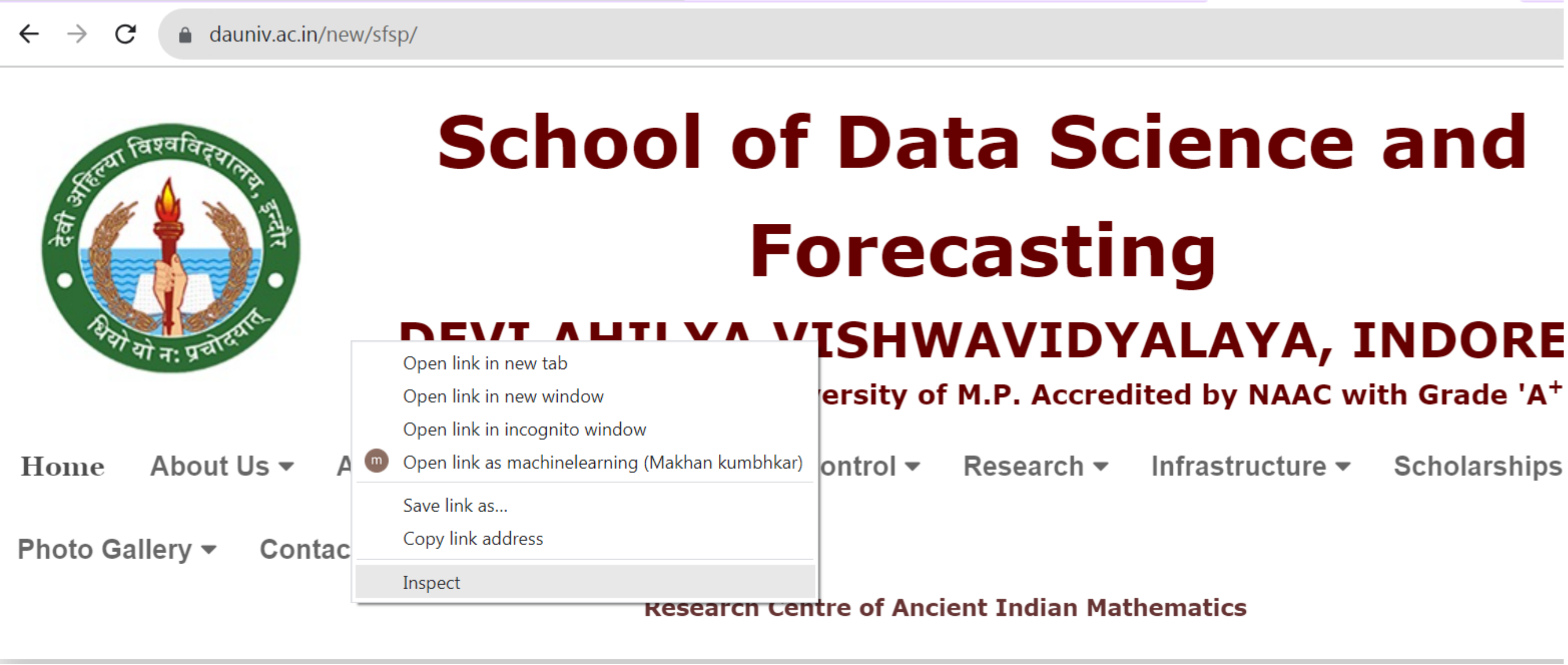
1. Beautiful Soup provides a few simple methods and Pythonic phrases for guiding, searching, and changing a parse tree: a toolkit for studying a document and removing what you need. It doesn’t take much code to document an application.

2. Beautiful Soup automatically converts incoming records to Unicode and outgoing forms to UTF-8. You don’t have to think about encodings unless the document doesn’t define an encoding, and Beautiful Soup can’t catch one. Then you just have to choose the original encoding.

3. Beautiful Soup sits on top of famous Python parsers like LXML and HTML, allowing you to try different parsing strategies or trade speed for flexibility.

Inspecting Website

Before getting out any information from the HTML of the page, we must understand the structure of the page. This is needed to be done in order to select the desired data from the entire page. We can do this by right-clicking on the page we want to scrape and select inspect element.

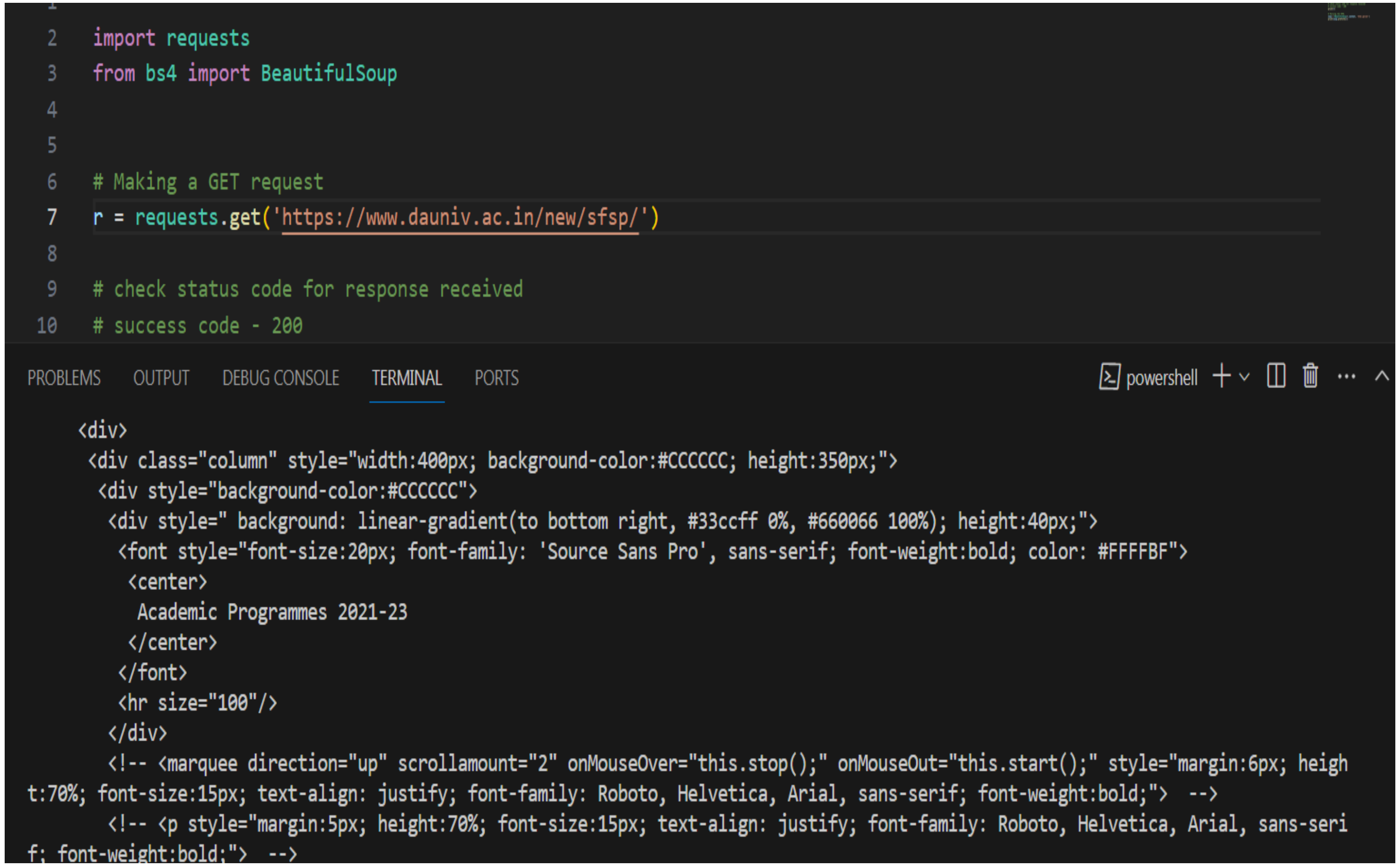


dauniv.ac.in/new/sfsp/
School of Data Science and Forecasting
DEVI AHILYA VISHWAVIDYALAYA, INDORE
ersity of M.P. Accredited by NAAC with Grade 'A+
Home
About Us
A
ontrol
Research
Infrastructure
Scholarships
Photo Gallery
Contac
Research Centre of Ancient Indian Mathematics
Open link in new tab
Open link in new window
Open link in incognito window
Open link as machinelearning (Makhan kumbhkar)
Save link as...
Copy link address
Inspect

Parsing the HTML

After getting the HTML of the page let’s see how to parse this raw HTML code into some useful information. First of all, we will create a BeautifulSoup object by specifying the parser we want to use.

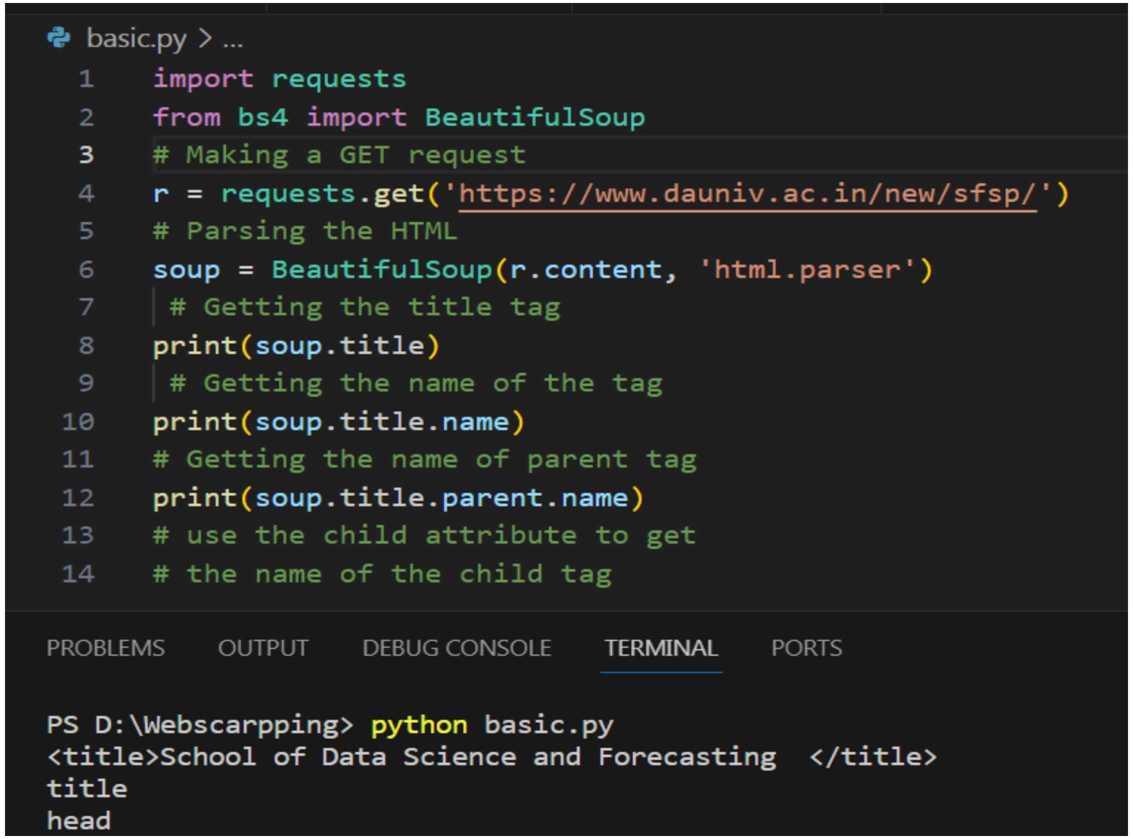
Note: BeautifulSoup library is built on top of the HTML parsing libraries like html5lib, lxml, html.parser, etc. So BeautifulSoup object and specify the parser library can be created at the same time.



```python
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.dauniv.ac.in/new/sfsp/')

# check status code for response received
# success code - 200
```

```
    <div>
     <div class="column" style="width:400px; background-color:#CCCCCC; height:350px;">
      <div style="background-color:#CCCCCC">
       <div style=" background: linear-gradient(to bottom right, #33ccff 0%, #660066 100%); height:40px;">
        <font style="font-size:20px; font-family: 'Source Sans Pro', sans-serif; font-weight:bold; color: #FFFFBF">
         <center>
          Academic Programmes 2021-23
         </center>
        </font>
        <hr size="100"/>
       </div>
       <!-- <marquee direction="up" scrollamount="2" onMouseOver="this.stop();" onMouseOut="this.start();" style="margin:6px; heigh
t:70%; font-size:15px; text-align: justify; font-family: Roboto, Helvetica, Arial, sans-serif; font-weight:bold;"> -->
       <!-- <p style="margin:5px; height:70%; font-size:15px; text-align: justify; font-family: Roboto, Helvetica, Arial, sans-seri
f; font-weight:bold;"> -->
```

Extract the title of the page



```python
import requests
from bs4 import BeautifulSoup
# Making a GET request
r = requests.get('https://www.dauniv.ac.in/new/sfsp/')
# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')
 # Getting the title tag
print(soup.title)
 # Getting the name of the tag
print(soup.title.name)
# Getting the name of parent tag
print(soup.title.parent.name)
# use the child attribute to get
# the name of the child tag
```

```
PS D:\Webscarpping> python basic.py
<title>School of Data Science and Forecasting  </title>
title
head
```

Finding Elements

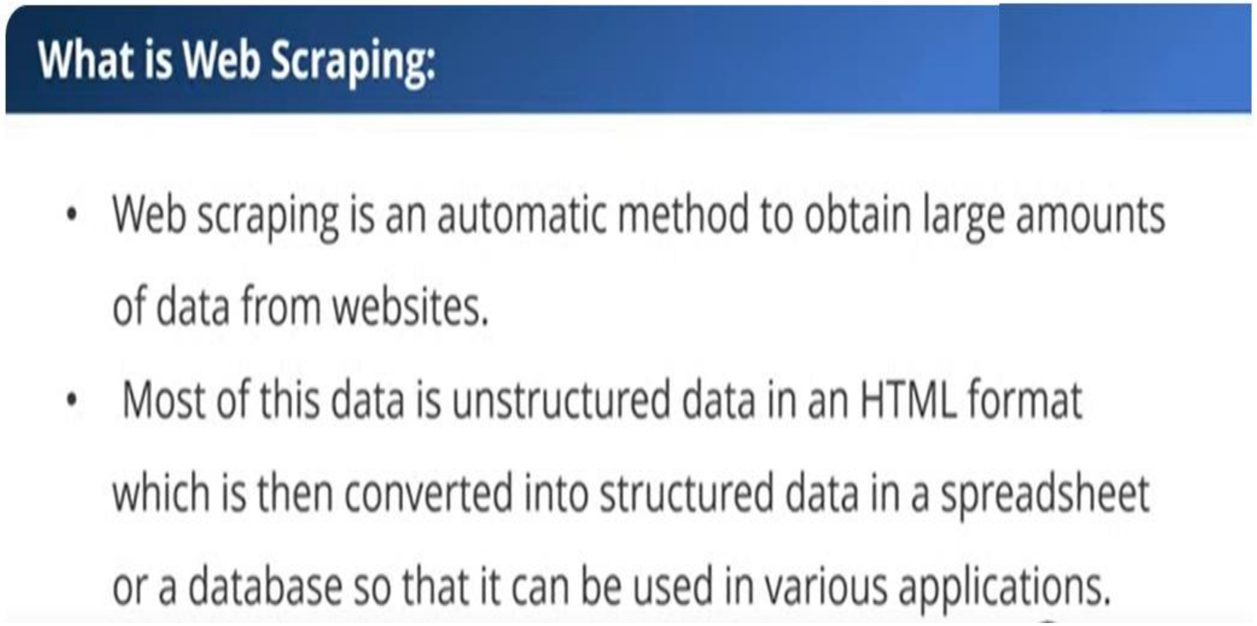
Now, we would like to extract some useful data from the HTML content. The soup object contains all the data in the nested structure which could be programmatically extracted. The website we want to scrape contains a lot of text so now let’s scrape all those content. First, let’s inspect the webpage we want to scrape.

Web Scrapping

Web Scrapping

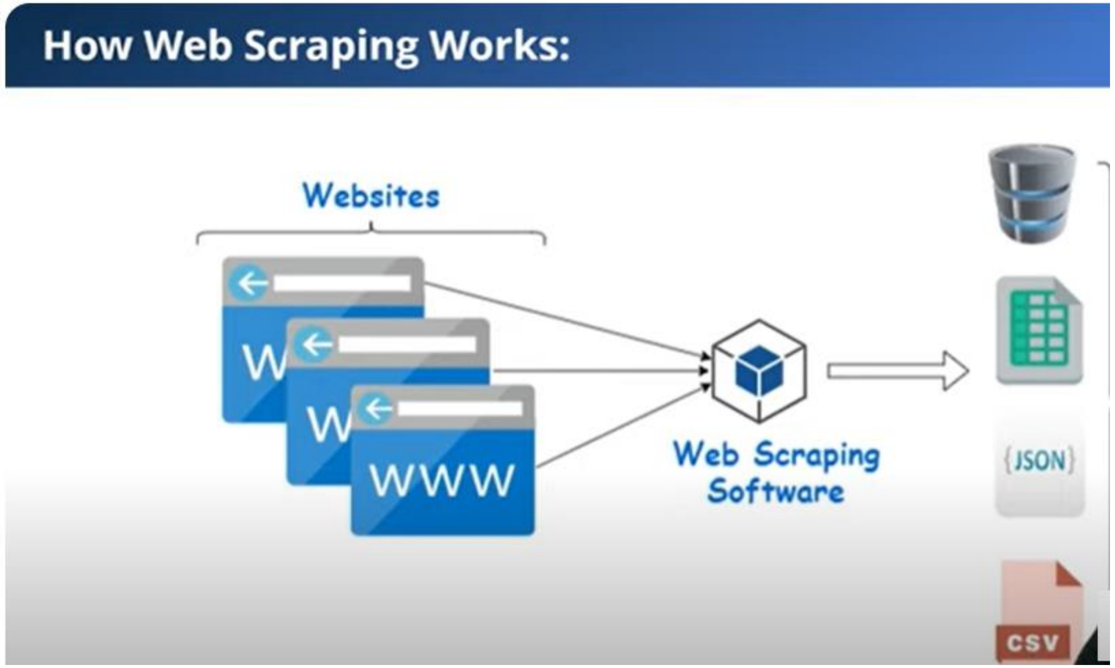
Web Scrapping

Web Scrapping

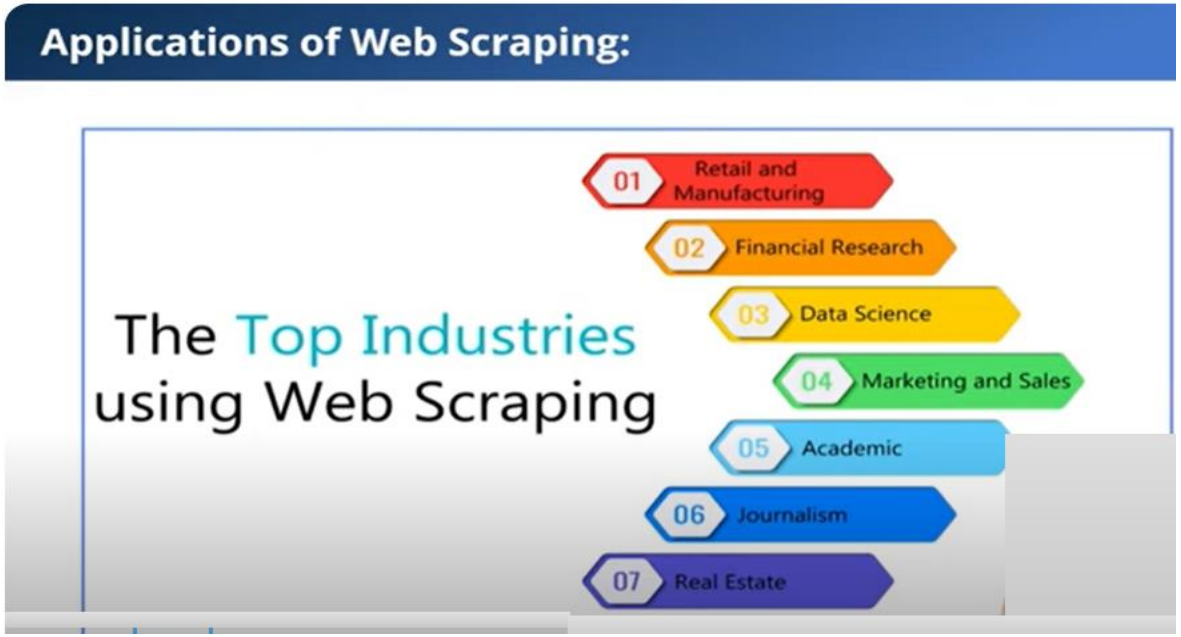


What is Web Scraping:

- Web scraping is an automatic method to obtain large amounts of data from websites.
- Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.



