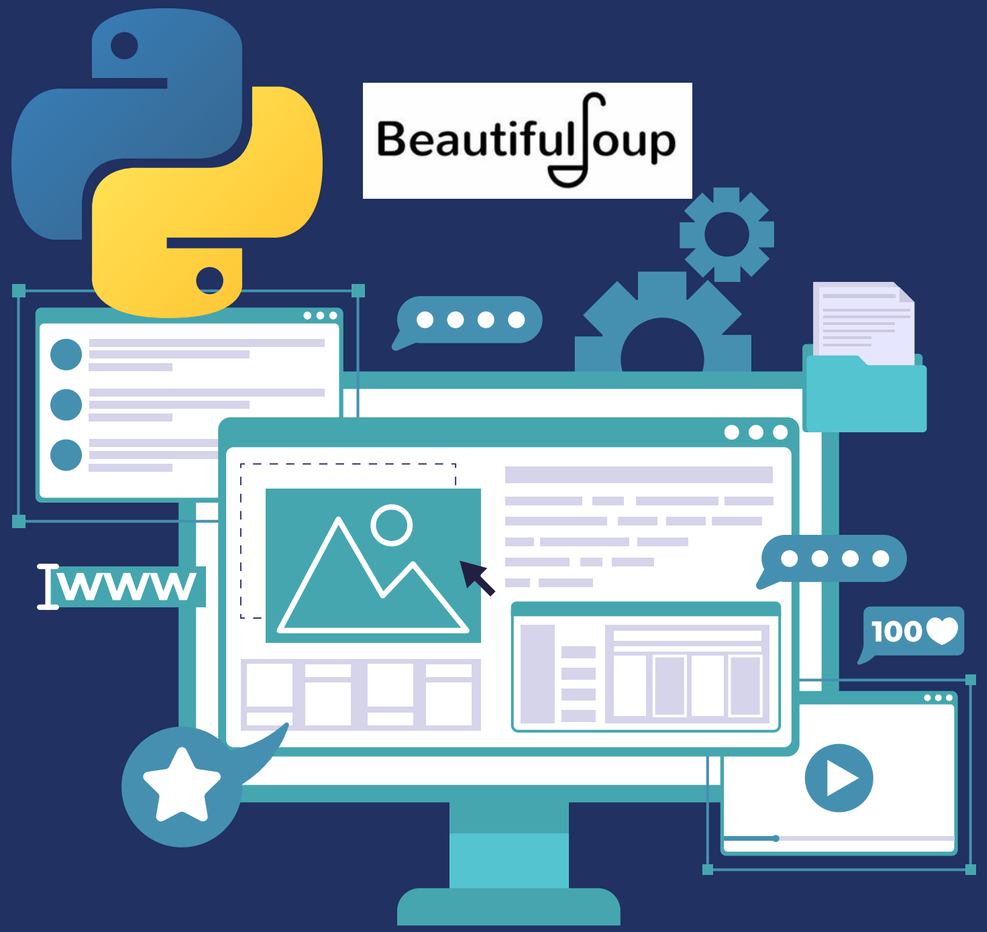
Python External Library: BeautifulSoup

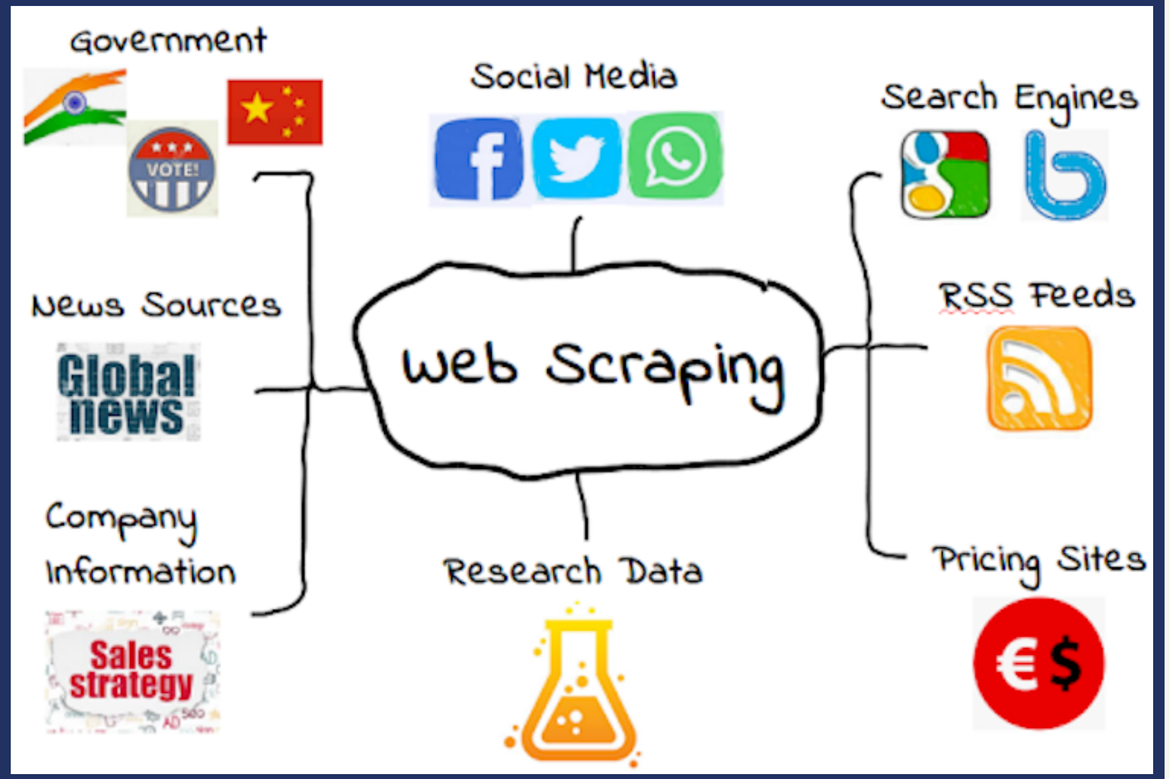
Data and Web Scraping Tool



ADVANCED PROGRAMMING FOR BUSINESS ANALYTICS

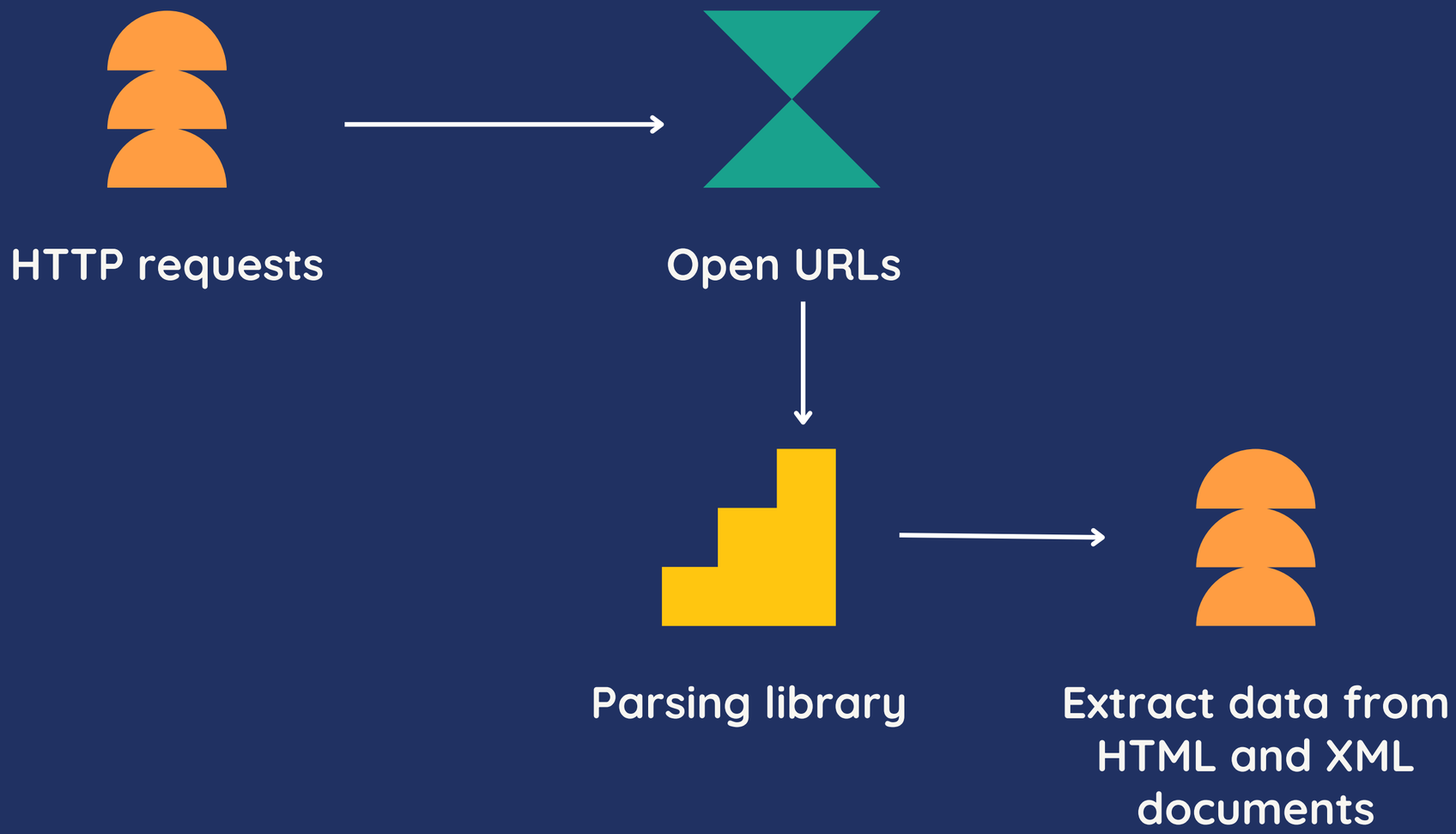
BY: DEEPANSHI BANSAL

Web Scraping



What is Web Scraping?	**Web scraping or crawling is the process of fetching data from a third-party website by downloading and parsing the HTML/XML code.**
Why Scrape Data?	• **Open Data Collection from obsolete or expired websites** • **Open Data Access in the absence of an API (Application Programming Interface)** • **Automated real-time data harvesting** • **Data Aggregation** • **Data Monitoring**

Process of Web Scraping



Introduction

BeautifulSoup

- BeautifulSoup is a python library for web scrapping and pulling data out of HTML and XML files.
- It works with parsers to provide idiomatic ways of navigating, searching, and modifying the parse tree.
- It commonly saves programmers hours or days of work.
- Installing BeautifulSoup -

