# Batch: H-ADS (H2\_3) Roll No.: 16010122221

**Experiment No. 9**

**Title :** To implement data scraping using Selenium/ ScraPy

**Aim:** To perform testing using web scrapers in Python

# Expected Outcome of Experiment:

**CO2:** Application of Exploratory data analysis (EDA) on Real world problems.

# Books/ Journals/ Websites referred:

1. https://realpython.com/python-web-scraping-practical-introduction/

**Web Scrapping:**

**Web scraping** is the process of collecting and parsing raw data from the Web.

Web scraping is an automated method used to extract large amounts of data from websites. The data on the websites are unstructured.

Web scraping helps collect these unstructured data and store it in a structured form. There are different ways to scrape websites such as online Services, APIs or writing your own code.

# Why is Web Scraping Used?

* Price Comparison: Services such as ParseHub use web scraping to collect data from online shopping websites and use it to compare the prices of products.
* Email address gathering: Many companies that use email as a medium for marketing, use web scraping to collect email ID and then send bulk emails.
* Social Media Scraping: Web scraping is used to collect data from Social Media websites such as Twitter to find out what’s trending.
* Research and Development: Web scraping is used to collect a large set of data (Statistics, General Information, Temperature, etc.) from websites, which are analyzed and used to carry out Surveys or for R&D.
* Job listings: Details regarding job openings, interviews are collected from different websites and then listed in one place so that it is easily accessible to the user.

# Why is Python Good for Web Scraping?

* Here is the list of features of Python which makes it more suitable for web scraping.
* Ease of Use: Python is simple to code. You do not have to add semi-colons “;” or curly- braces “{}” anywhere. This makes it less messy and easy to use.
* Large Collection of Libraries: Python has a huge collection of libraries such as Numpy, Matlplotlib, Pandas etc., which provides methods and services for various purposes. Hence, it is suitable for web scraping and for further manipulation of extracted data.
* Dynamically typed: In Python, you don’t have to define datatypes for variables, you can directly use the variables wherever required. This saves time and makes your job faster.

# How Do You Scrape Data From A Website?

* When you run the code for web scraping, a request is sent to the URL that you have mentioned. As a response to the request, the server sends the data and allows you to read the HTML or XML page. The code then, parses the HTML or XML page, finds the data and extracts it.
* To extract data using web scraping with python, you need to follow these basic steps:
  + Find the URL that you want to scrape
  + Inspecting the Page
  + Find the data you want to extract
  + Write the code
  + Run the code and extract the data
  + Store the data in the required format

**Libraries used for Web Scraping**

* + **Selenium:** Selenium is a web testing library. It is used to automate browser activities.
  + **BeautifulSoup:** Beautiful Soup is a Python package for parsing HTML and XML documents. It creates parse trees that is helpful to extract the data easily.
  + **Pandas:** Pandas is a library used for data manipulation and analysis. It is used to extract the data and store it in the desired format.

# Implementation:

import bs4

from bs4 import BeautifulSoup as bs import requests

link='https:/[/www.flipka](http://www.flipkart.com/search?q=tv&as=on&as-)r[t.com/search?q=tv&as=on&as-](http://www.flipkart.com/search?q=tv&as=on&as-) show=on&otracker=AS\_Query\_TrendingAutoSuggest\_8\_0\_na\_na\_na&otracker 1=AS\_Query\_TrendingAutoSuggest\_8\_0\_na\_na\_na&as-pos=8&as- type=TRENDING&suggestionId=tv&requestId=9c9fa553-b7e5-454b-a65b- bbb7a9c74a29'

page=requests.get(link) page.content

soup=bs(page.content, 'html.parser') # print(soup.prettify())

#extract name of product name=soup.find('div',class\_="\_4rR01T") print(name.text)

#extract rating of product rating=soup.find('div',class\_="\_3LWZlK") print(rating.text)

#extracting other details details=soup.find('div',class\_="fMghEO") print(details.text)