How Web Scraping Works:

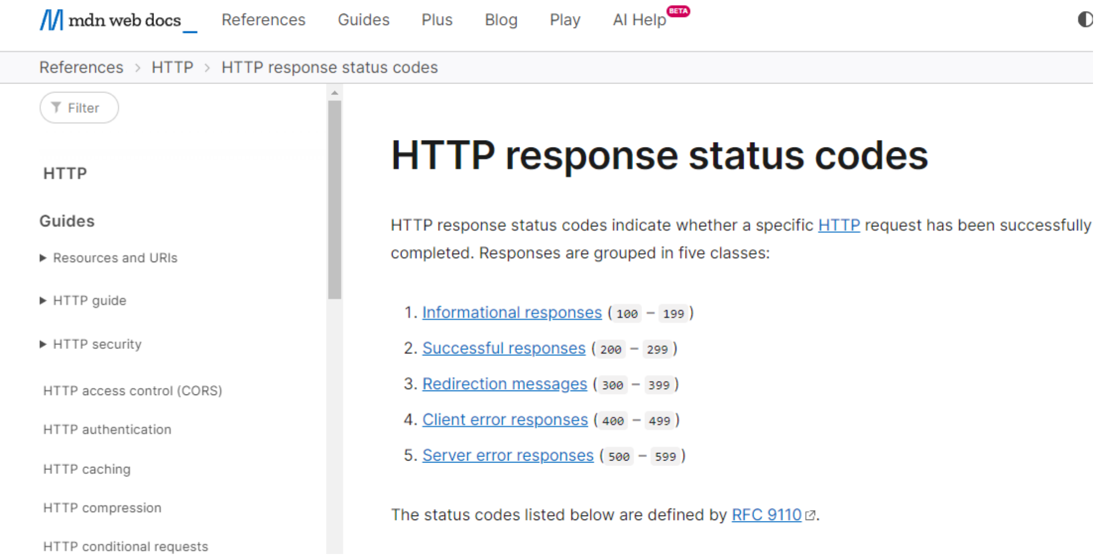


Applications of Web Scraping:

The Top Industries using Web Scraping

- 01 Retail and Manufacturing
- 02 Financial Research
- 03 Data Science
- 04 Marketing and Sales
- 05 Academic
- 06 Journalism
- 07 Real Estate

Web Scrapping



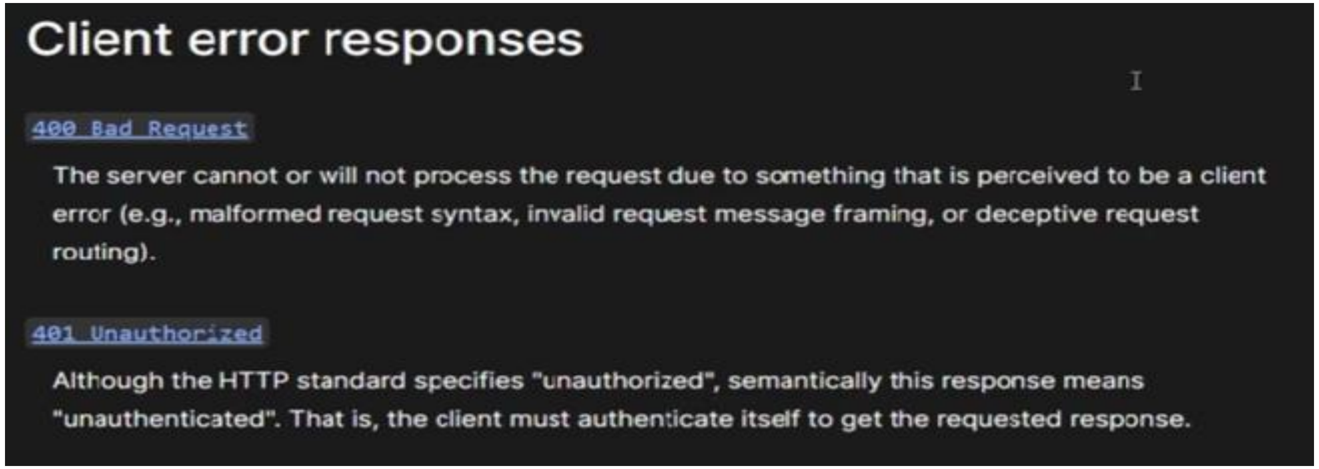
HTTP response status codes

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100 – 199)
2. Successful responses (200 – 299)
3. Redirection messages (300 – 399)
4. Client error responses (400 – 499)
5. Server error responses (500 – 599)

The status codes listed below are defined by RFC 9110.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status



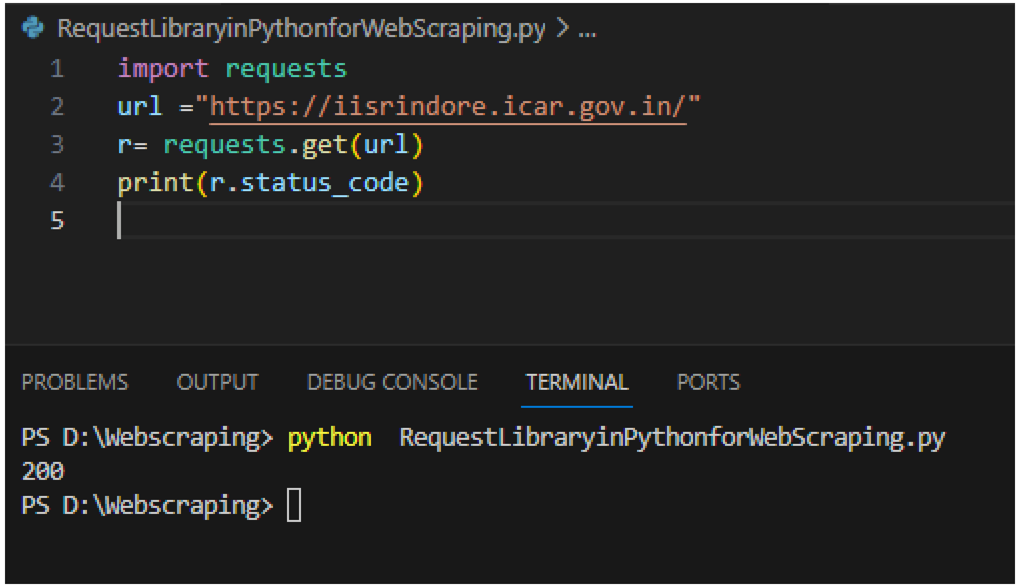
Client error responses

400 Bad Request

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

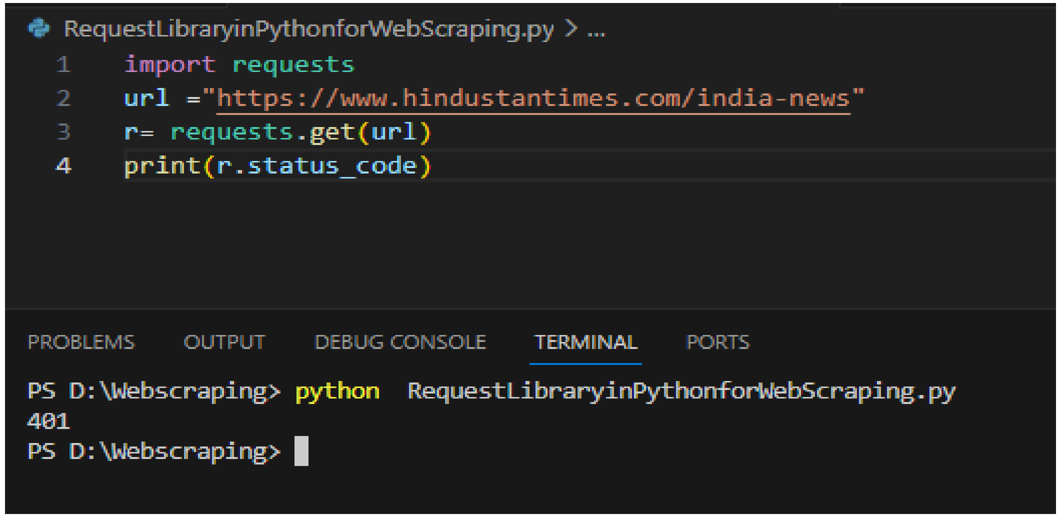
401 Unauthorized

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, the client must authenticate itself to get the requested response.



```python
import requests
url ="https://iisrindore.icar.gov.in/"
r= requests.get(url)
print(r.status_code)
```

```
PS D:\Webscraping> python  RequestLibraryinPythonforWebScraping.py
200
PS D:\Webscraping>
```

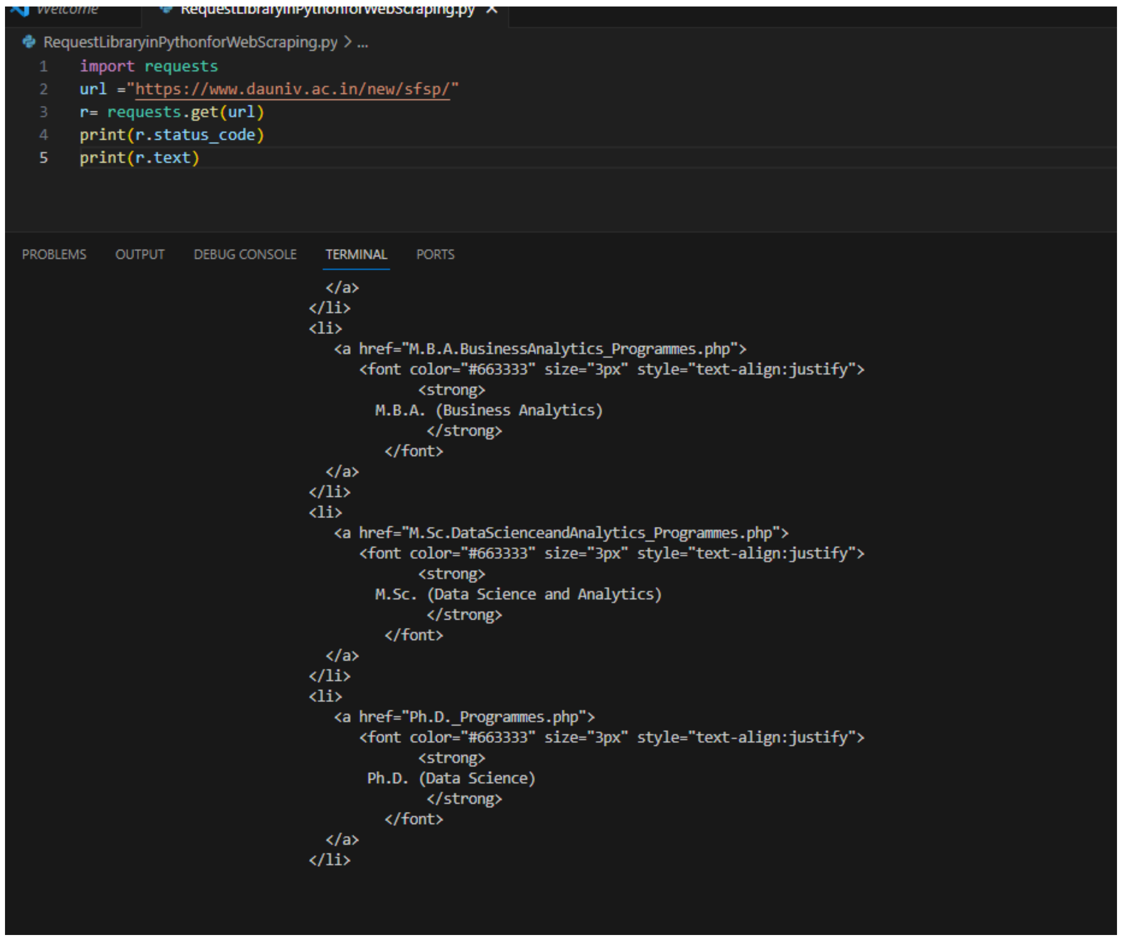


```python
import requests
url ="https://www.hindustantimes.com/india-news"
r= requests.get(url)
print(r.status_code)
```

```
PS D:\Webscraping> python  RequestLibraryinPythonforWebScraping.py
401
PS D:\Webscraping>
```

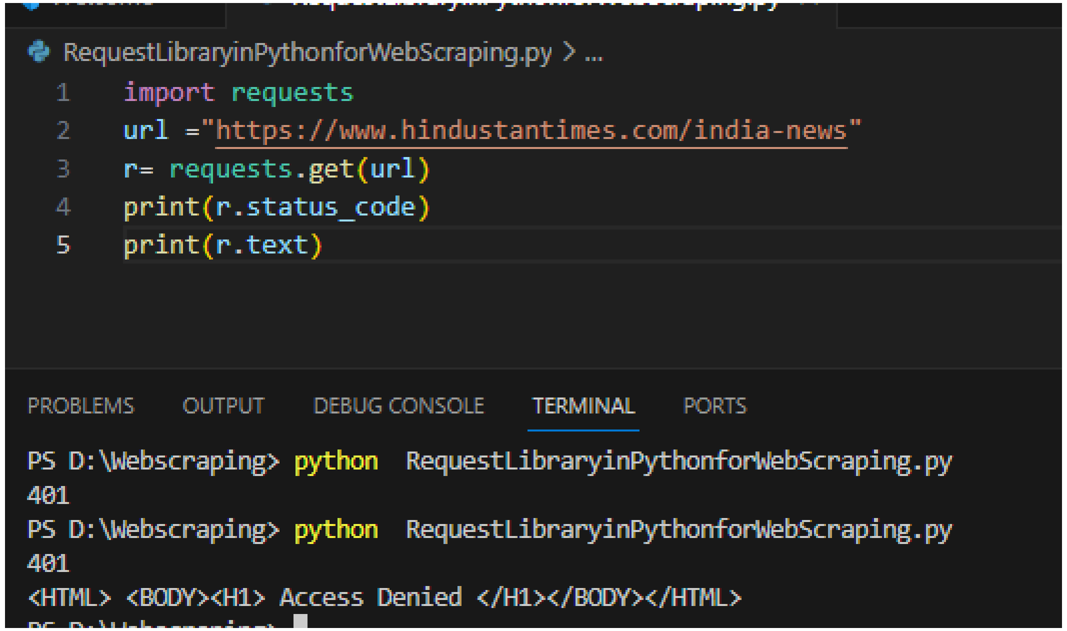
Print HTML

Example
Make a request to a web page, and print the response text:


Welcome
RequestLibraryinPythonforWebScraping.py
RequestLibraryinPythonforWebScraping.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

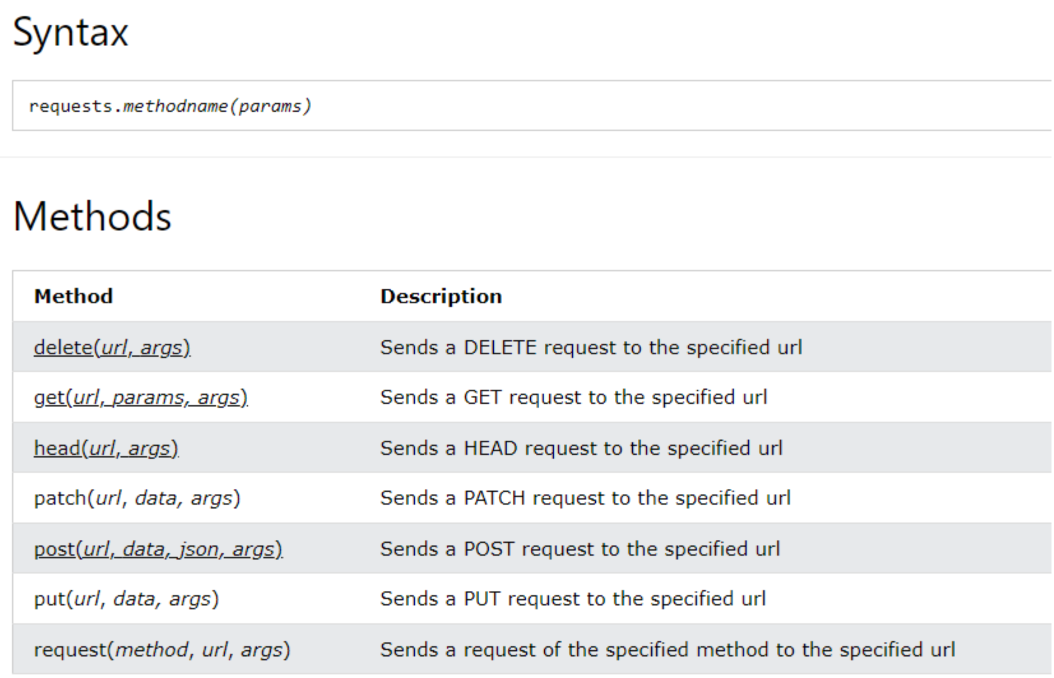
```python
import requests
url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
print(r.status_code)
print(r.text)
```

```
  </a>
</li>
<li>
   <a href="M.B.A.BusinessAnalytics_Programmes.php">
      <font color="#663333" size="3px" style="text-align:justify">
             <strong>
        M.B.A. (Business Analytics)
              </strong>
         </font>
  </a>
</li>
<li>
   <a href="M.Sc.DataScienceandAnalytics_Programmes.php">
      <font color="#663333" size="3px" style="text-align:justify">
             <strong>
        M.Sc. (Data Science and Analytics)
              </strong>
         </font>
  </a>
</li>
<li>
   <a href="Ph.D._Programmes.php">
      <font color="#663333" size="3px" style="text-align:justify">
             <strong>
       Ph.D. (Data Science)
              </strong>
         </font>
  </a>
</li>
```



```python
import requests
url ="https://www.hindustantimes.com/india-news"
r= requests.get(url)
print(r.status_code)
print(r.text)
```

```
PS D:\Webscraping> python  RequestLibraryinPythonforWebScraping.py
401
PS D:\Webscraping> python  RequestLibraryinPythonforWebScraping.py
401
<HTML> <BODY><H1> Access Denied </H1></BODY></HTML>
PS D:\Webscraping>
```