  $pip3 install BeautifulSoup4

Parser Libraries Needed

html.parser

- decent speed
- does not work so well in older versions of Python

built-in parser

BeautifulSoup(markup, "html.parser")

lxml

- very fast
- can be used to quickly parse given HTML

$ pip3 install lxml

BeautifulSoup(markup, "lxml")
BeautifulSoup(markup, "lxml-xml")
BeautifulSoup(markup, "xml")

html5lib

- very slow
- extremely lenient

$ pip3 install html5lib

BeautifulSoup(markup, "html5lib")

Objects

Tags

refers to an actual XML or HTML tag in the document

- access different attributes like the class and id of a tag using tag['class'] and tag['id']
- access the whole dictionary of attributes using tag.attrs

NavigableString

refers to fetching text within a tag

- methods like replace_with("string") to replace the text within a tag.
- convert to unicode string using unicode().

BeautifulSoup

used to represent the document as a whole

Two arguments.

- The first argument is the actual markup
- Second argument is the parser that we want to use

Comment

to access the comments in a webpage

Objects - Code

BeautifulSoup

Tags

NavigableString

```
#tag
soup = BeautifulSoup('<b id ="weather" class="abc">Today is a good weather</b>', 'html.parser')
tag = soup.b
type(tag)
tag.name
tag['id']
tag.attrs
tag.attrs.keys()

#string
tag.string
#tag.string.replace_with("No longer a good day")
tag
```

```
<b class="abc" id="weather">Today is a good weather</b>
```

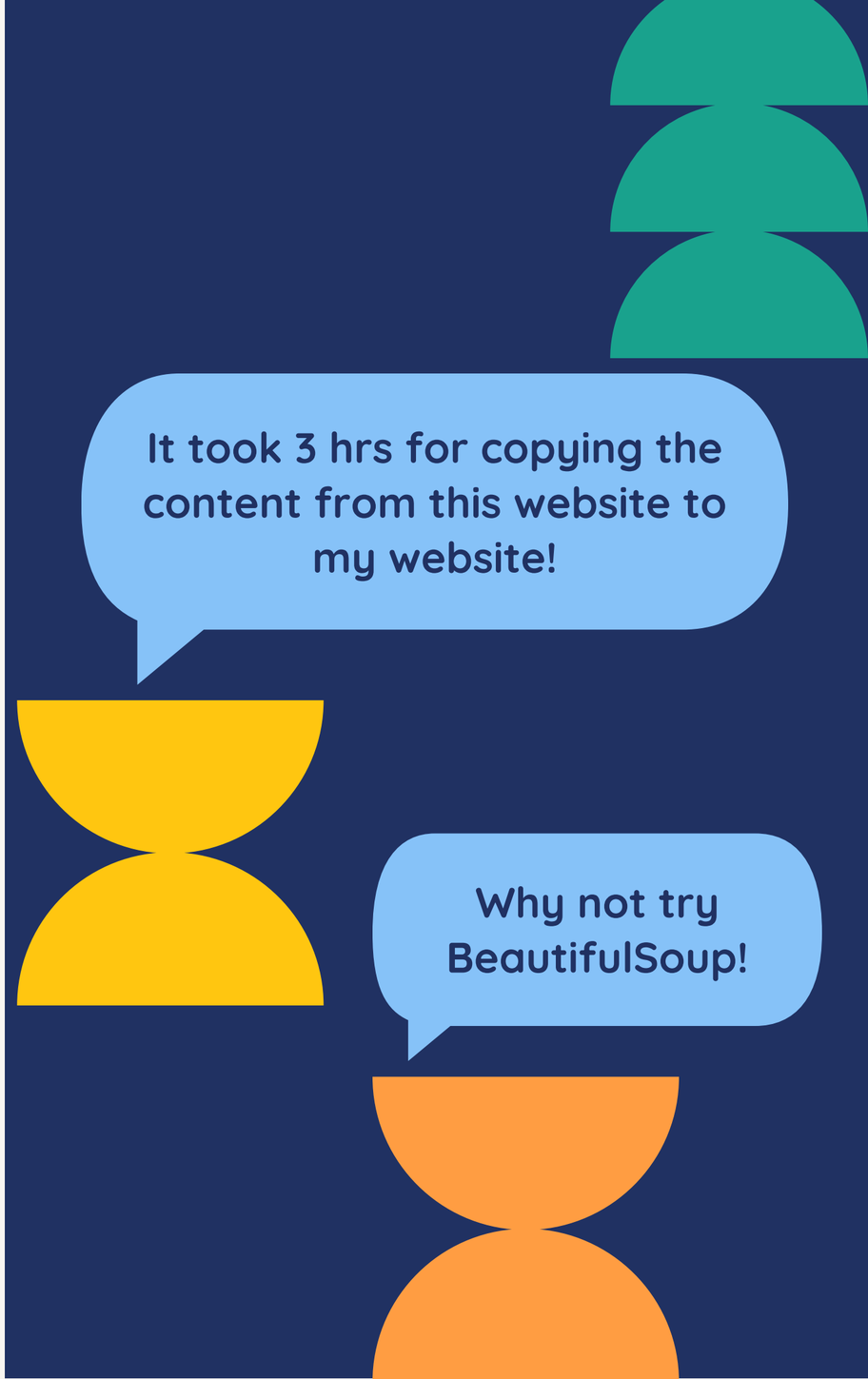
Comment

```
#comment
markup = "<b><!--this is a comment - explains beautifulsoup comment object--></b>"
soup = BeautifulSoup(markup, 'html.parser')
comment = soup.b.string
type(comment)
print(comment)
```

```
this is a comment - explains beautifulsoup comment object
```