#to get all details seperately for each in details:

spec=each.find\_all('li',class\_='rgWa7D') print(spec[0].text)

print(spec[1].text) print(spec[2].text) print(spec[4].text) print(spec[5].text) print(spec[7].text)

#extract price of the product price=soup.find('div',class\_='\_30jeq3 \_1\_WHN1') print(price.text)



import bs4

from bs4 import BeautifulSoup as bs import requests

import pandas as pd link='https:/[/www.flipkart.com/search?q=tv&as=on&as](http://www.flipkart.com/search?q=tv&as=on&as-)- show=on&otracker=AS\_Query\_TrendingAutoSuggest\_8\_0\_na\_na\_na&otracker 1=AS\_Query\_TrendingAutoSuggest\_8\_0\_na\_na\_na&as-pos=8&as- type=TRENDING&suggestionId=tv&requestId=9c9fa553-b7e5-454b-a65b- bbb7a9c74a29'

page=requests.get(link) page.content

soup=bs(page.content, 'html.parser') # print(soup.prettify())

products=[] #List to store the name of the product prices=[] #List to store price of the product ratings=[] #List to store rating of the product apps = [] #List to store supported apps

os = [] #List to store operating system hd = [] #List to store resolution

sound = [] #List to store sound output

for data in soup.findAll('div',class\_='\_3pLy-c row'): names=data.find('div', attrs={'class':'\_4rR01T'}) price=data.find('div', attrs={'class':'\_30jeq3 \_1\_WHN1'}) rating=data.find('div', attrs={'class':'\_3LWZlK'}) specification = data.find('div', attrs={'class':'fMghEO'})

for each in specification: col=each.find\_all('li', attrs={'class':'rgWa7D'}) app =col[0].text

os\_ = col[1].text hd\_ = col[2].text sound\_ = col[3].text

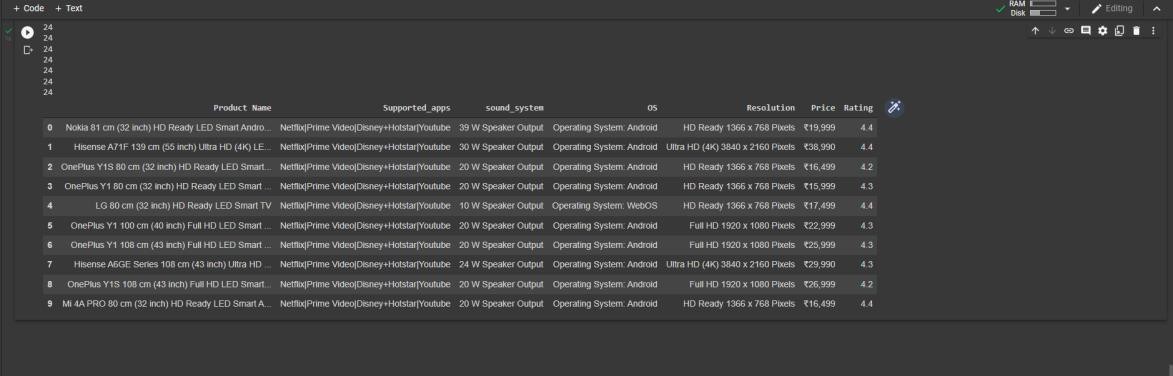
products.append(names.text) # Add product name to list

prices.append(price.text) # Add price to list apps.append(app)# Add supported apps specifications to list os.append(os\_) # Add operating system specifications to list hd.append(hd\_) # Add resolution specifications to list sound.append(sound\_) # Add sound specifications to list ratings.append(rating.text) #Add rating specifications tolist #printing the length of list

print(len(products)) print(len(ratings)) print(len(prices)) print(len(apps)) print(len(sound)) print(len(os)) print(len(hd)) df=pd.DataFrame({'Product

Name':products,'Supported\_apps':apps,'sound\_system':sound,'OS':os,"Resolu tion":hd,'Price':prices,'Rating':ratings})

df.head(10)



# Conclusion:

The implementation of data scraping using Selenium and Scrapy proved to be a robust approach for extracting information from web sources. Both tools demonstrated their efficacy in navigating complex web structures and retrieving targeted data efficiently. While Selenium offered more flexibility in handling dynamic content through browser automation, Scrapy excelled in scalability and performance with its asynchronous processing capabilities. Overall, leveraging these tools opens avenues for streamlined and automated data extraction processes, empowering users with valuable insights from online sources.