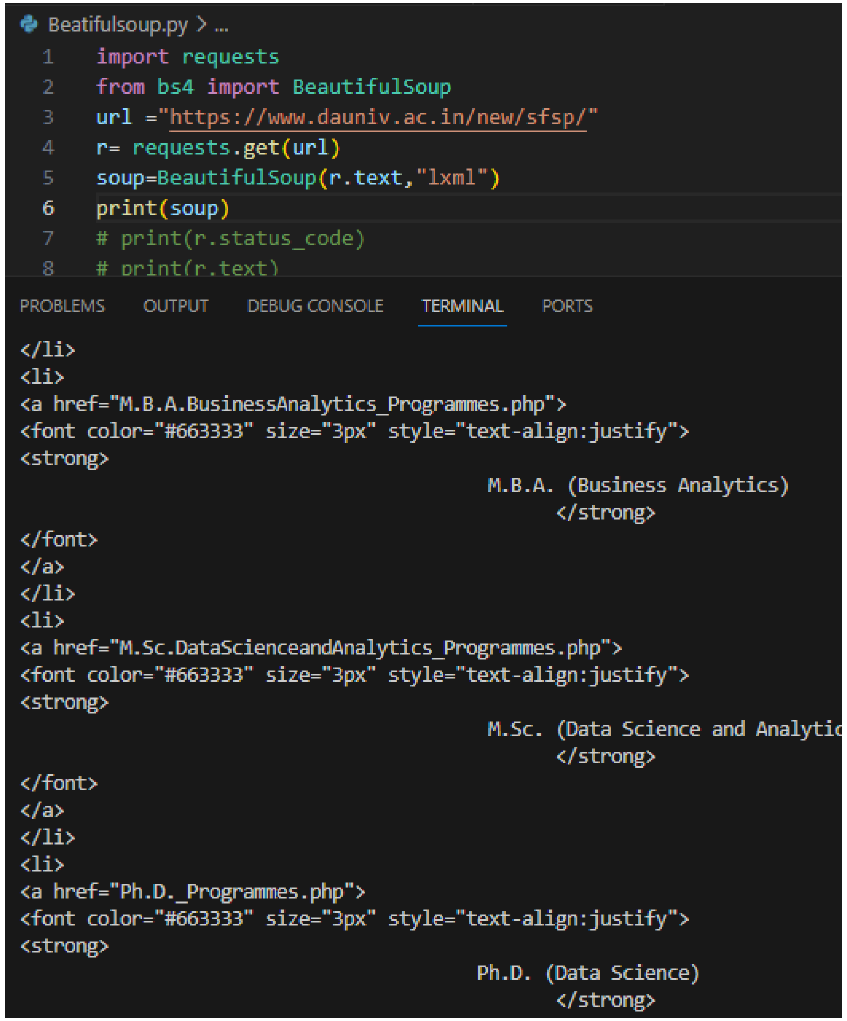
Syntax

```
requests.methodname(params)
```

Methods

Method	Description
delete(*url*, *args*)	Sends a DELETE request to the specified url
get(*url*, *params*, *args*)	Sends a GET request to the specified url
head(*url*, *args*)	Sends a HEAD request to the specified url
patch(*url*, *data*, *args*)	Sends a PATCH request to the specified url
post(*url*, *data*, *json*, *args*)	Sends a POST request to the specified url
put(*url*, *data*, *args*)	Sends a PUT request to the specified url
request(*method*, *url*, *args*)	Sends a request of the specified method to the specified url

How obtain HTML using BeautifulSoup



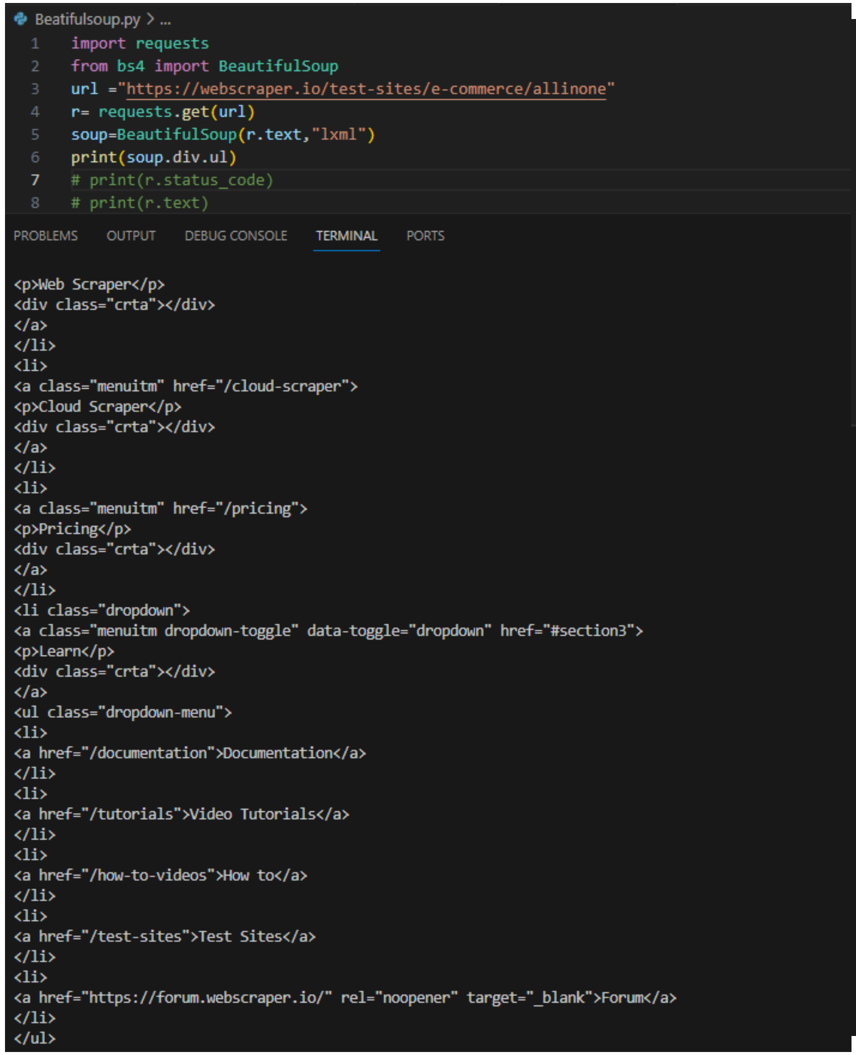
```python
import requests
from bs4 import BeautifulSoup
url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
print(soup)
# print(r.status_code)
# print(r.text)
```

```
</li>
<li>
<a href="M.B.A.BusinessAnalytics_Programmes.php">
<font color="#663333" size="3px" style="text-align:justify">
<strong>
                                          M.B.A. (Business Analytics)
                                                </strong>
</font>
</a>
</li>
<li>
<a href="M.Sc.DataScienceandAnalytics_Programmes.php">
<font color="#663333" size="3px" style="text-align:justify">
<strong>
                                          M.Sc. (Data Science and Analytic
                                                </strong>
</font>
</a>
</li>
<li>
<a href="Ph.D._Programmes.php">
<font color="#663333" size="3px" style="text-align:justify">
<strong>
                                         Ph.D. (Data Science)
                                                </strong>
```

Different tested template for webscraping

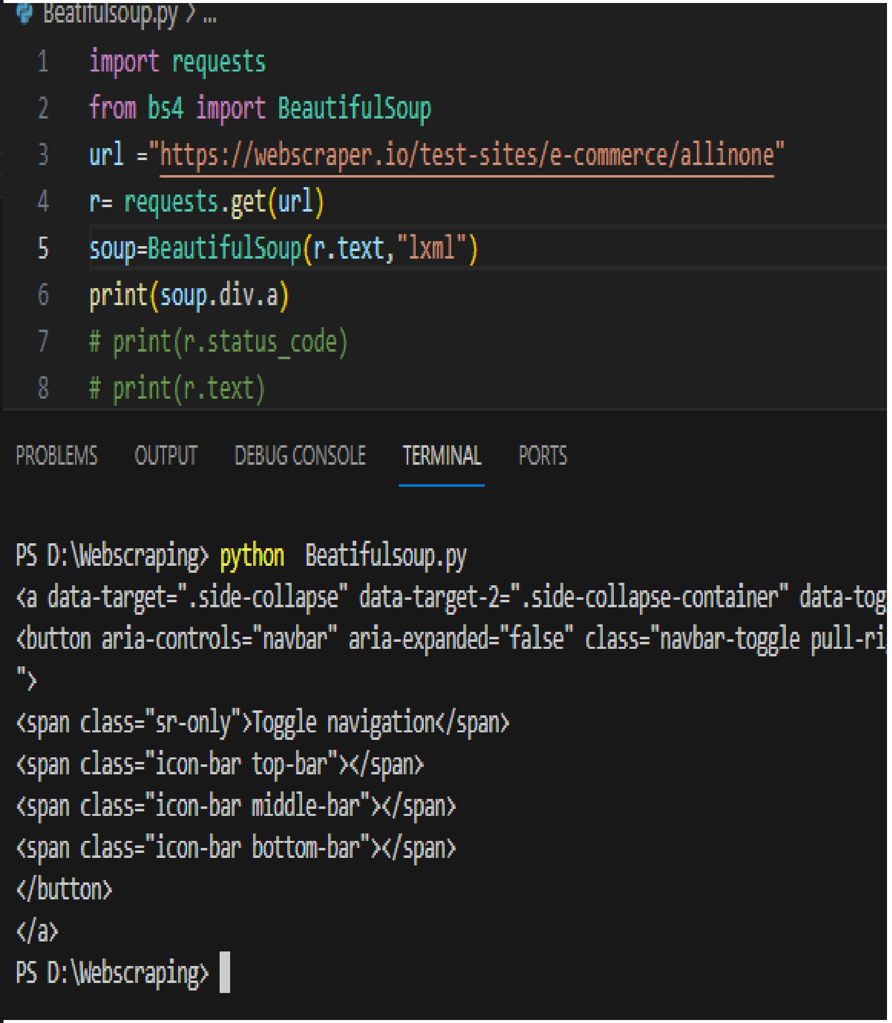
https://webscraper.io/test-sites/e-commerce/allinone/computers

How to see attributes



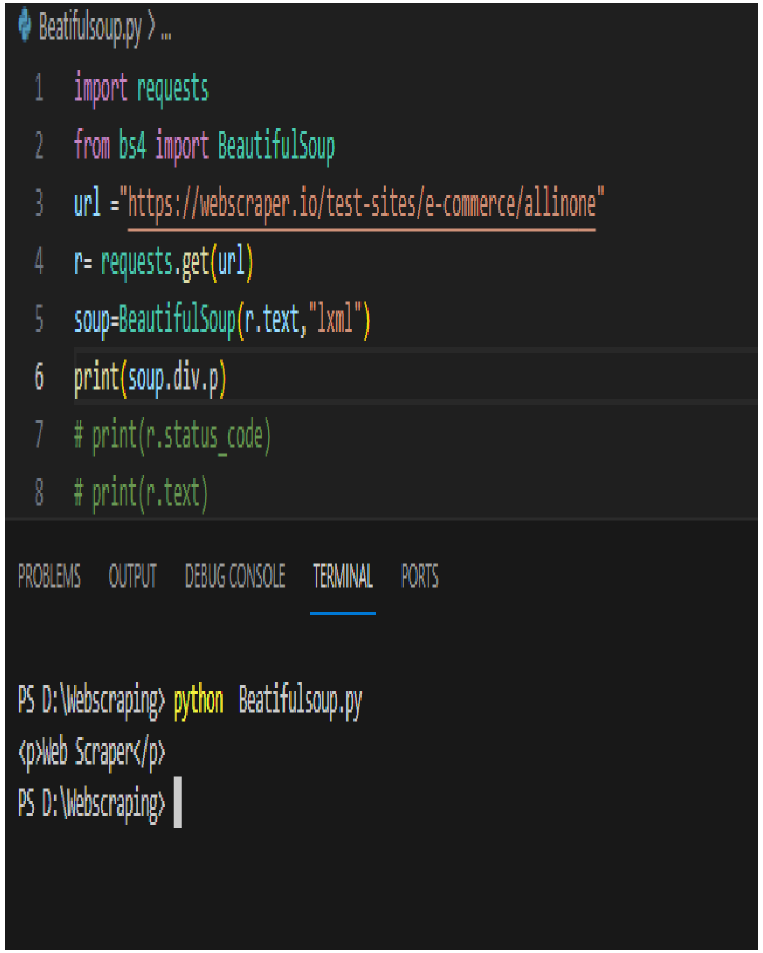
```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
print(soup.div.ul)
# print(r.status_code)
# print(r.text)
```

```
<p>Web Scraper</p>
<div class="crta"></div>
</a>
</li>
<li>
<a class="menuitm" href="/cloud-scraper">
<p>Cloud Scraper</p>
<div class="crta"></div>
</a>
</li>
<li>
<a class="menuitm" href="/pricing">
<p>Pricing</p>
<div class="crta"></div>
</a>
</li>
<li class="dropdown">
<a class="menuitm dropdown-toggle" data-toggle="dropdown" href="#section3">
<p>Learn</p>
<div class="crta"></div>
</a>
<ul class="dropdown-menu">
<li>
<a href="/documentation">Documentation</a>
</li>
<li>
<a href="/tutorials">Video Tutorials</a>
</li>
<li>
<a href="/how-to-videos">How to</a>
</li>
<li>
<a href="/test-sites">Test Sites</a>
</li>
<li>
<a href="https://forum.webscraper.io/" rel="noopener" target="_blank">Forum</a>
</li>
</ul>
```



```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
print(soup.div.a)
# print(r.status_code)
# print(r.text)
```

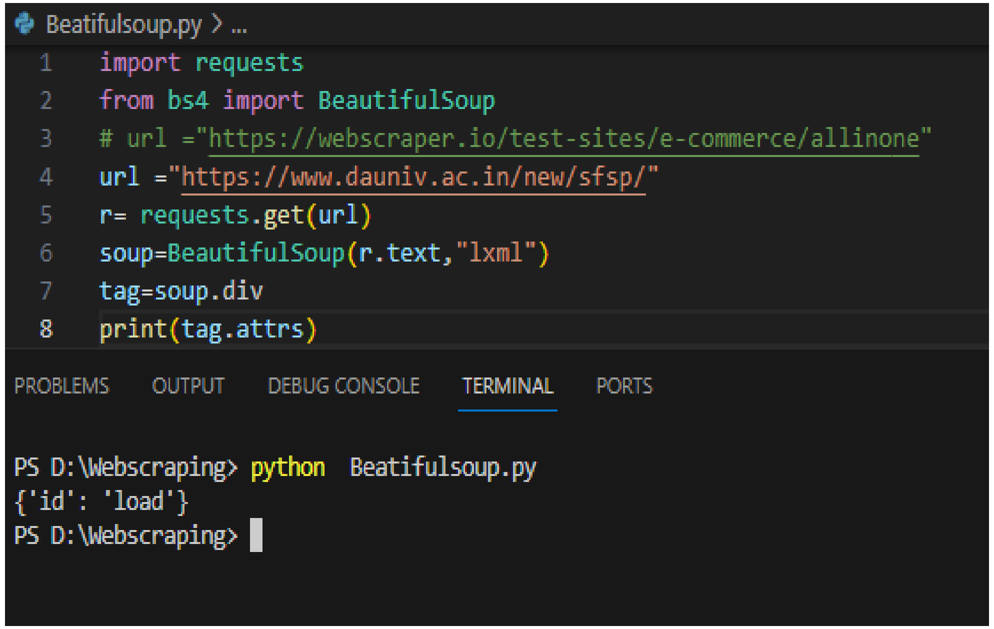
```
PS D:\Webscraping> python  Beatifulsoup.py
<a data-target=".side-collapse" data-target-2=".side-collapse-container" data-tog
<button aria-controls="navbar" aria-expanded="false" class="navbar-toggle pull-ri
">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar top-bar"></span>
<span class="icon-bar middle-bar"></span>
<span class="icon-bar bottom-bar"></span>
</button>
</a>
PS D:\Webscraping>
```



```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
print(soup.div.p)
# print(r.status_code)
# print(r.text)
```

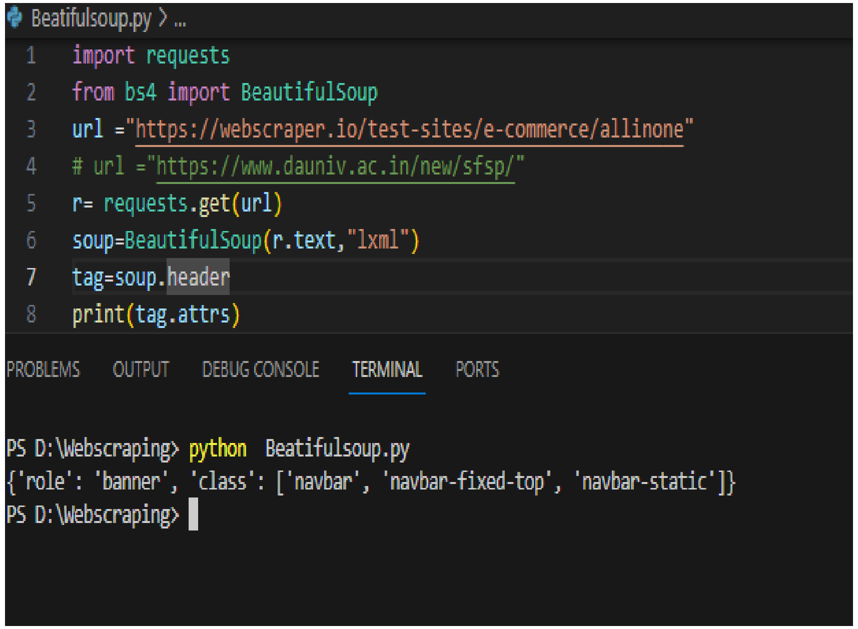
```
PS D:\Webscraping> python  Beatifulsoup.py
<p>Web Scraper</p>
PS D:\Webscraping>
```

how to get attributes inside tag



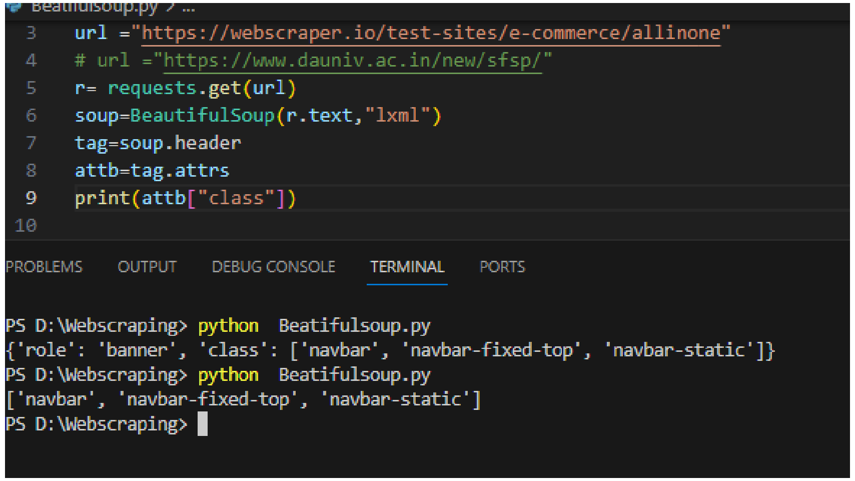
```python
import requests
from bs4 import BeautifulSoup
# url ="https://webscraper.io/test-sites/e-commerce/allinone"
url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
tag=soup.div
print(tag.attrs)
```

```
PS D:\Webscraping> python  Beatifulsoup.py
{'id': 'load'}
PS D:\Webscraping> 
```



```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone"
# url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
tag=soup.header
print(tag.attrs)
```

```
PS D:\Webscraping> python  Beatifulsoup.py
{'role': 'banner', 'class': ['navbar', 'navbar-fixed-top', 'navbar-static']}
PS D:\Webscraping> 
```