Some Use Cases

- Bots to scrap any kind of datasets and use it further for processing
- Comparing the car price before buying
- Tracking jobs
- Tracking Apartment/House prices
- Scrap my documents on the website of universities etc.
- Getting news about economy/sports etc.
- Social media research



Code

First Example

```
from bs4 import BeautifulSoup
import urllib.request

# Fetching the content from the Wikipedia URL
get_data = urllib.request.urlopen('https://en.wikipedia.org/wiki/Entrepreneurship')
read_page = get_data.read()
#Parse the text data

#Parsing the URL content and storing the page text
parse_page = BeautifulSoup(read_page,'html.parser')

#Returning all the <p> tags
paragraphs = parse_page.find_all('p')
page_content = ''

#Looping each paragraphs and add them to the variable
for p in paragraphs:
    page_content += p.text

print(p.prettify())

#Looping each paragraph and add them to the variable after prettifying

for p in paragraphs:
    page_content += p.text
    print(p.get_text())
```

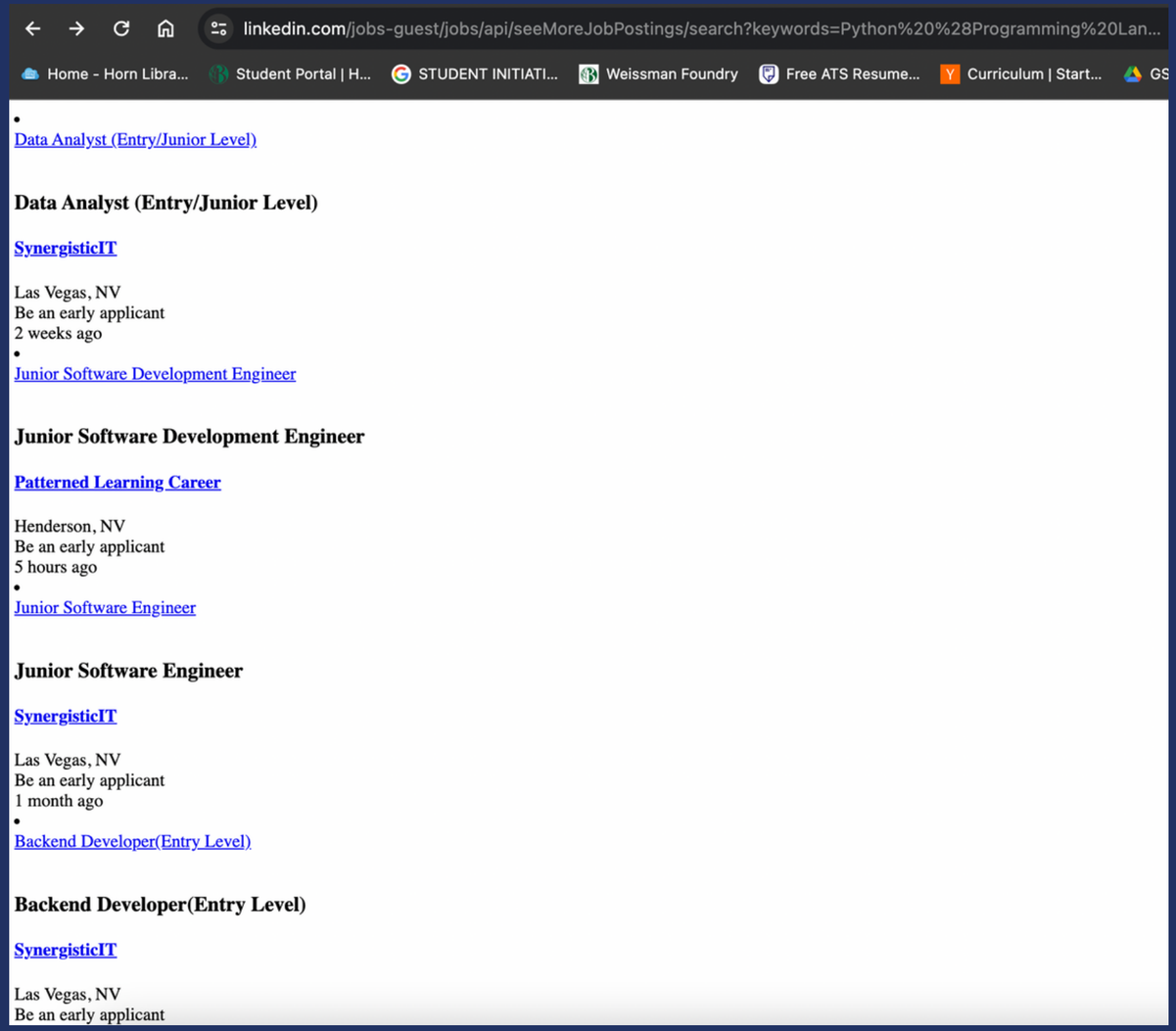
Output

```
Entrepreneurship is the creation or extraction of economic value in ways that generally entail beyond the minimal a
mount of risk (assumed by a traditional business), and potentially involving values besides simply economic ones.

An entrepreneur is an individual who creates and/or invests in one or more businesses, bearing most of the risks an
d enjoying most of the rewards.[1] The process of setting up a business is known as "entrepreneurship". The entrepr
eneur is commonly seen as an innovator, a source of new ideas, goods, services, and business/or procedures.

More narrow definitions have described entrepreneurship as the process of designing, launching and running a new bu
siness,  often similar to a small business, or (per Business Dictionary) as the "capacity and willingness to develo
p, organize and manage a business venture along with any of its risks to make a profit".[2] The people who create t
hese businesses are often referred to as "entrepreneurs".[3][4] While definitions of entrepreneurship typically foc
us on the launch and operation of businesses, due to the high risks involved in launching a startup company, a sign
ificant proportion of startups have to close (in Mikal Belicove's words) due to "lack of funding, bad business deci
sions, government policies, an economic crisis, a lack of market demand, or a combination of all of these."[5]

In the field of economics, the term entrepreneur is used for an entity which has the ability to translate invention
s or technologies into products and services.[6] In this sense, entrepreneurship describes activities on the part o
f both established firms and new businesses
```