# Post lab Questions:

* 1. **What are the different HTTP response status codes?**

HTTP response status codes are standardized codes that indicate the outcome of an HTTP request. They are sent by a server in response to a client's request. Here are some of the most common HTTP response status codes:

# 1xx Informational:

* + 100 Continue
  + 101 Switching Protocols
  + 102 Processing (WebDAV; RFC 2518)

# 2xx Success:

* + 200 OK
  + 201 Created
  + 202 Accepted
  + 204 No Content
  + 206 Partial Content

# 3xx Redirection:

* + 300 Multiple Choices
  + 301 Moved Permanently
  + 302 Found (previously "Moved Temporarily")
  + 304 Not Modified
  + 307 Temporary Redirect
  + 308 Permanent Redirect

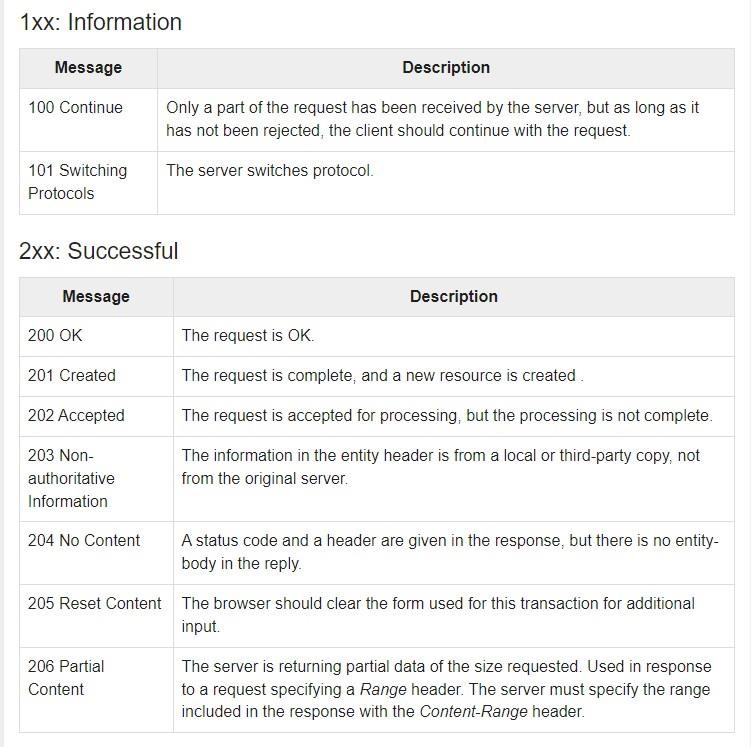
# 4xx Client Error:

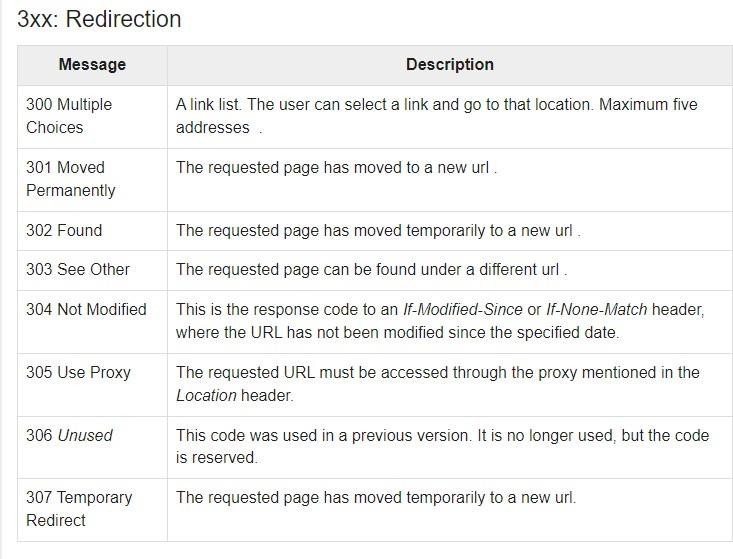
* + 400 Bad Request
  + 401 Unauthorized
  + 403 Forbidden
  + 404 Not Found
  + 405 Method Not Allowed
  + 409 Conflict
  + 410 Gone
  + 429 Too Many Requests

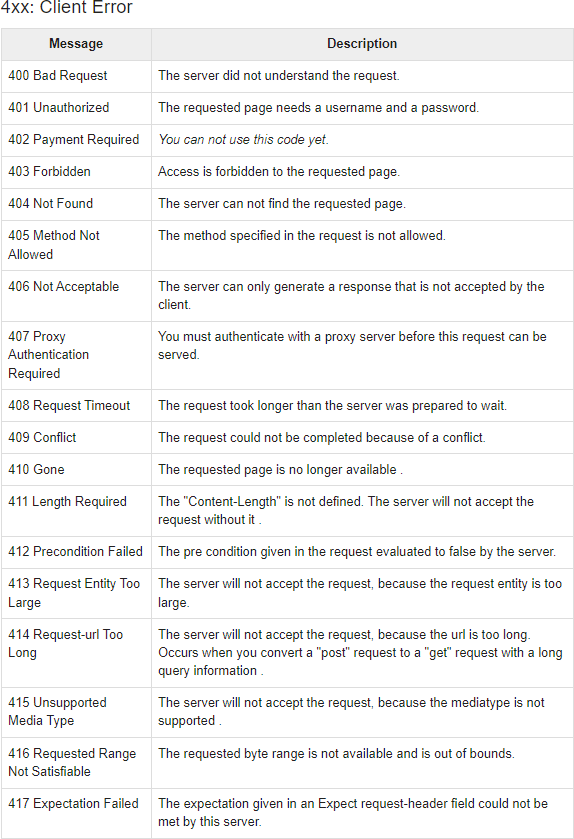
# 5xx Server Error:

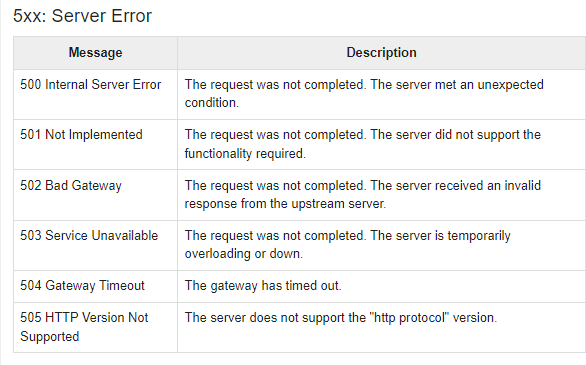
* + 500 Internal Server Error
  + 501 Not Implemented
  + 502 Bad Gateway
  + 503 Service Unavailable
  + 504 Gateway Timeout
  + 505 HTTP Version Not Supported











# How to get the Updated Daily News using Python

import requests

from bs4 import BeautifulSoup

url = 'https://[www.bbc.com/news'](http://www.bbc.com/news%27) response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser') headlines = soup.find('body').find\_all('h3')

unwanted = ['BBC World News TV', 'BBC World Service Radio', 'News daily newsletter', 'Mobile app', 'Get in touch']

for x in list(dict.fromkeys(headlines)): if x.text.strip() not in unwanted:

print(x.text.strip())