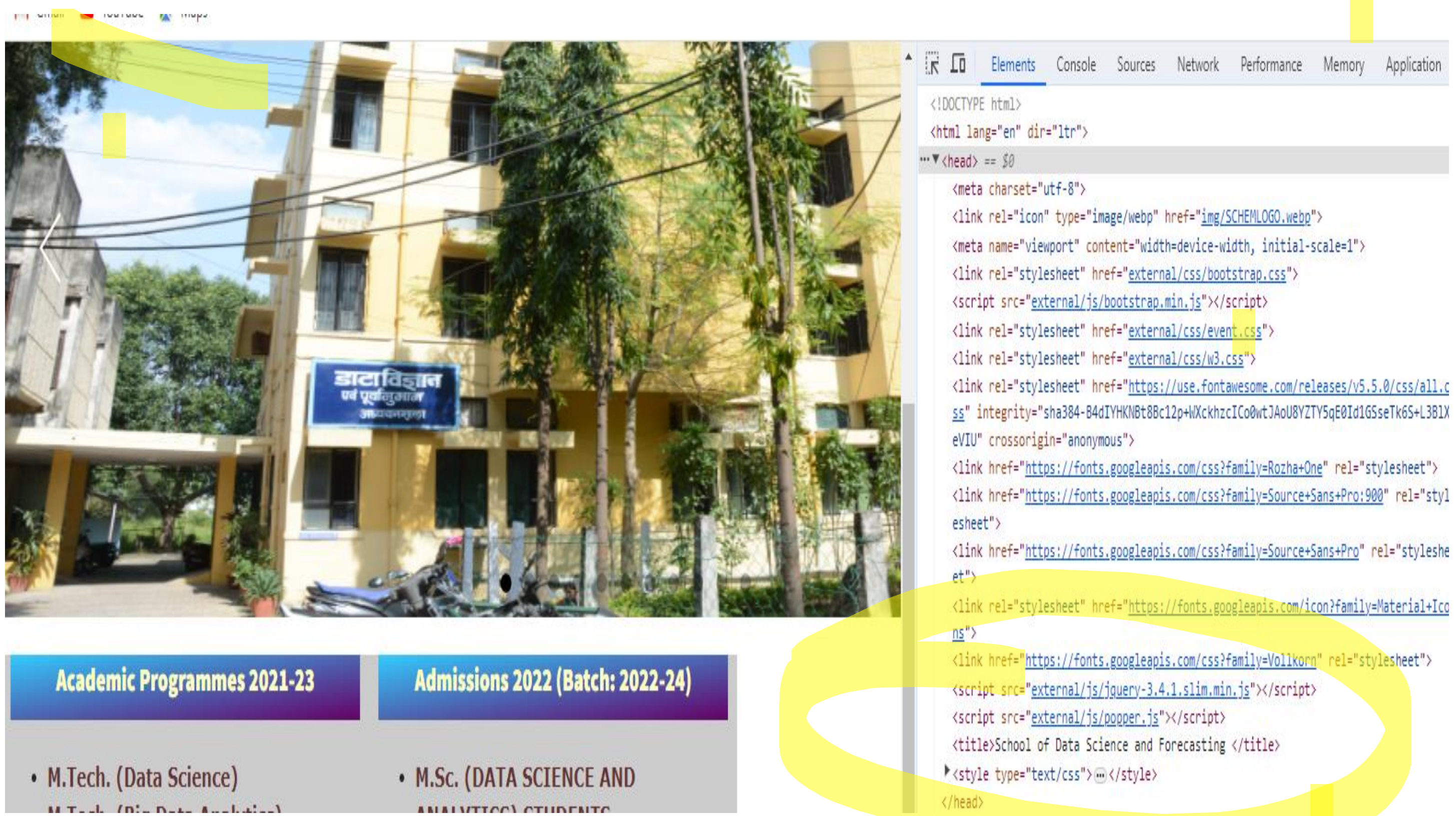


```python
url ="https://webscraper.io/test-sites/e-commerce/allinone"
# url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
tag=soup.header
attb=tag.attrs
print(attb["class"])

```

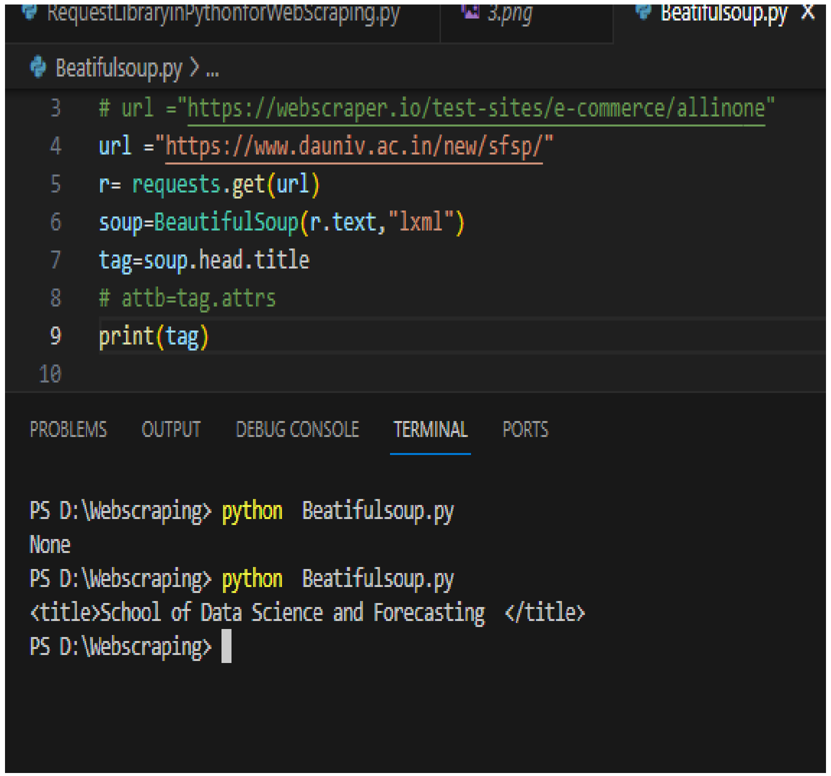
```
PS D:\Webscraping> python  Beatifulsoup.py
{'role': 'banner', 'class': ['navbar', 'navbar-fixed-top', 'navbar-static']}
PS D:\Webscraping> python  Beatifulsoup.py
['navbar', 'navbar-fixed-top', 'navbar-static']
PS D:\Webscraping> 
```

Navigation string in the HTML

A Navigable string object holds the text within an HTML or an XML tag.



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
··· ▼<head> == $0
    <meta charset="utf-8">
    <link rel="icon" type="image/webp" href="img/SCHEMLOGO.webp">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="external/css/bootstrap.css">
    <script src="external/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="external/css/event.css">
    <link rel="stylesheet" href="external/css/w3.css">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.c
    ss" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlX
    eVIU" crossorigin="anonymous">
    <link href="https://fonts.googleapis.com/css?family=Rozha+One" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:900" rel="styl
    esheet">
    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro" rel="styleshe
    et">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Ico
    ns">
    <link href="https://fonts.googleapis.com/css?family=Vollkorn" rel="stylesheet">
    <script src="external/js/jquery-3.4.1.slim.min.js"></script>
    <script src="external/js/popper.js"></script>
    <title>School of Data Science and Forecasting </title>
   ▸<style type="text/css">⋯</style>
  </head>
```



```python
# url ="https://webscraper.io/test-sites/e-commerce/allinone"
url ="https://www.dauniv.ac.in/new/sfsp/"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
tag=soup.head.title
# attb=tag.attrs
print(tag)
```

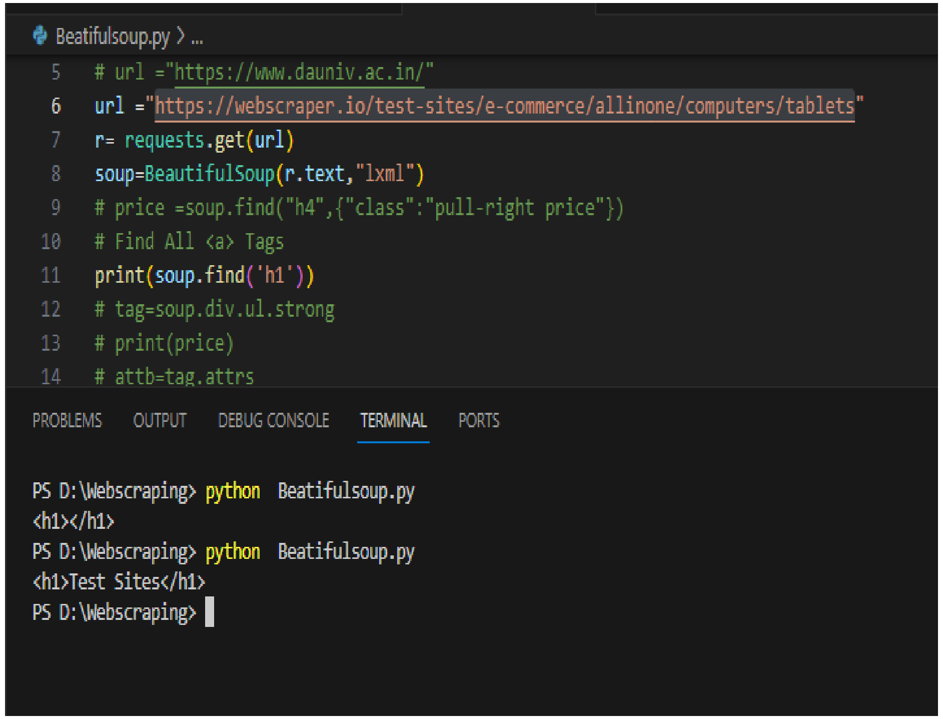
```
PS D:\Webscraping> python  Beatifulsoup.py
None
PS D:\Webscraping> python  Beatifulsoup.py
<title>School of Data Science and Forecasting  </title>
PS D:\Webscraping> ▌
```

BeautifulSoup .find() Method

We should use the .find() method when there is only one element that matches our query criteria, or just want the first element.

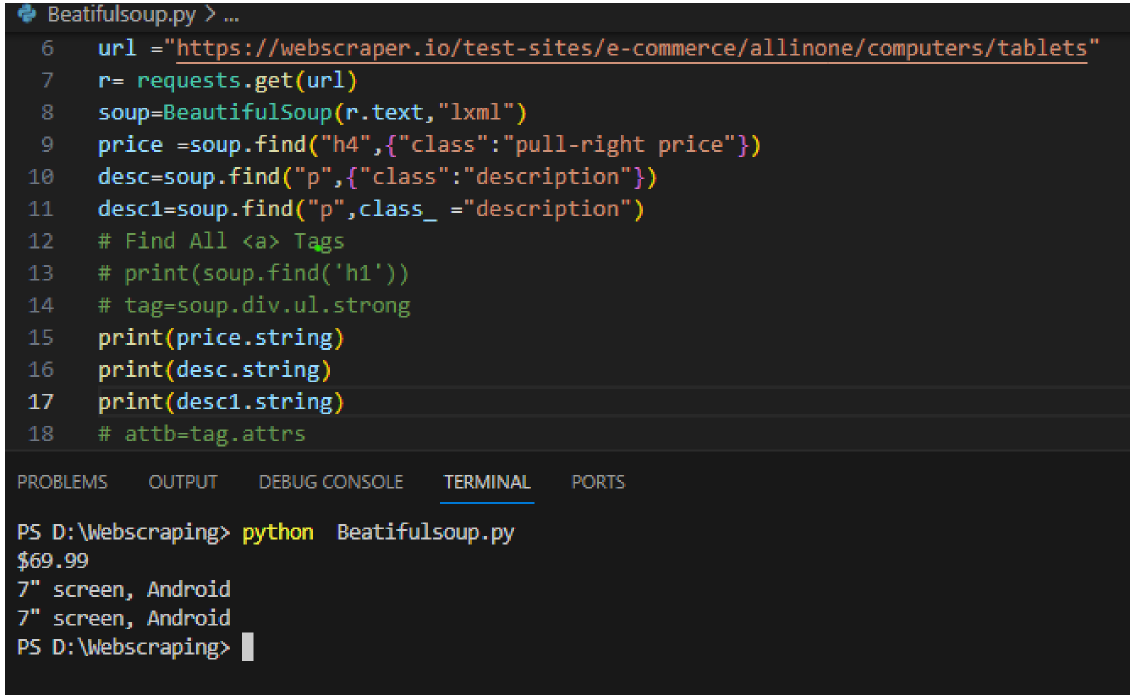
The .find() returns the first element that matches your query criteria.

https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets



```
# url ="https://www.dauniv.ac.in/"
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
# price =soup.find("h4",{"class":"pull-right price"})
# Find All <a> Tags
print(soup.find('h1'))
# tag=soup.div.ul.strong
# print(price)
# attb=tag.attrs
```

```
PS D:\Webscraping> python  Beatifulsoup.py
<h1></h1>
PS D:\Webscraping> python  Beatifulsoup.py
<h1>Test Sites</h1>
PS D:\Webscraping>
```



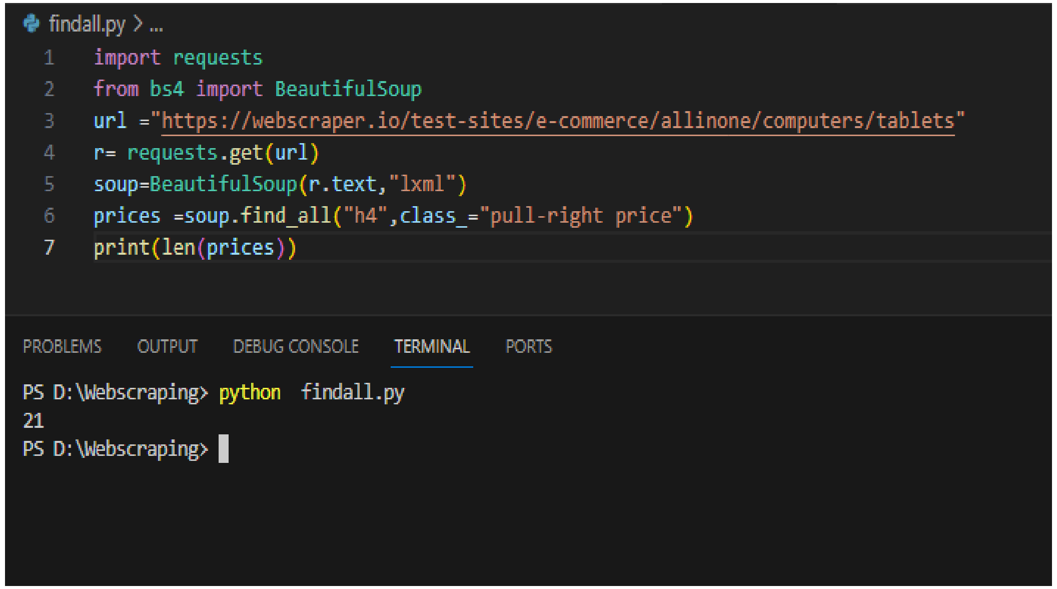
```
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
price =soup.find("h4",{"class":"pull-right price"})
desc=soup.find("p",{"class":"description"})
desc1=soup.find("p",class_ ="description")
# Find All <a> Tags
# print(soup.find('h1'))
# tag=soup.div.ul.strong
print(price.string)
print(desc.string)
print(desc1.string)
# attb=tag.attrs
```