Second Example

Linkedin Job Search

API
HTML
URL



Second Example

Linkedin Job Search

Code

Importing necessary libraries

```
import requests
from bs4 import BeautifulSoup
import math
import pandas as pd
import time

l=[]
o={}
k=[]
headers={"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0
target_url='https://www.linkedin.com/jobs-guest/jobs/api/seeMoreJobPostings/search?keywords=Python%20%28Programming%
```

fetching number of jobs in each page and storing the job ID in list l

```
for i in range(0,math.ceil(117/25)):
    res = requests.get(target_url.format(i))
    soup=BeautifulSoup(res.text,'html.parser')
    alljobs_on_this_page=soup.find_all("li")
    print(len(alljobs_on_this_page))
    for x in range(0,len(alljobs_on_this_page)):
        jobid = alljobs_on_this_page[x].find("div",{"class":"base-card"}).get('data-entity-urn').split(":")[3]
        l.append(jobid)
```

```
0
0
10
10
10
```

Second Example

Linkedin Job Search

Code

made a generic URL to keep job ID dynamic

```
target_url='https://www.linkedin.com/jobs-guest/jobs/api/jobPosting/{}'
```

saved company name , job title and job level in dictionary ‘o’ and adding the dictionary to list ‘k’

```
for j in range(0,len(l)):
    resp = requests.get(target_url.format(l[j]))
    soup=BeautifulSoup(resp.text,'html.parser')
    try:
        o["company"]=soup.find("div",{"class":"top-card-layout__card"}).find("a").find("img").get('alt')
    except:
        o["company"]=None
    try:
        o["job-title"]=soup.find("div",{"class":"top-card-layout__entity-info"}).find("a").text.strip()
    except:
        o["job-title"]=None
    try:
        o["level"]=soup.find("ul",{"class":"description__job-criteria-list"}).find("li").text.replace("Seniority lev
    except:
        o["level"]=None
    k.append(o)
    o={}
```

Second Example

Linkedin Job Search

Code - final step

saved the list of job details in csv file

```
df = pd.DataFrame(k)
df.to_csv('linkedinjobs.csv', index=False, encoding='utf-8')
print(k)
```

```
[{'company': 'SynergisticIT', 'job-title': 'Junior Software Engineer', 'level': 'Entry level'}, {'company': None,
'job-title': None, 'level': None}, {'company': 'SynergisticIT', 'job-title': 'Backend Developer(Entry Level)', 'le
el': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Data Scientist(Rermote) - Junior/Entry Level', 'le
el': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Entry Level Programmer/Coder/Developer/Data Scient
st/Analyst/Engineer', 'level': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Jr software engineer', '
evel': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Data Analyst/Python Programmer - Remote', 'leve
l': 'Entry level'}, {'company': 'Veteran Benefits Guide', 'job-title': 'AI Software Engineer', 'level': 'Entry lev
l'}, {'company': 'The Boring Company', 'job-title': 'Automation Control Software Engineer', 'level': 'Not Applicab
e'}, {'company': 'Patterned Learning Career', 'job-title': 'Full Stack Web Developer', 'level': 'Entry level'}, {'
ompany': 'Patterned Learning Career', 'job-title': 'Junior Software Development Engineer', 'level': 'Entry level'}
{'company': None, 'job-title': None, 'level': None}, {'company': 'SynergisticIT', 'job-title': 'Data Scientist(Rer
ote) - Junior/Entry Level', 'level': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Entry Level Progra
mer/Coder/Developer/Data Scientist/Analyst/Engineer', 'level': 'Entry level'}, {'company': 'Allegiant', 'job-titl
e': 'Business Intelligence Analyst', 'level': 'Associate'}, {'company': 'SynergisticIT', 'job-title': 'Jr software
engineer', 'level': 'Entry level'}, {'company': 'SynergisticIT', 'job-title': 'Data Analyst/Python Programmer - Re
ote', 'level': 'Entry level'}, {'company': 'Veteran Benefits Guide', 'job-title': 'AI Software Engineer', 'level':
'Entry level'}, {'company': 'The Boring Company', 'job-title': 'Automation Control Software Engineer', 'level': 'N
t Applicable'}, {'company': 'Patterned Learning Career', 'job-title': 'Full Stack Web Developer', 'level': 'Entry
evel'}, {'company': 'SynergisticIT', 'job-title': 'Data Analyst (Entry/Junior Level)', 'level': 'Mid-Senior leve
l'}, {'company': 'SynergisticIT', 'job-title': 'Entry Level Software Engineer', 'level': 'Entry level'}, {'compan
y': None, 'job-title': None, 'level': None}, {'company': 'Patterned Learning Career', 'job-title': 'Junior Softwar
```