```
PS D:\Webscraping> python  Beatifulsoup.py
$69.99
7" screen, Android
7" screen, Android
PS D:\Webscraping>
```

Find_all method with tag

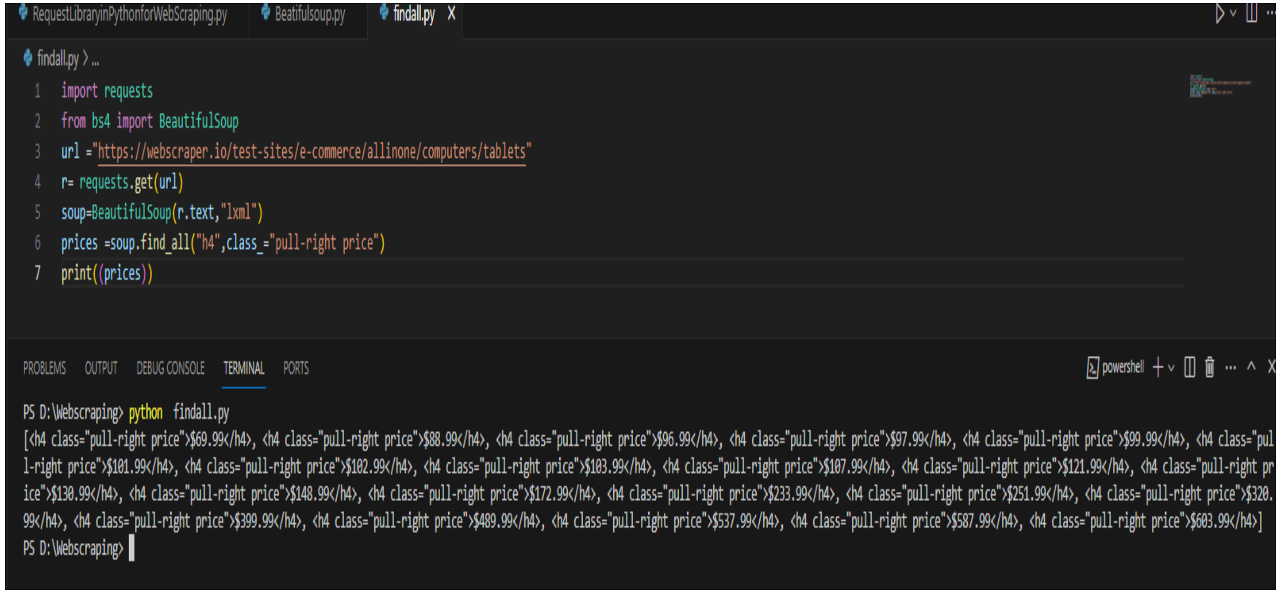
Beautiful Soup's find_all() method returns a list of all the tags or strings that match a particular criteria.

https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets



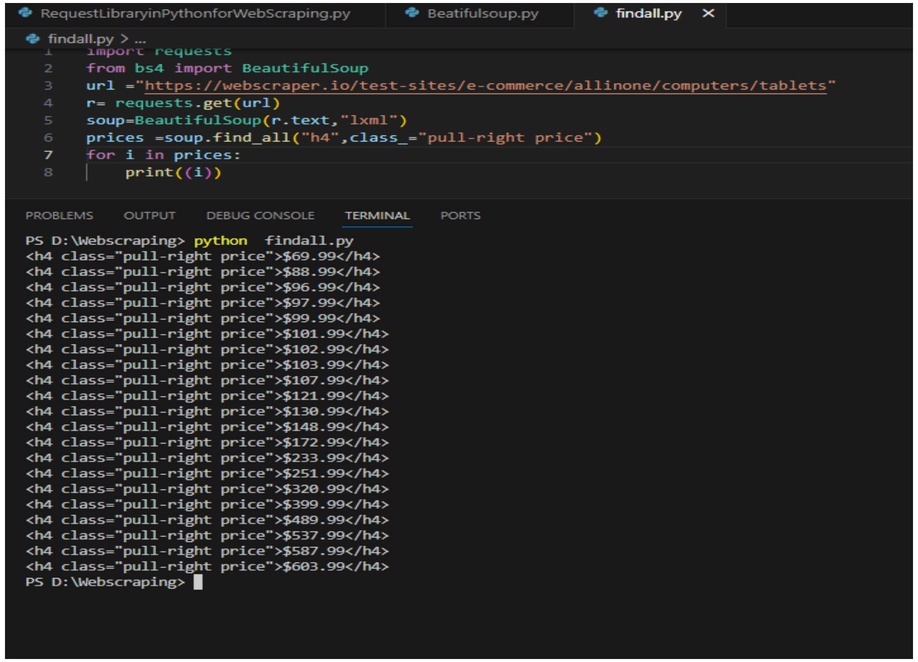
```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
prices =soup.find_all("h4",class_="pull-right price")
print(len(prices))
```

```
PS D:\Webscraping> python  findall.py
21
PS D:\Webscraping> 
```



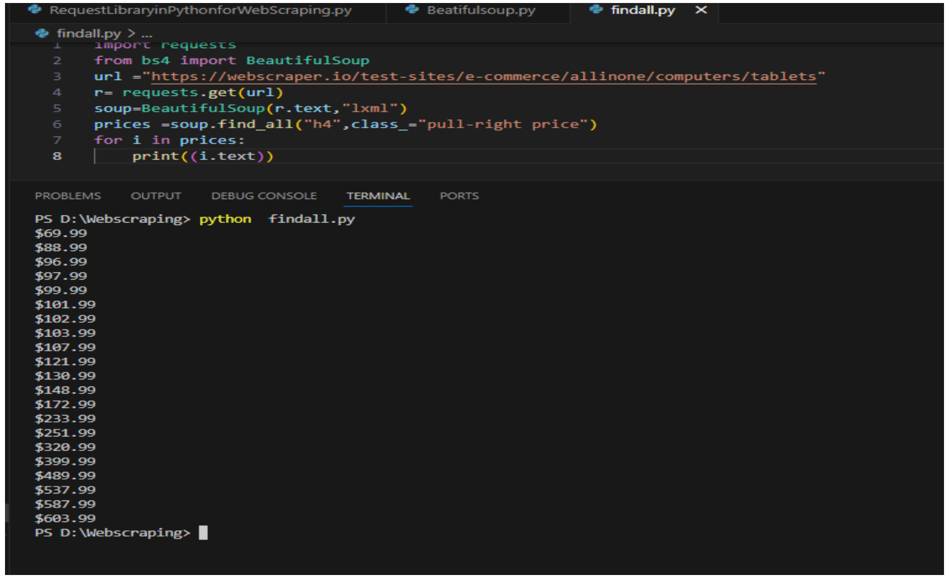
```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
prices =soup.find_all("h4",class_="pull-right price")
print((prices))
```

```
PS D:\Webscraping> python  findall.py
[<h4 class="pull-right price">$69.99</h4>, <h4 class="pull-right price">$88.99</h4>, <h4 class="pull-right price">$96.99</h4>, <h4 class="pull-right price">$97.99</h4>, <h4 class="pull-right price">$99.99</h4>, <h4 class="pul
l-right price">$101.99</h4>, <h4 class="pull-right price">$102.99</h4>, <h4 class="pull-right price">$103.99</h4>, <h4 class="pull-right price">$107.99</h4>, <h4 class="pull-right price">$121.99</h4>, <h4 class="pull-right pr
ice">$130.99</h4>, <h4 class="pull-right price">$148.99</h4>, <h4 class="pull-right price">$172.99</h4>, <h4 class="pull-right price">$233.99</h4>, <h4 class="pull-right price">$251.99</h4>, <h4 class="pull-right price">$320.
99</h4>, <h4 class="pull-right price">$399.99</h4>, <h4 class="pull-right price">$489.99</h4>, <h4 class="pull-right price">$537.99</h4>, <h4 class="pull-right price">$587.99</h4>, <h4 class="pull-right price">$603.99</h4>]
PS D:\Webscraping> 
```



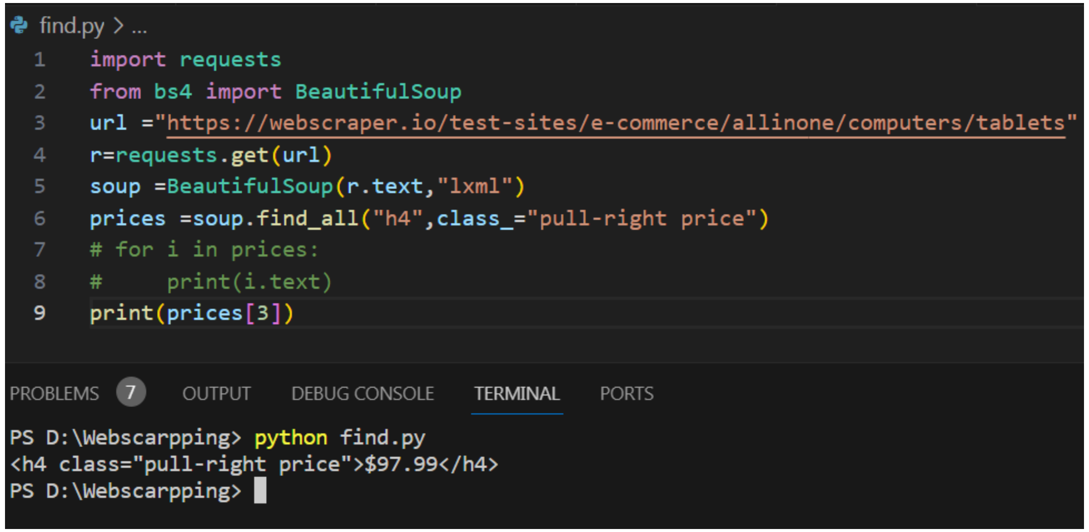
```
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
prices =soup.find_all("h4",class_="pull-right price")
for i in prices:
    print((i))
```

```
PS D:\Webscraping> python  findall.py
<h4 class="pull-right price">$69.99</h4>
<h4 class="pull-right price">$88.99</h4>
<h4 class="pull-right price">$96.99</h4>
<h4 class="pull-right price">$97.99</h4>
<h4 class="pull-right price">$99.99</h4>
<h4 class="pull-right price">$101.99</h4>
<h4 class="pull-right price">$102.99</h4>
<h4 class="pull-right price">$103.99</h4>
<h4 class="pull-right price">$107.99</h4>
<h4 class="pull-right price">$121.99</h4>
<h4 class="pull-right price">$130.99</h4>
<h4 class="pull-right price">$148.99</h4>
<h4 class="pull-right price">$172.99</h4>
<h4 class="pull-right price">$233.99</h4>
<h4 class="pull-right price">$251.99</h4>
<h4 class="pull-right price">$320.99</h4>
<h4 class="pull-right price">$399.99</h4>
<h4 class="pull-right price">$489.99</h4>
<h4 class="pull-right price">$537.99</h4>
<h4 class="pull-right price">$587.99</h4>
<h4 class="pull-right price">$603.99</h4>
PS D:\Webscraping>
```



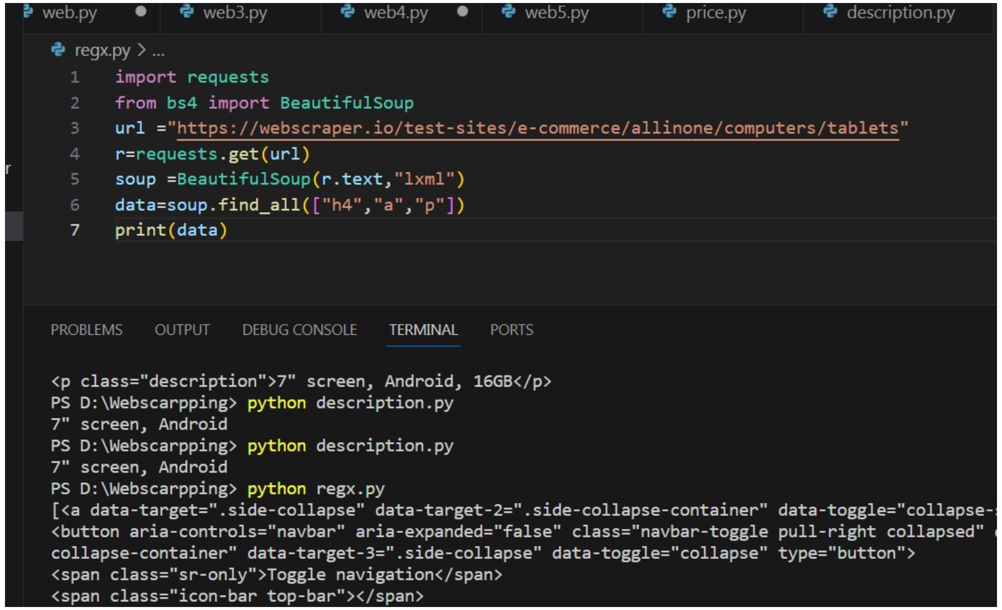
```
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r= requests.get(url)
soup=BeautifulSoup(r.text,"lxml")
prices =soup.find_all("h4",class_="pull-right price")
for i in prices:
    print((i.text))
```

```
PS D:\Webscraping> python  findall.py
$69.99
$88.99
$96.99
$97.99
$99.99
$101.99
$102.99
$103.99
$107.99
$121.99
$130.99
$148.99
$172.99
$233.99
$251.99
$320.99
$399.99
$489.99
$537.99
$587.99
$603.99
PS D:\Webscraping>
```



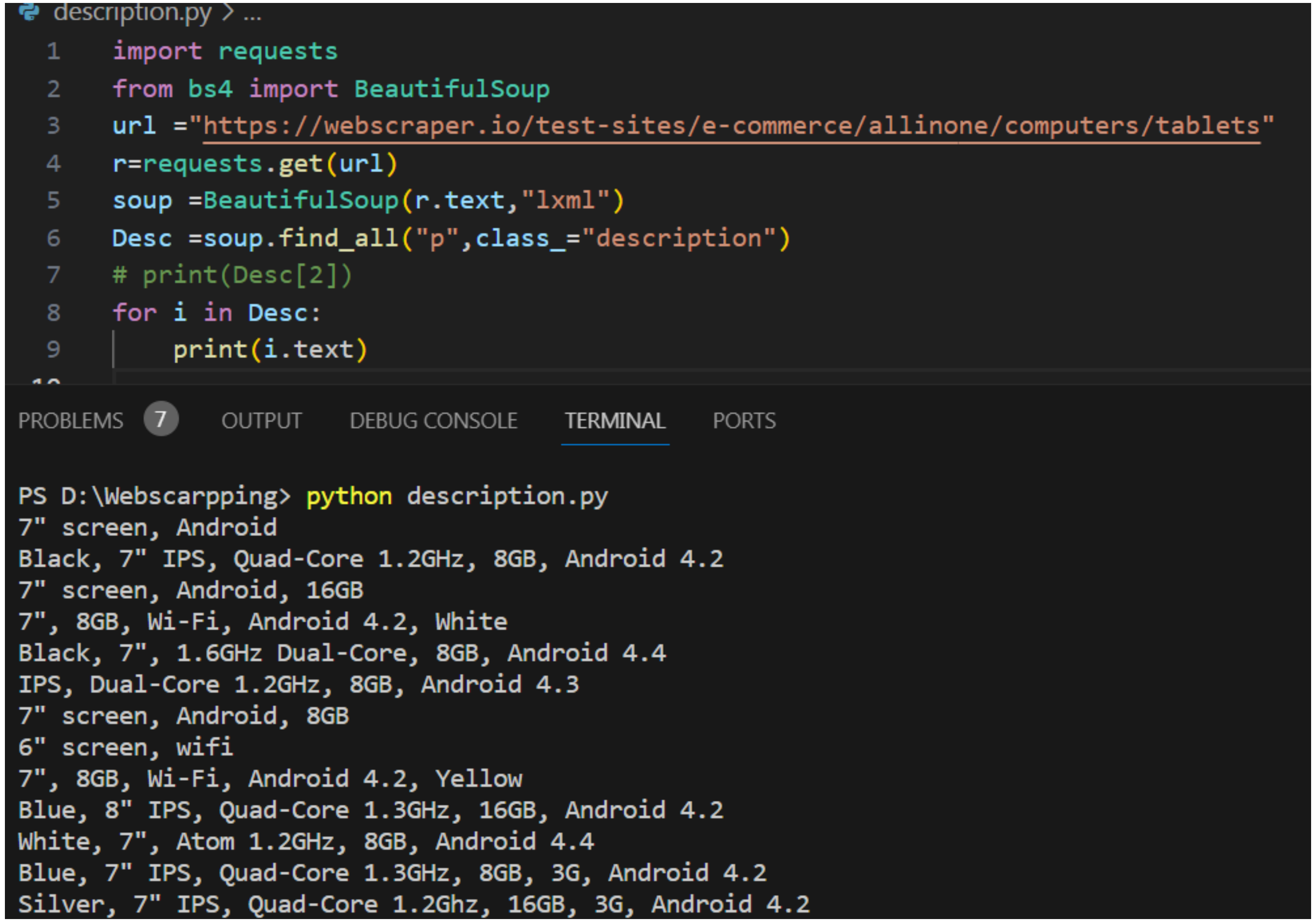
```
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
prices =soup.find_all("h4",class_="pull-right price")
# for i in prices:
#     print(i.text)
print(prices[3])
```

```
PS D:\Webscarpping> python find.py
<h4 class="pull-right price">$97.99</h4>
PS D:\Webscarpping>
```



```
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
data=soup.find_all(["h4","a","p"])
print(data)
```

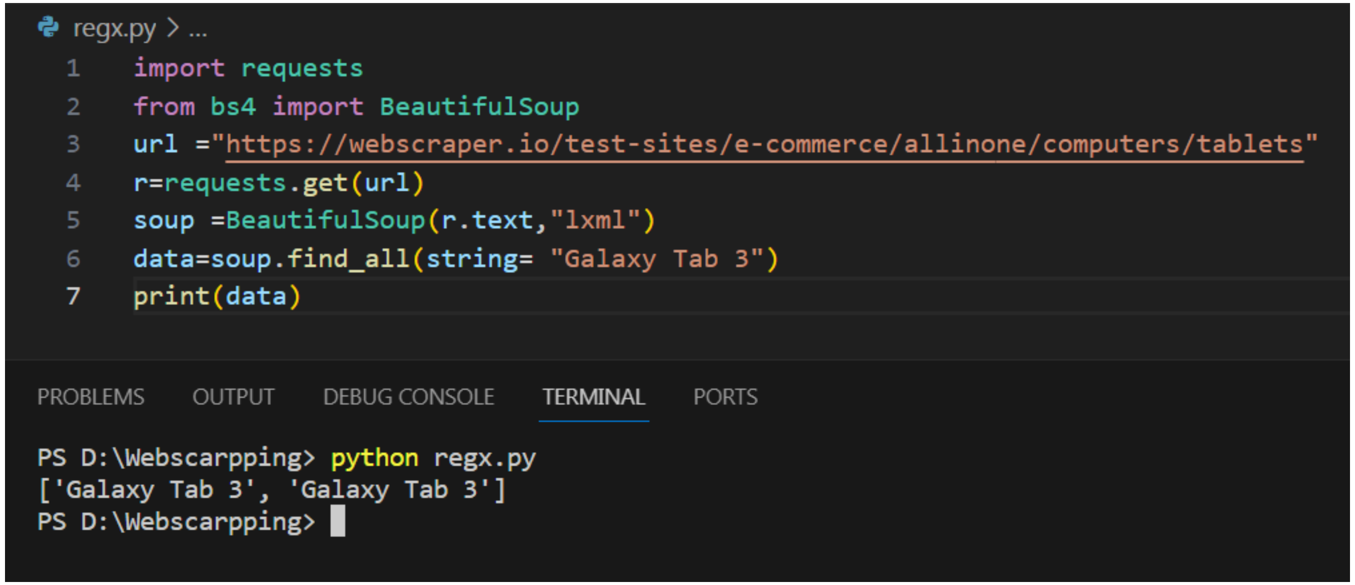
```
<p class="description">7" screen, Android, 16GB</p>
PS D:\Webscarpping> python description.py
7" screen, Android
PS D:\Webscarpping> python description.py
7" screen, Android
PS D:\Webscarpping> python regx.py
[<a data-target=".side-collapse" data-target-2=".side-collapse-container" data-toggle="collapse-
<button aria-controls="navbar" aria-expanded="false" class="navbar-toggle pull-right collapsed"
collapse-container" data-target-3=".side-collapse" data-toggle="collapse" type="button">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar top-bar"></span>
```



```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
Desc =soup.find_all("p",class_="description")
# print(Desc[2])
for i in Desc:
    print(i.text)
```

```
PS D:\Webscarpping> python description.py
7" screen, Android
Black, 7" IPS, Quad-Core 1.2GHz, 8GB, Android 4.2
7" screen, Android, 16GB
7", 8GB, Wi-Fi, Android 4.2, White
Black, 7", 1.6GHz Dual-Core, 8GB, Android 4.4
IPS, Dual-Core 1.2GHz, 8GB, Android 4.3
7" screen, Android, 8GB
6" screen, wifi
7", 8GB, Wi-Fi, Android 4.2, Yellow
Blue, 8" IPS, Quad-Core 1.3GHz, 16GB, Android 4.2
White, 7", Atom 1.2GHz, 8GB, Android 4.4
Blue, 7" IPS, Quad-Core 1.3GHz, 8GB, 3G, Android 4.2
Silver, 7" IPS, Quad-Core 1.2Ghz, 16GB, 3G, Android 4.2
```

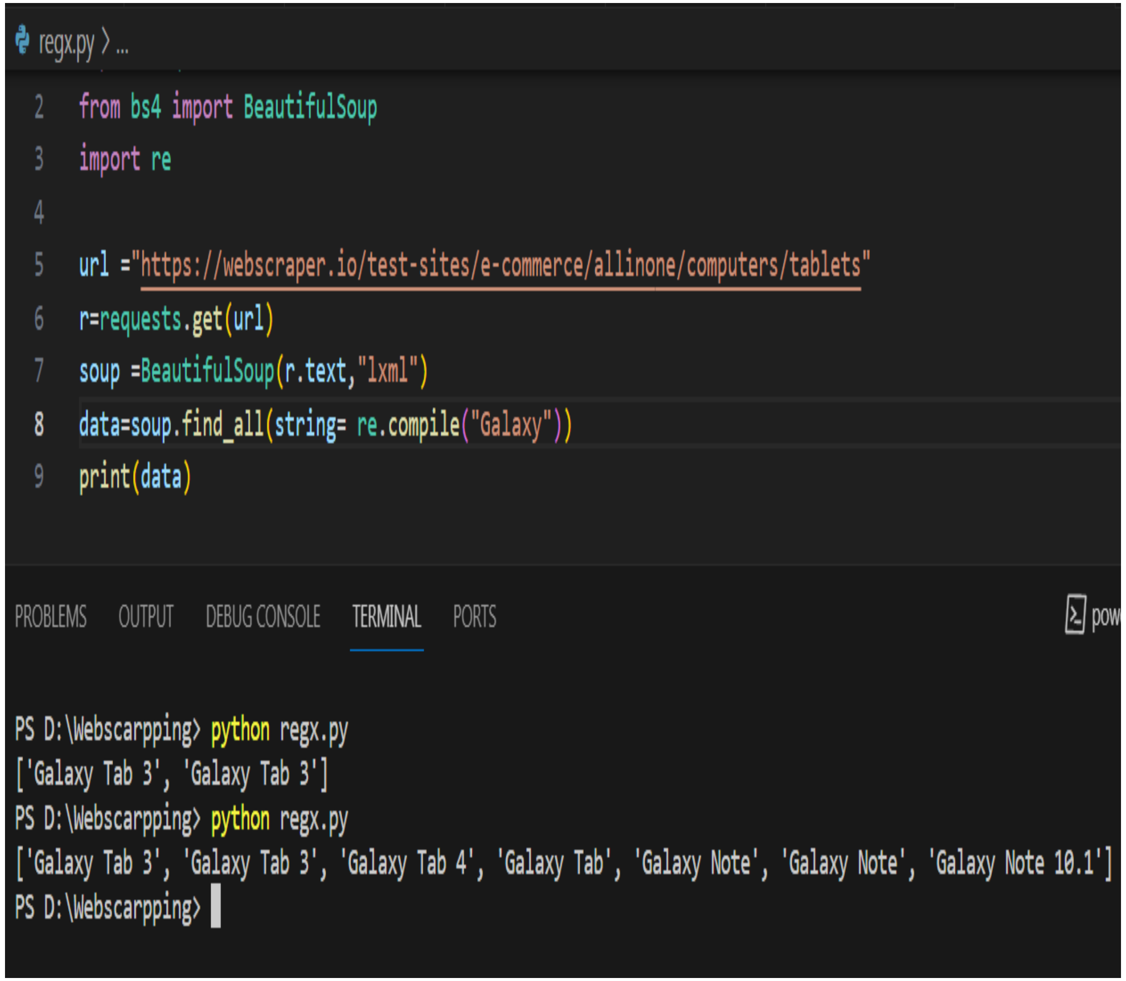
Search with String



```python
import requests
from bs4 import BeautifulSoup
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
data=soup.find_all(string= "Galaxy Tab 3")
print(data)
```

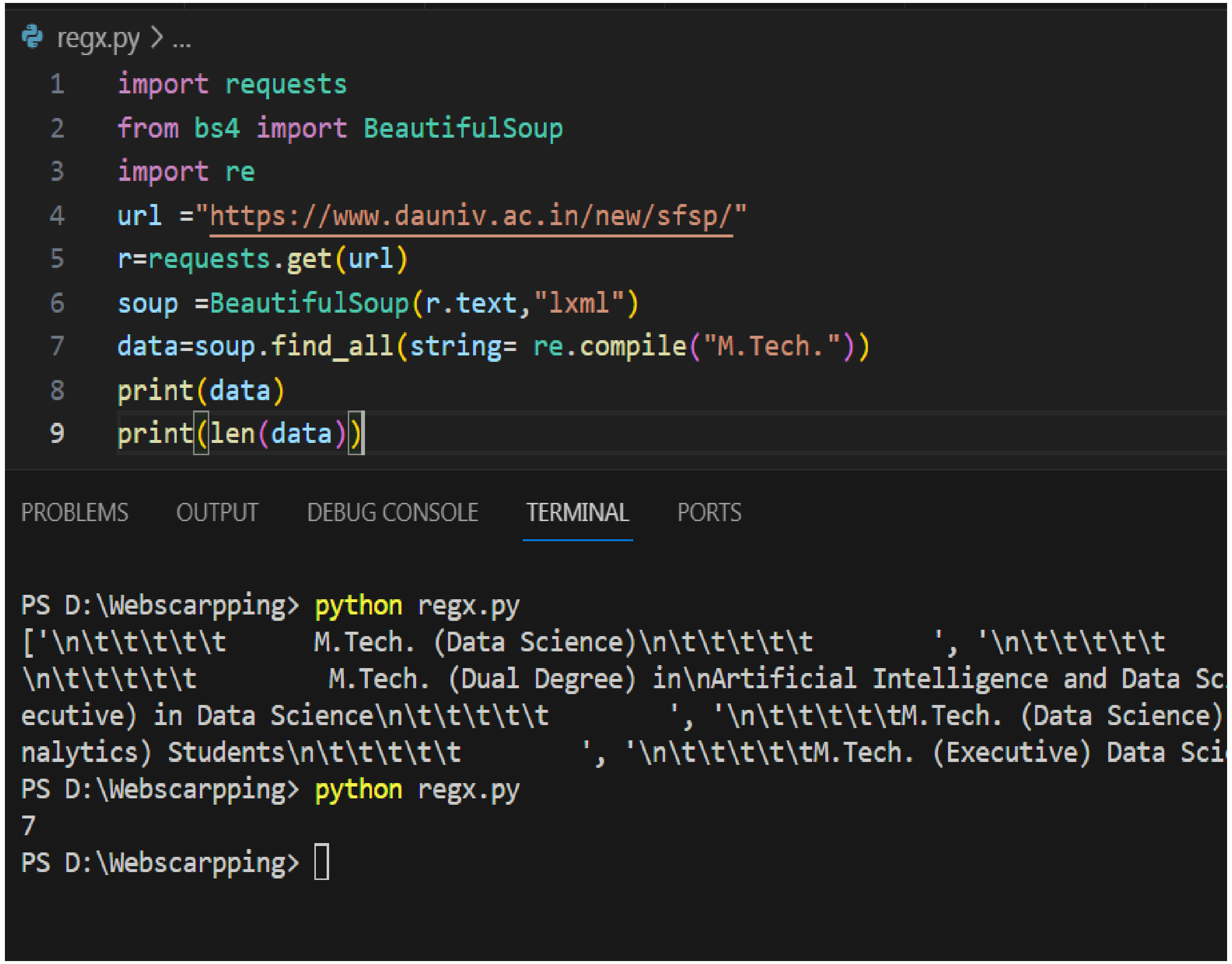
```
PS D:\Webscarpping> python regx.py
['Galaxy Tab 3', 'Galaxy Tab 3']
PS D:\Webscarpping>
```

Beautifulsoup -Find_all() with RegEx



```python
from bs4 import BeautifulSoup
import re

url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
data=soup.find_all(string= re.compile("Galaxy"))
print(data)
```

```
PS D:\Webscarpping> python regx.py
['Galaxy Tab 3', 'Galaxy Tab 3']
PS D:\Webscarpping> python regx.py
['Galaxy Tab 3', 'Galaxy Tab 3', 'Galaxy Tab 4', 'Galaxy Tab', 'Galaxy Note', 'Galaxy Note', 'Galaxy Note 10.1']
PS D:\Webscarpping>
```

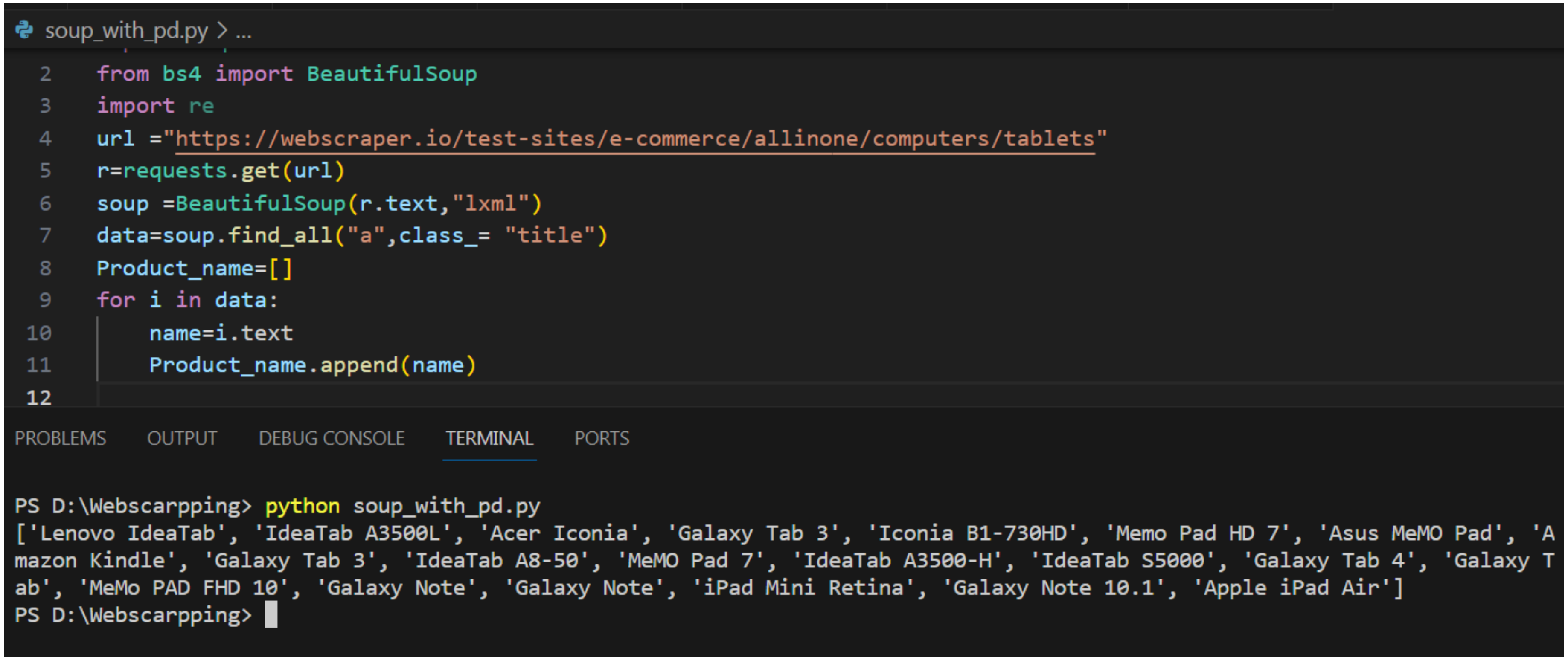


```python
import requests
from bs4 import BeautifulSoup
import re
url ="https://www.dauniv.ac.in/new/sfsp/"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
data=soup.find_all(string= re.compile("M.Tech."))
print(data)
print(len(data))
```

```
PS D:\Webscarpping> python regx.py
['\n\t\t\t\t\t      M.Tech. (Data Science)\n\t\t\t\t\t        ', '\n\t\t\t\t\t
\n\t\t\t\t\t        M.Tech. (Dual Degree) in\nArtificial Intelligence and Data Sc
ecutive) in Data Science\n\t\t\t\t\t        ', '\n\t\t\t\t\tM.Tech. (Data Science)
nalytics) Students\n\t\t\t\t\t        ', '\n\t\t\t\t\tM.Tech. (Executive) Data Sci
PS D:\Webscarpping> python regx.py
7
PS D:\Webscarpping>
```

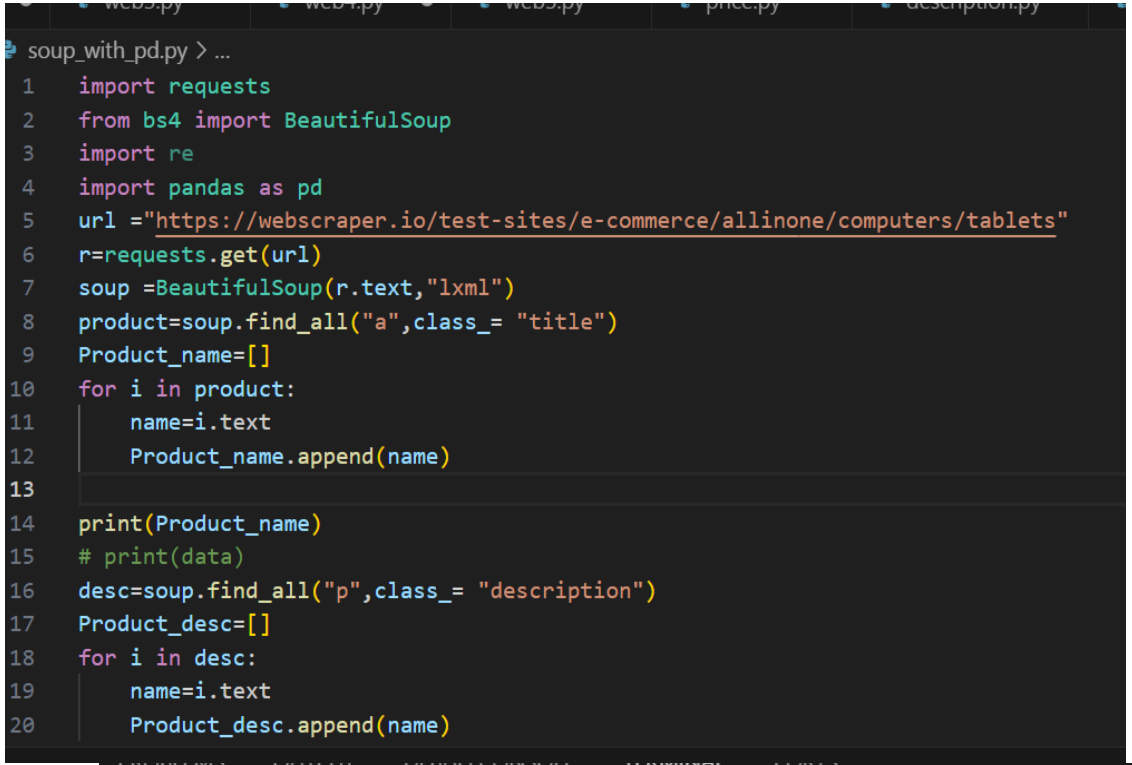
Web Scraping with Beautiful Soup and Pandas

We scrapped all the data of Product name, Description and review form given link

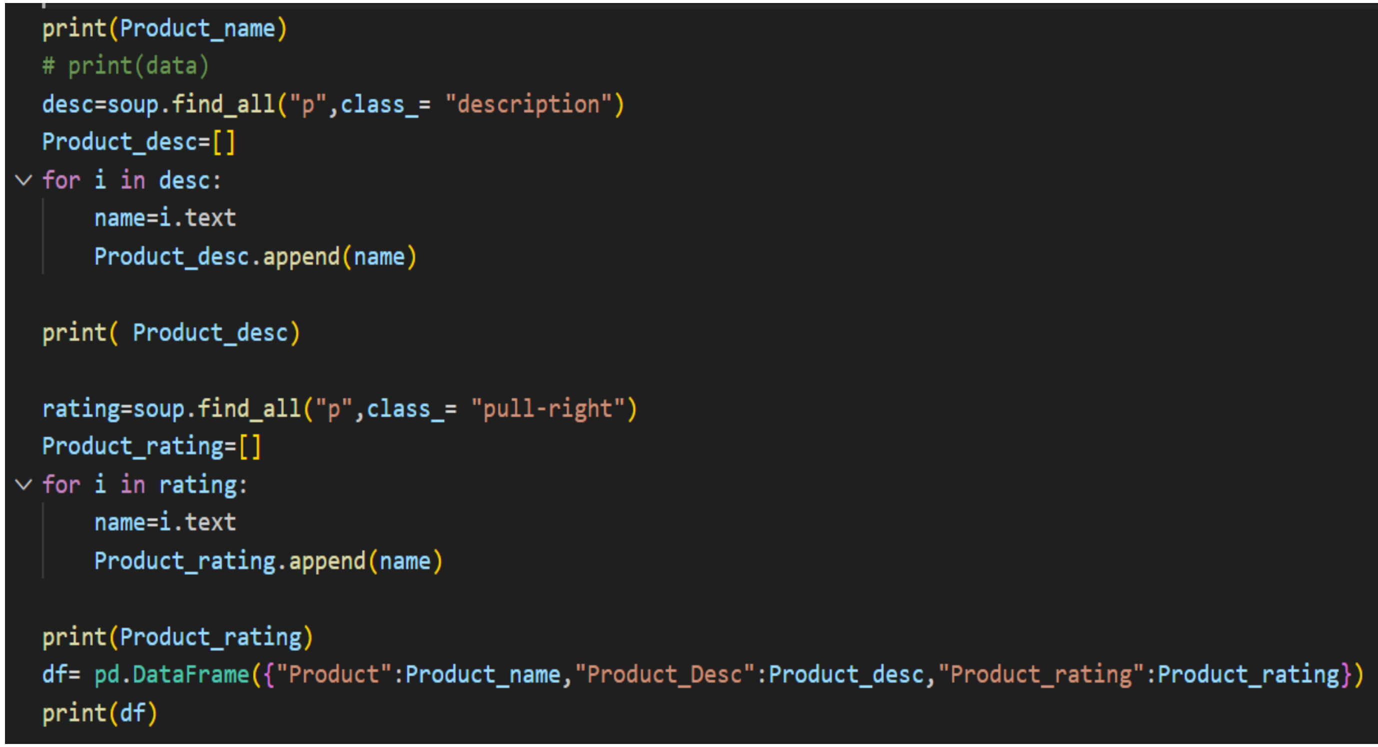