Second Example

Output

linkedinjobs

company	job-title	level
SynergisticIT	Junior Software Engineer	Entry level
SynergisticIT	Backend Developer(Entry Level)	Entry level
SynergisticIT	Data Scientist(Rermote) - Junior/Entry Level	Entry level
SynergisticIT	Entry Level Programmer/Coder/Developer/Data Scientist/Analyst/Engineer	Entry level
SynergisticIT	Jr software engineer	Entry level
SynergisticIT	Data Analyst/Python Programmer - Remote	Entry level
Veteran Benefits Guide	AI Software Engineer	Entry level
The Boring Company	Automation Control Software Engineer	Not Applicable
Patterned Learning Career	Full Stack Web Developer	Entry level
Patterned Learning Career	Junior Software Development Engineer	Entry level
SynergisticIT	Data Scientist(Rermote) - Junior/Entry Level	Entry level
SynergisticIT	Entry Level Programmer/Coder/Developer/Data Scientist/Analyst/Engineer	Entry level
Allegiant	Business Intelligence Analyst	Associate
SynergisticIT	Jr software engineer	Entry level
SynergisticIT	Data Analyst/Python Programmer - Remote	Entry level
Veteran Benefits Guide	AI Software Engineer	Entry level
The Boring Company	Automation Control Software Engineer	Not Applicable
Patterned Learning Career	Full Stack Web Developer	Entry level
SynergisticIT	Data Analyst (Entry/Junior Level)	Mid-Senior level
SynergisticIT	Entry Level Software Engineer	Entry level
Patterned Learning Career	Junior Software Development Engineer	Entry level

Third Example

Amazon Search Code

Defined URL from Amazon, searching for speakers.

```
# URL to scrape
URL = 'https://www.amazon.com/s?k=speakers'
#URL = 'https://www.amazon.com/s?k=earphones'
#URL = 'https://www.amazon.com/s?k=charger'

# Header details obtained from the url inspect file
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3
    "Accept-Encoding":"gzip, deflate",
    "Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "DNT":"1",
    "Connection":"close",
    "Upgrade-Insecure-Requests":"1"
}
# Send a request to the website
page = requests.get(URL, headers=headers)
```

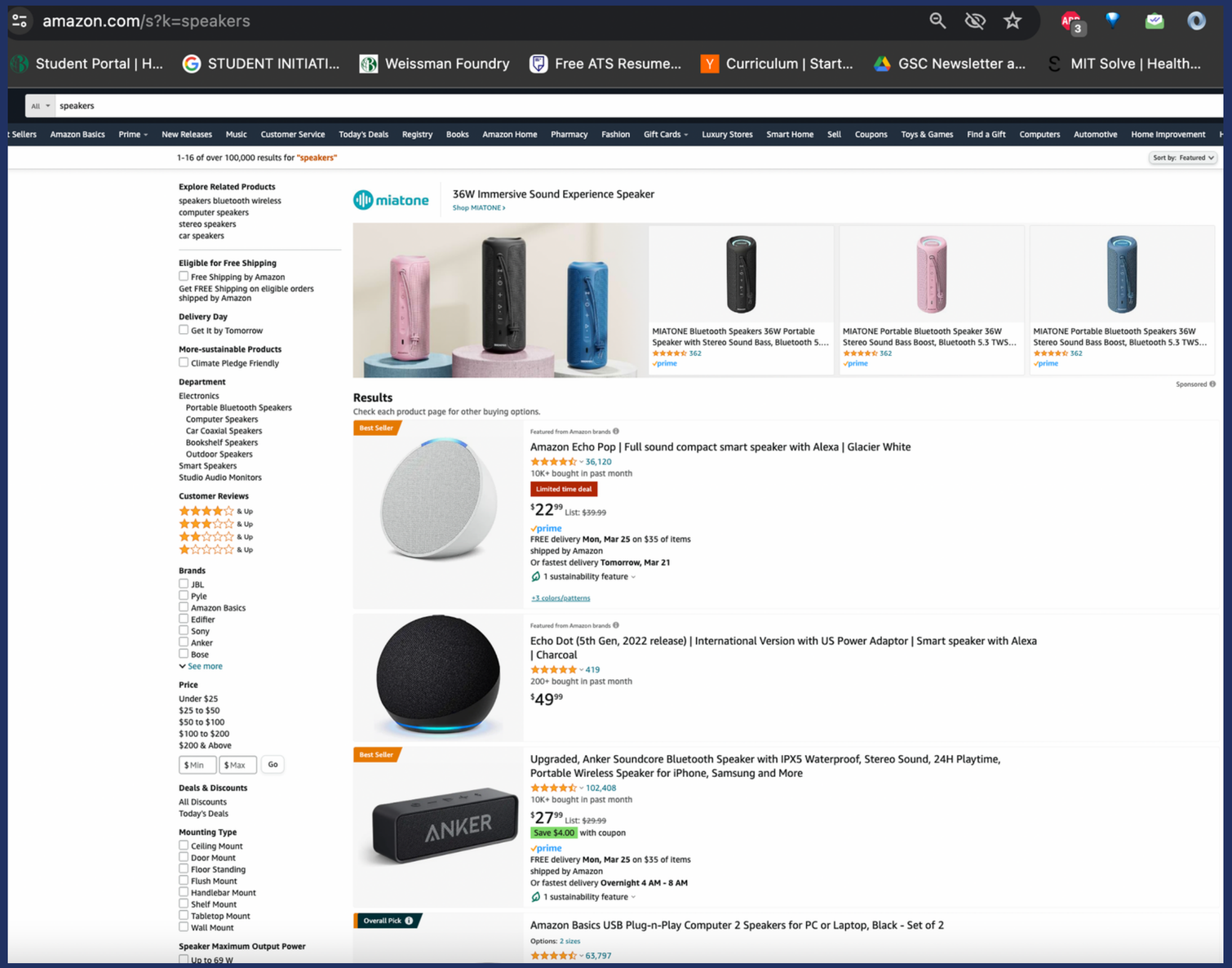
Defined BeautifulSoup object and fetched all the product details visible on Amazon for speakers

```
# Initiate BeautifulSoup
soup = BeautifulSoup(page.content, 'html.parser')

# Initializing list
data = []

# Using soup object to extract information of each product contained in a div tag
products = soup.find_all('div', class_='s-result-item')
product
```