```python
from bs4 import BeautifulSoup
import re
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
data=soup.find_all("a",class_= "title")
Product_name=[]
for i in data:
    name=i.text
    Product_name.append(name)

```

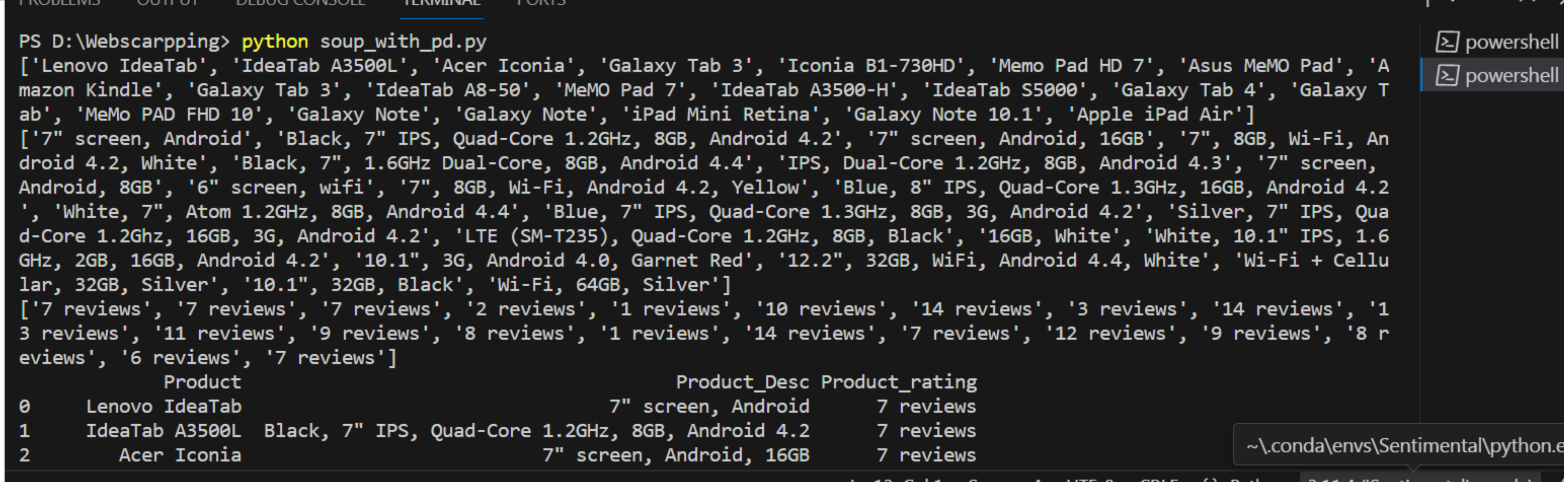
```
PS D:\Webscarpping> python soup_with_pd.py
['Lenovo IdeaTab', 'IdeaTab A3500L', 'Acer Iconia', 'Galaxy Tab 3', 'Iconia B1-730HD', 'Memo Pad HD 7', 'Asus MeMO Pad', 'A
mazon Kindle', 'Galaxy Tab 3', 'IdeaTab A8-50', 'MeMO Pad 7', 'IdeaTab A3500-H', 'IdeaTab S5000', 'Galaxy Tab 4', 'Galaxy T
ab', 'MeMo PAD FHD 10', 'Galaxy Note', 'Galaxy Note', 'iPad Mini Retina', 'Galaxy Note 10.1', 'Apple iPad Air']
PS D:\Webscarpping> 
```



```python
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find_all("a",class_= "title")
Product_name=[]
for i in product:
    name=i.text
    Product_name.append(name)

print(Product_name)
# print(data)
desc=soup.find_all("p",class_= "description")
Product_desc=[]
for i in desc:
    name=i.text
    Product_desc.append(name)
```

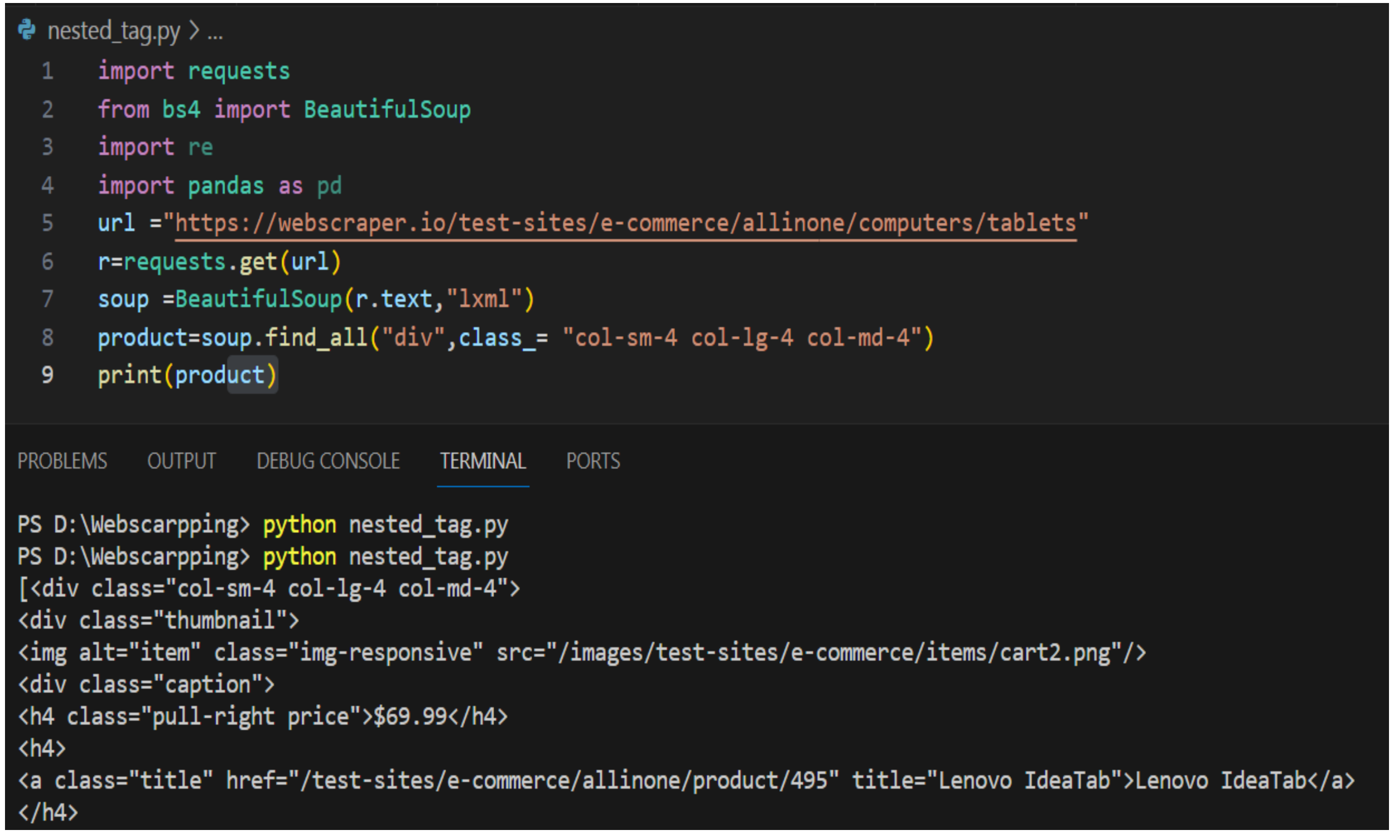


```python
print(Product_name)
# print(data)
desc=soup.find_all("p",class_= "description")
Product_desc=[]
for i in desc:
    name=i.text
    Product_desc.append(name)

print( Product_desc)

rating=soup.find_all("p",class_= "pull-right")
Product_rating=[]
for i in rating:
    name=i.text
    Product_rating.append(name)

print(Product_rating)
df= pd.DataFrame({"Product":Product_name,"Product_Desc":Product_desc,"Product_rating":Product_rating})
print(df)
```



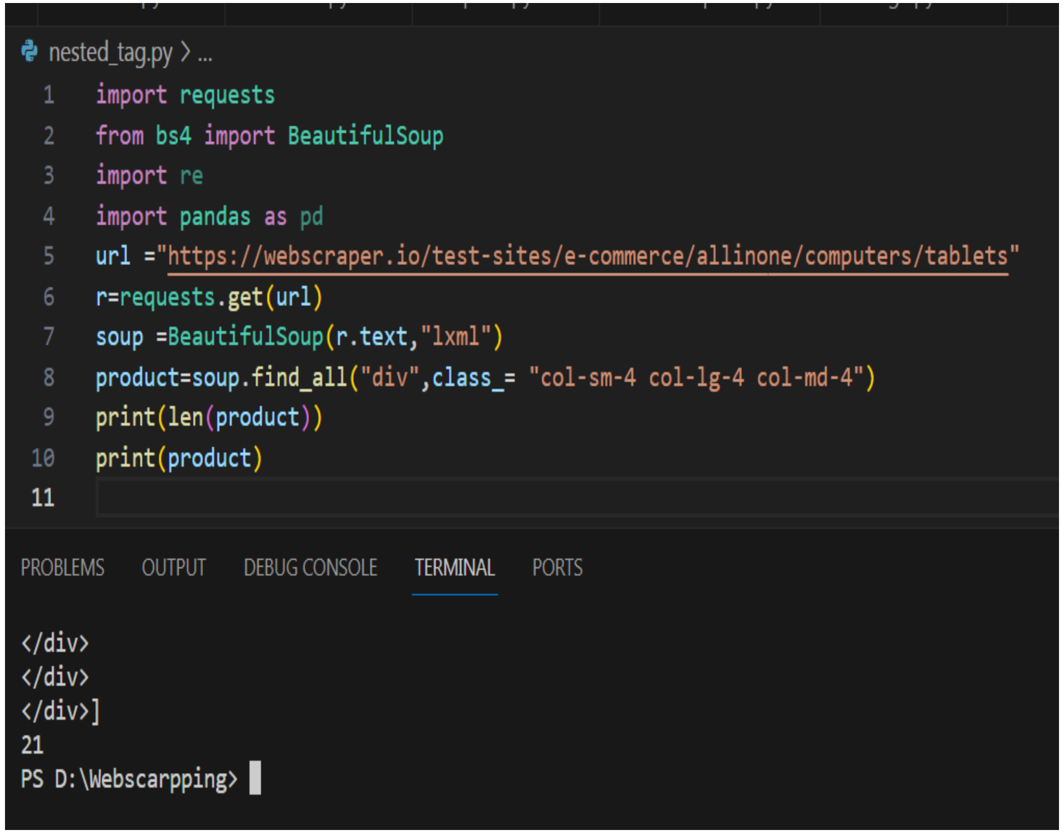
```
PS D:\Webscarpping> python soup_with_pd.py
['Lenovo IdeaTab', 'IdeaTab A3500L', 'Acer Iconia', 'Galaxy Tab 3', 'Iconia B1-730HD', 'Memo Pad HD 7', 'Asus MeMO Pad', 'A
mazon Kindle', 'Galaxy Tab 3', 'IdeaTab A8-50', 'MeMO Pad 7', 'IdeaTab A3500-H', 'IdeaTab S5000', 'Galaxy Tab 4', 'Galaxy T
ab', 'MeMo PAD FHD 10', 'Galaxy Note', 'Galaxy Note', 'iPad Mini Retina', 'Galaxy Note 10.1', 'Apple iPad Air']
['7" screen, Android', 'Black, 7" IPS, Quad-Core 1.2GHz, 8GB, Android 4.2', '7" screen, Android, 16GB', '7", 8GB, Wi-Fi, An
droid 4.2, White', 'Black, 7", 1.6GHz Dual-Core, 8GB, Android 4.4', 'IPS, Dual-Core 1.2GHz, 8GB, Android 4.3', '7" screen,
Android, 8GB', '6" screen, wifi', '7", 8GB, Wi-Fi, Android 4.2, Yellow', 'Blue, 8" IPS, Quad-Core 1.3GHz, 16GB, Android 4.2
', 'White, 7", Atom 1.2GHz, 8GB, Android 4.4', 'Blue, 7" IPS, Quad-Core 1.3GHz, 8GB, 3G, Android 4.2', 'Silver, 7" IPS, Qua
d-Core 1.2Ghz, 16GB, 3G, Android 4.2', 'LTE (SM-T235), Quad-Core 1.2GHz, 8GB, Black', '16GB, White', 'White, 10.1" IPS, 1.6
GHz, 2GB, 16GB, Android 4.2', '10.1", 3G, Android 4.0, Garnet Red', '12.2", 32GB, WiFi, Android 4.4, White', 'Wi-Fi + Cellu
lar, 32GB, Silver', '10.1", 32GB, Black', 'Wi-Fi, 64GB, Silver']
['7 reviews', '7 reviews', '7 reviews', '2 reviews', '1 reviews', '10 reviews', '14 reviews', '3 reviews', '14 reviews', '1
3 reviews', '11 reviews', '9 reviews', '8 reviews', '1 reviews', '14 reviews', '7 reviews', '12 reviews', '9 reviews', '8 r
eviews', '6 reviews', '7 reviews']
          Product                                       Product_Desc Product_rating
0     Lenovo IdeaTab                                 7" screen, Android      7 reviews
1     IdeaTab A3500L  Black, 7" IPS, Quad-Core 1.2GHz, 8GB, Android 4.2      7 reviews
2        Acer Iconia                           7" screen, Android, 16GB      7 reviews
```

Extract Data from Nested HTML Tags



```python
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find_all("div",class_= "col-sm-4 col-lg-4 col-md-4")
print(product)
```

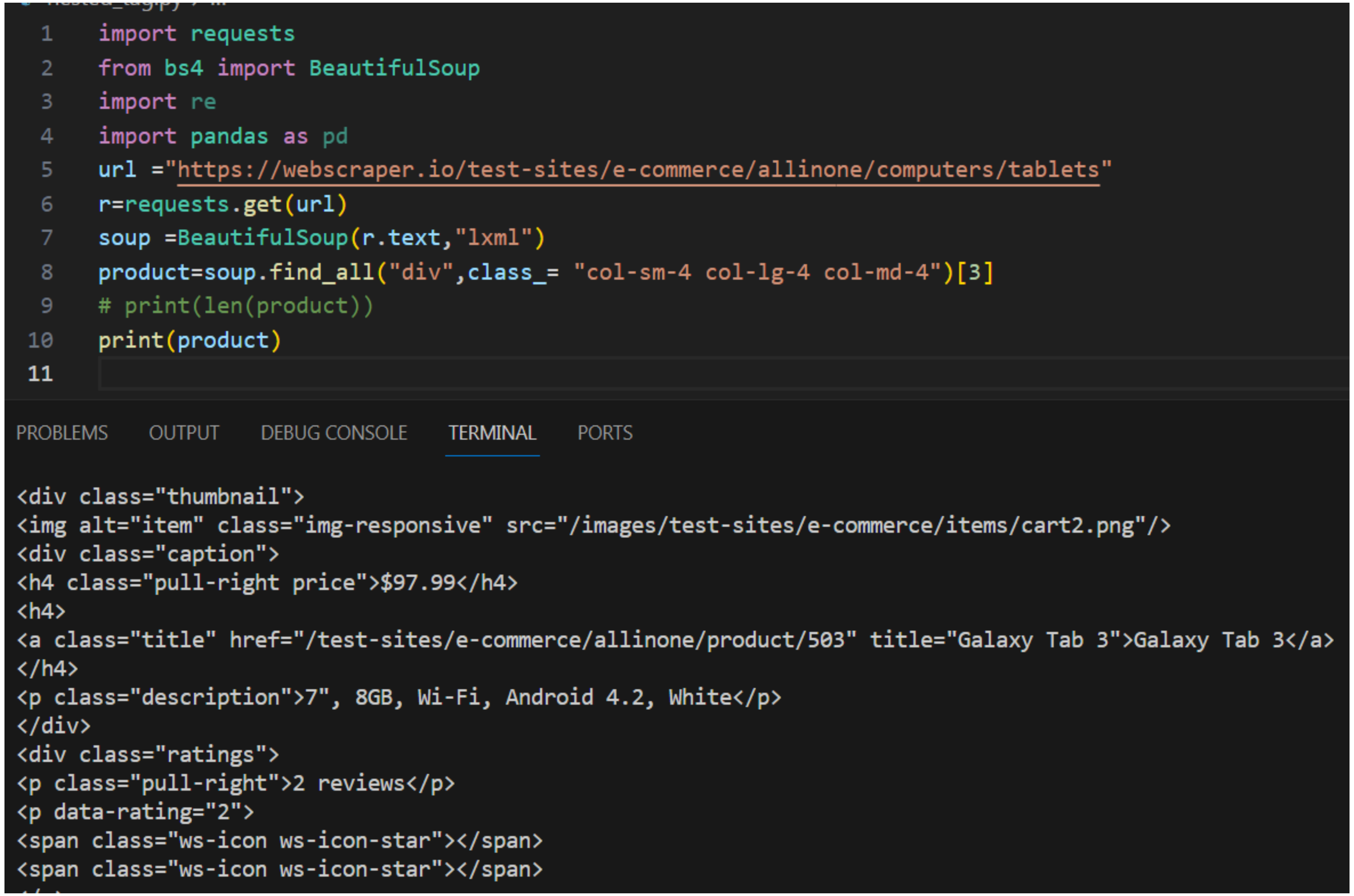
```
PS D:\Webscarpping> python nested_tag.py
PS D:\Webscarpping> python nested_tag.py
[<div class="col-sm-4 col-lg-4 col-md-4">
<div class="thumbnail">
<img alt="item" class="img-responsive" src="/images/test-sites/e-commerce/items/cart2.png"/>
<div class="caption">
<h4 class="pull-right price">$69.99</h4>
<h4>
<a class="title" href="/test-sites/e-commerce/allinone/product/495" title="Lenovo IdeaTab">Lenovo IdeaTab</a>
</h4>
```



```python
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find_all("div",class_= "col-sm-4 col-lg-4 col-md-4")
print(len(product))
print(product)

```

```
</div>
</div>
</div>]
21
PS D:\Webscarpping>
```

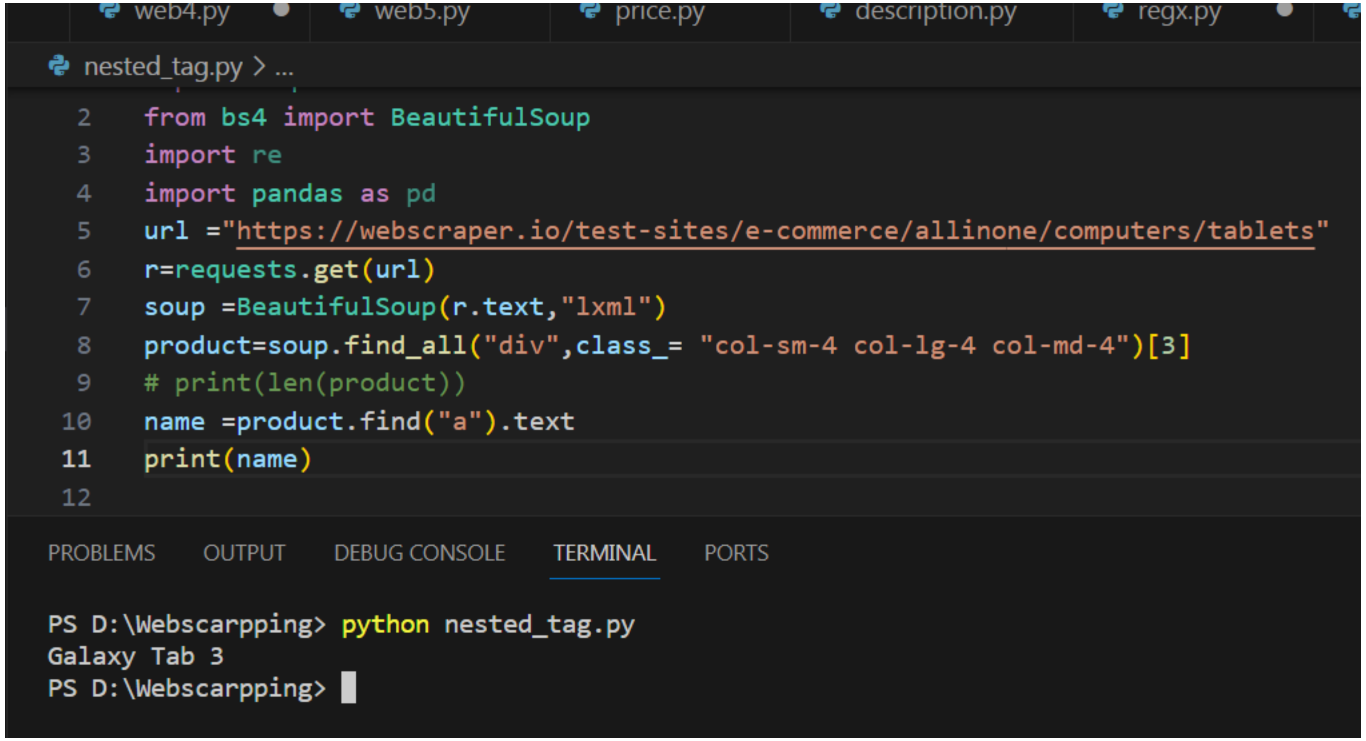
Extract particular Data from Nested HTML Tags



```python
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find_all("div",class_= "col-sm-4 col-lg-4 col-md-4")[3]
# print(len(product))
print(product)

```

```
<div class="thumbnail">
<img alt="item" class="img-responsive" src="/images/test-sites/e-commerce/items/cart2.png"/>
<div class="caption">
<h4 class="pull-right price">$97.99</h4>
<h4>
<a class="title" href="/test-sites/e-commerce/allinone/product/503" title="Galaxy Tab 3">Galaxy Tab 3</a>
</h4>
<p class="description">7", 8GB, Wi-Fi, Android 4.2, White</p>
</div>
<div class="ratings">
<p class="pull-right">2 reviews</p>
<p data-rating="2">
<span class="ws-icon ws-icon-star"></span>
<span class="ws-icon ws-icon-star"></span>
```

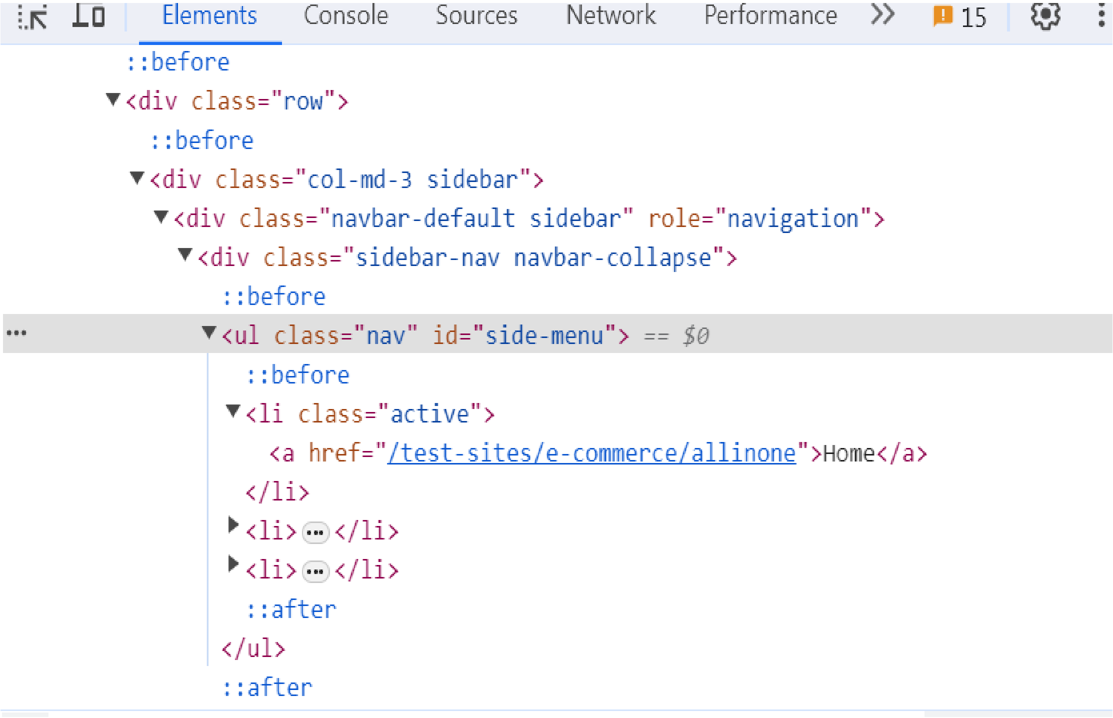
Print Particular name



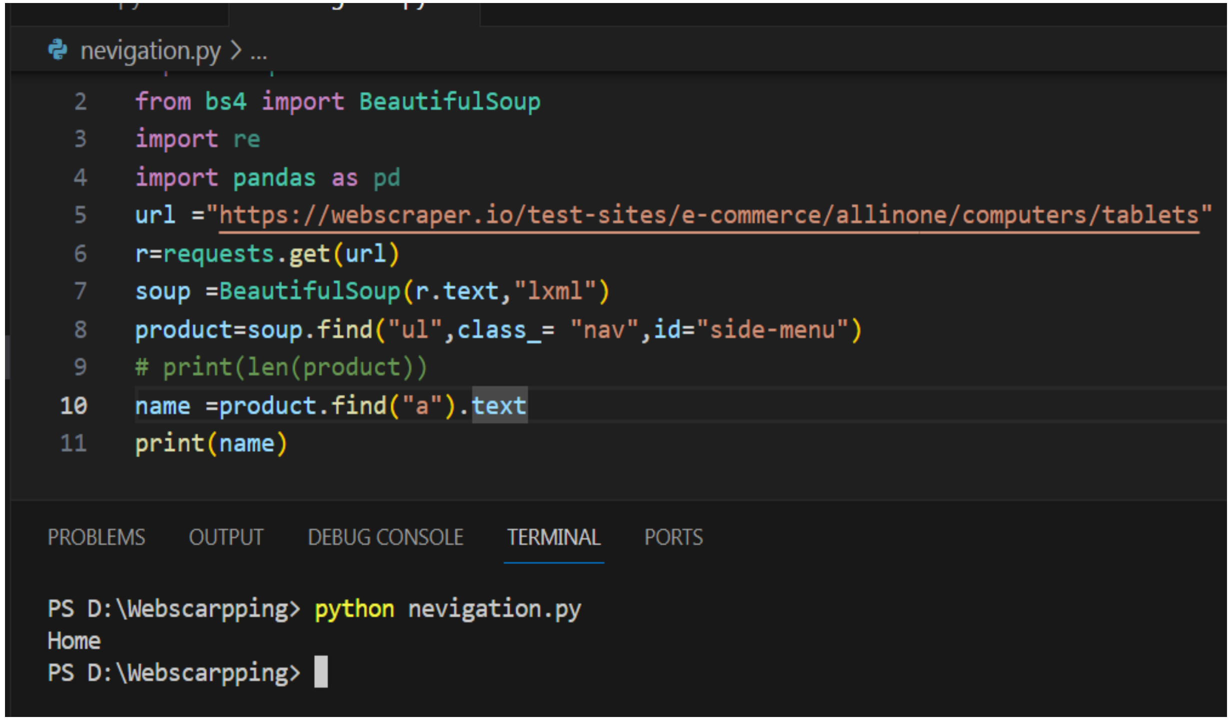
```python
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find_all("div",class_= "col-sm-4 col-lg-4 col-md-4")[3]
# print(len(product))
name =product.find("a").text
print(name)

```

```
PS D:\Webscarpping> python nested_tag.py
Galaxy Tab 3
PS D:\Webscarpping> 
```

Web Scrapping

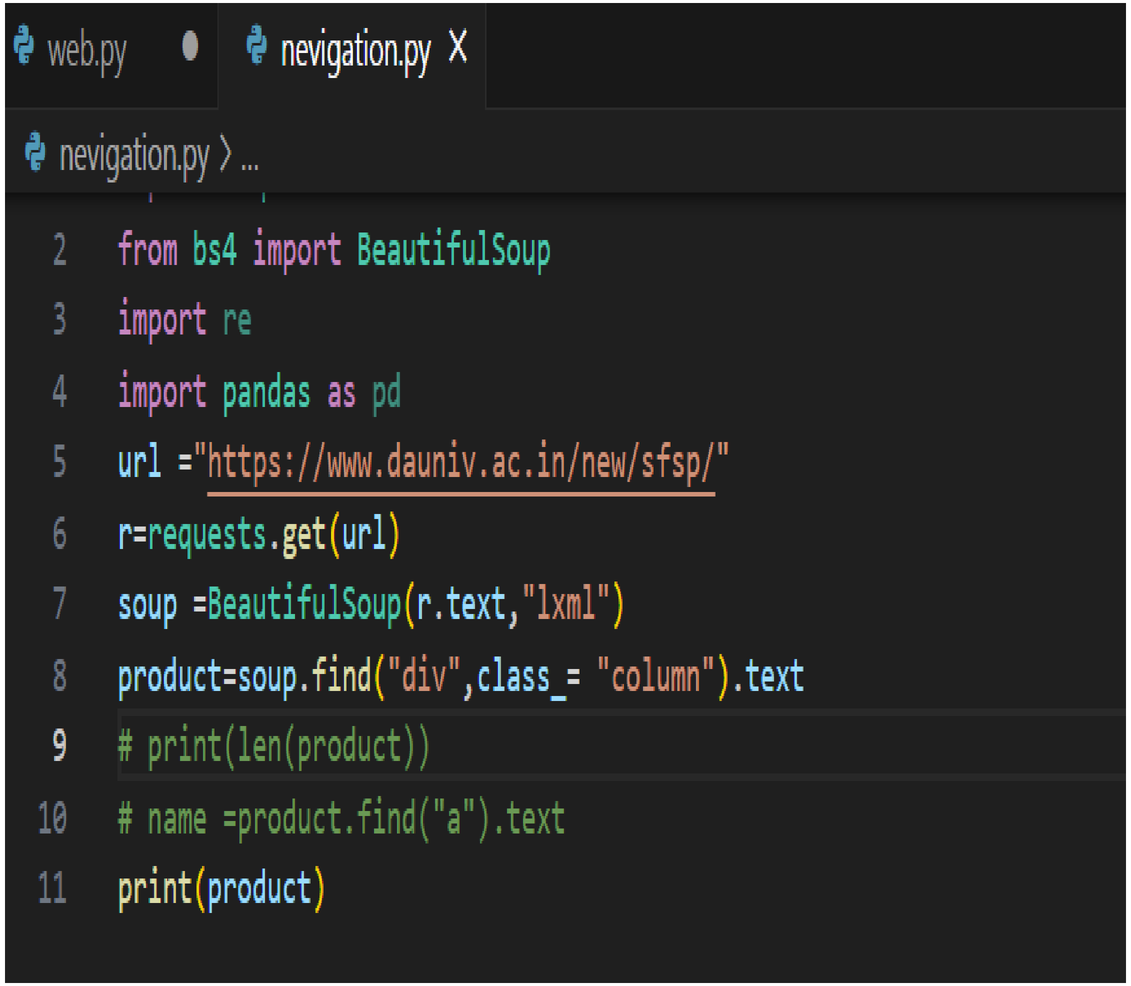


```
    ::before
  ▼<div class="row">
     ::before
   ▼<div class="col-md-3 sidebar">
     ▼<div class="navbar-default sidebar" role="navigation">
       ▼<div class="sidebar-nav navbar-collapse">
          ::before
         ▼<ul class="nav" id="side-menu"> == $0
            ::before
           ▼<li class="active">
              <a href="/test-sites/e-commerce/allinone">Home</a>
            </li>
           ▶<li>…</li>
           ▶<li>…</li>
            ::after
          </ul>
          ::after
```

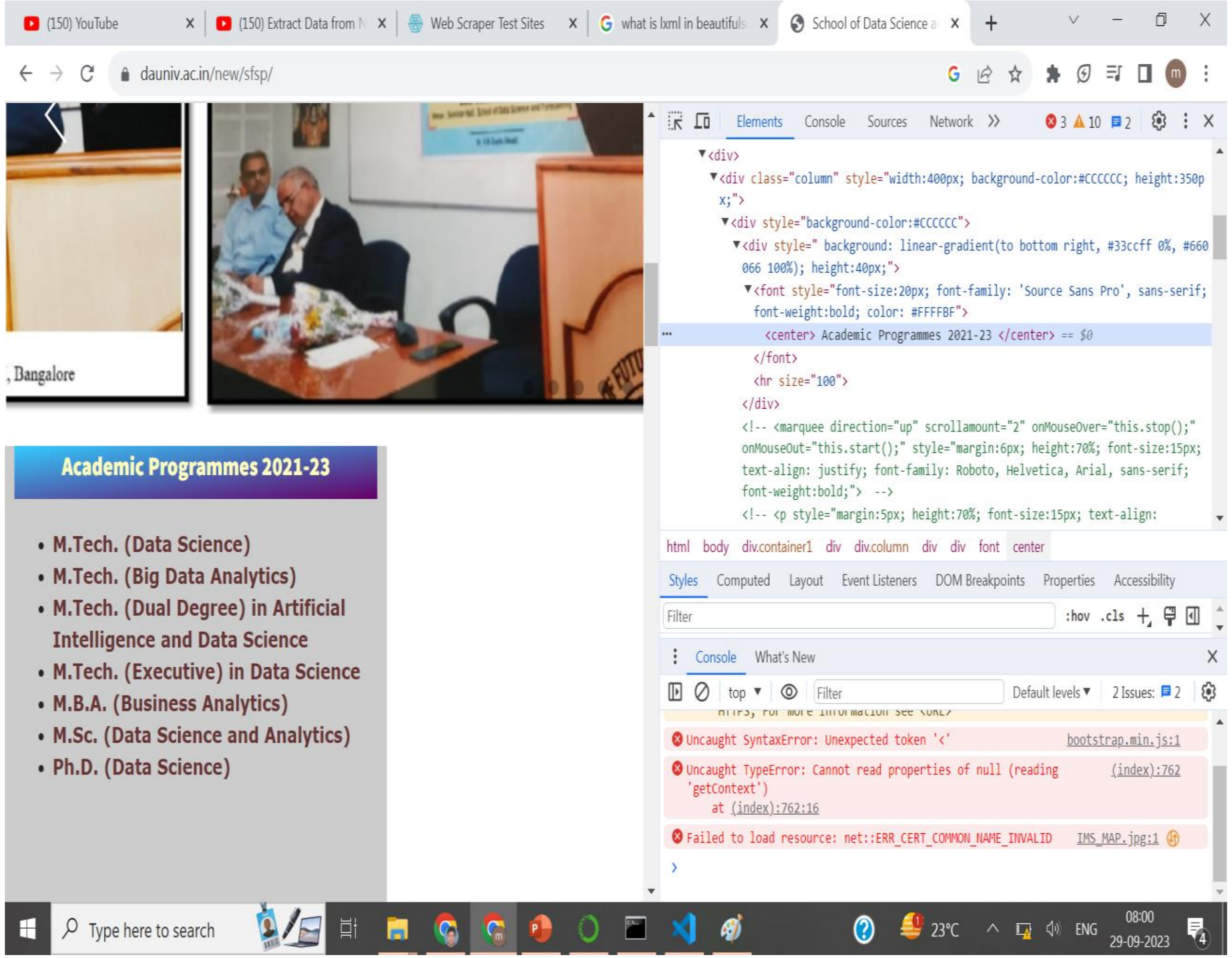


```python
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://webscraper.io/test-sites/e-commerce/allinone/computers/tablets"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find("ul",class_= "nav",id="side-menu")
# print(len(product))
name =product.find("a").text
print(name)
```

```
PS D:\Webscarpping> python nevigation.py
Home
PS D:\Webscarpping> 
```



```python
from bs4 import BeautifulSoup
import re
import pandas as pd
url ="https://www.dauniv.ac.in/new/sfsp/"
r=requests.get(url)
soup =BeautifulSoup(r.text,"lxml")
product=soup.find("div",class_= "column").text
# print(len(product))
# name =product.find("a").text
print(product)
```



Academic Programmes 2021-23

- M.Tech. (Data Science)
- M.Tech. (Big Data Analytics)
- M.Tech. (Dual Degree) in Artificial Intelligence and Data Science
- M.Tech. (Executive) in Data Science
- M.B.A. (Business Analytics)
- M.Sc. (Data Science and Analytics)
- Ph.D. (Data Science)

```html
▼<div>
  ▼<div class="column" style="width:400px; background-color:#CCCCCC; height:350p
  x;">
    ▼<div style="background-color:#CCCCCC">
      ▼<div style=" background: linear-gradient(to bottom right, #33ccff 0%, #660
      066 100%); height:40px;">
        ▼<font style="font-size:20px; font-family: 'Source Sans Pro', sans-serif;
        font-weight:bold; color: #FFFFBF">
            <center> Academic Programmes 2021-23 </center> == $0
         </font>
         <hr size="100">
      </div>
      <!-- <marquee direction="up" scrollamount="2" onMouseOver="this.stop();"
      onMouseOut="this.start();" style="margin:6px; height:70%; font-size:15px;
      text-align: justify; font-family: Roboto, Helvetica, Arial, sans-serif;
      font-weight:bold;">  -->
      <!-- <p style="margin:5px; height:70%; font-size:15px; text-align:
```

```
HTTPS, For more information see <URL>
Uncaught SyntaxError: Unexpected token '<'                    bootstrap.min.js:1
Uncaught TypeError: Cannot read properties of null (reading   (index):762
'getContext')
    at (index):762:16
Failed to load resource: net::ERR_CERT_COMMON_NAME_INVALID    IMS_MAP.jpg:1
>
```

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping

Web Scrapping