Third Example

Amazon Search

Amazon Search URL



Third Example

Amazon Search Code

Extract all the product information from the div tag and append it as dictionary

```
# Parsing through all span values in products

# Extracting text from span tag for relevant information and setting 'N/A' if values not found

for product in products:
    name = product.find('span', class_='a-size-medium a-color-base a-text-normal').get_text(strip=True) if produc
    rating = product.find('span', class_='a-icon-alt').get_text(strip=True) if product.find('span', class_='a-ico
    customer_count = product.find('span', class_='a-size-base s-underline-text').get_text(strip=True) if product.
    price = product.find('span', class_='a-offscreen').get_text(strip=True) if product.find('span', class_='a-off
    delivery = product.find('span', class_='a-color-base a-text-bold').get_text(strip=True) if product.find('span
    # Only adding rows where product name is available
    if name != 'N/A':
        data.append({
            'Product': name,
            'Rating': rating,
            'Customer Count': customer_count,
            'Price': price,
            'Delivery': delivery
        })
```

name = product.find('span', class_='a-size-medium a-color-base a-text-normal').get_text(strip=True)
if product.find('span', class_='a-size-medium a-color-base a-text-normal') else 'N/A'

if-else to keep the length of list as same and matching right data with right product

Third Example

Amazon Search Output

COnverting list to DataFrame and showing the output of scraping

```
# Convert dictionary into DataFrame for a presentable table
df = pd.DataFrame(data)
df
```

	Product	Rating	Customer Count	Price	Delivery
0	Amazon Echo Pop | Full sound compact smart spe...	4.7 out of 5 stars	36,117	$22.99	Mon, Mar 25
1	Echo Dot (5th Gen, 2022 release) | Internation...	4.8 out of 5 stars	419	$49.99	N/A
2	Amazon Basics USB Plug-n-Play Computer 2 Speak...	4.4 out of 5 stars	63,798	$14.99	Mon, Mar 25
3	Upgraded, Anker Soundcore Bluetooth Speaker wi...	4.6 out of 5 stars	102,408	$27.99	Mon, Mar 25
4	Edifier R1280T Powered Bookshelf Speakers - 2....	4.6 out of 5 stars	17,540	$99.99	Mon, Mar 25
5	Bluetooth Speakers, Wireless TWS Portable Spea...	4.4 out of 5 stars	2,896	$79.99	Mon, Mar 25
6	JBL GO2 - Waterproof Ultra-Portable Bluetooth ...	4.6 out of 5 stars	39,632	$28.84	Mon, Mar 25
7	Certified Refurbished Echo Studio - High-fidel...	4.4 out of 5 stars	3,069	$144.99	Tue, Mar 26
8	Bluetooth Speaker with HD Sound, Portable Wire...	4.5 out of 5 stars	5,597	$26.99	Mon, Mar 25
9	Amazon Basics Computer Speakers for Desktop or...	4.2 out of 5 stars	17,615	$18.32	Mon, Mar 25
10	Creative Pebble 2.0 USB-Powered Desktop Speake...	4.5 out of 5 stars	82,005	$18.99	Mon, Mar 25
11	Wireless Bluetooth Portable Speaker 15W Stereo...	4.5 out of 5 stars	1,114	$19.99	Tue, Mar 26
12	Echo Show 5 (3rd Gen, 2023 release) | Smart di...	4.5 out of 5 stars	24,876	$89.99	Mon, Mar 25
13	Vanzon Bluetooth Speakers V40 Portable Wireles...	4.5 out of 5 stars	12,475	$39.99	Mon, Mar 25
14	Sony SSCS5 3-Way 3-Driver Bookshelf Speaker Sy...	4.7 out of 5 stars	8,320	$123.00	Fri, Mar 22
15	JBL Flip 5 Waterproof Portable Wireless Blueto...	4.8 out of 5 stars	18,461	$94.95	Mon, Mar 25
16	JBL Clip 3, Blue - Waterproof, Durable & Porta...	4.7 out of 5 stars	74,101	$38.06	Mon, Mar 25
17	MIATONE Outdoor Portable Bluetooth Speakers Wi...	4.6 out of 5 stars	22,796	$39.99	Mon, Mar 25
18	LENRUE Bluetooth Speakers, Waterproof Portable...	4.4 out of 5 stars	2,039	$11.99	Mon, Mar 25
19	SOULION R50 Bluetooth Computer Speakers, 3.5mm...	4.3 out of 5 stars	1,624	$18.99	Mon, Mar 25
20	JBL Charge 4 - Waterproof Portable Bluetooth S...	4.8 out of 5 stars	53,920	$104.95	Mon, Mar 25
21	Computer Speakers, Desktop Speakers with 6 Col...	4.4 out of 5 stars	769	$26.88	Mon, Mar 25
22	Echo Studio | Our best-sounding smart speaker ...	4.5 out of 5 stars	41,074	$199.99	Mon, Mar 25
23	Bluetooth Speakers, Ortizan 40W Loud Stereo Po...	4.4 out of 5 stars	3,476	$49.99	Mon, Mar 25

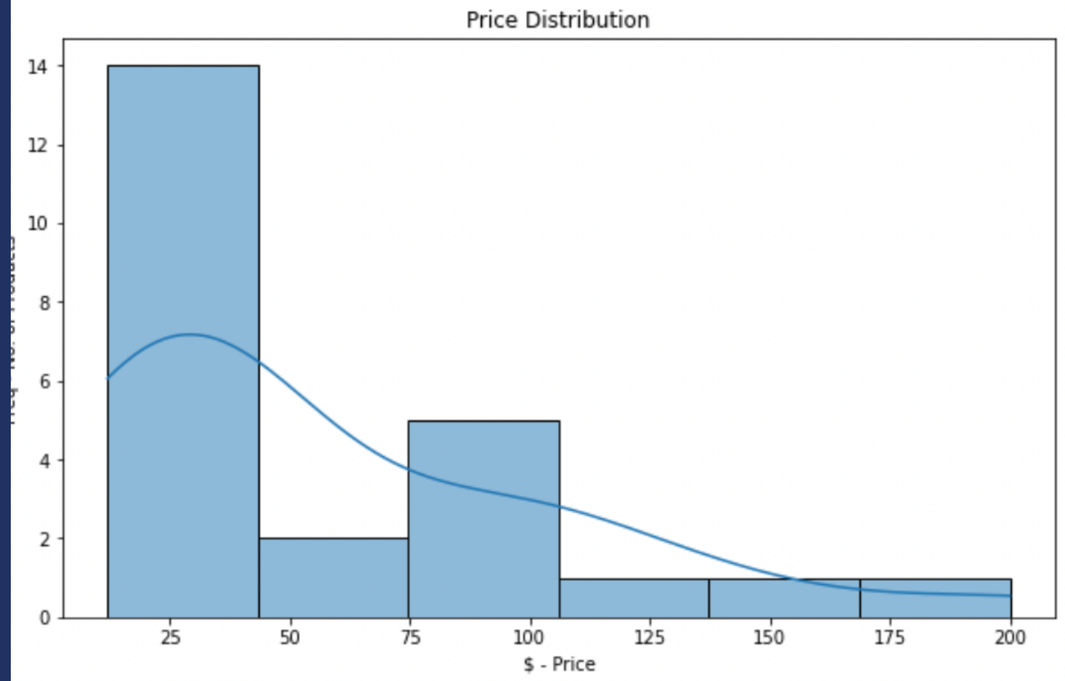
As mentioned on previous slide code, only printed entries where product name is not null

Third Example

Amazon Search

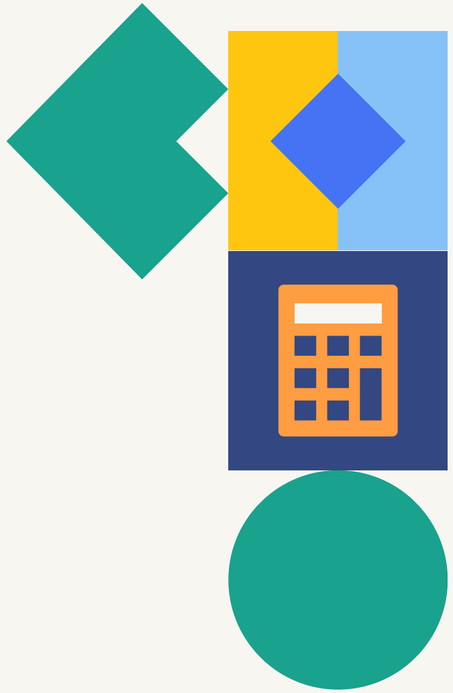
Representation

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Converting price to float value
df['Price'] = df['Price'].replace('[\$,]', '', regex=True).astype(float)

# Visualization
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], kde=True)
plt.title('Price Distribution')
plt.xlabel('$ - Price')
plt.ylabel('Freq - No. of Products')
plt.show()
```



Challenges

- For effective web scraping, script customizations, error handling, data cleaning, and results storage, are necessary steps and thus making the process time and resources intensive.
- Websites are Dynamic and always in development. The content of a website changing reflects in the tags that are selected in your script. This is especially important to consider in the case of real-time data scraping.
- Python packages and updates sometimes constitute in unstable scripts
- It is extremely important to throttle your request or cap it to a specific amount for server considerations to not overload the server usage.



Alternatives

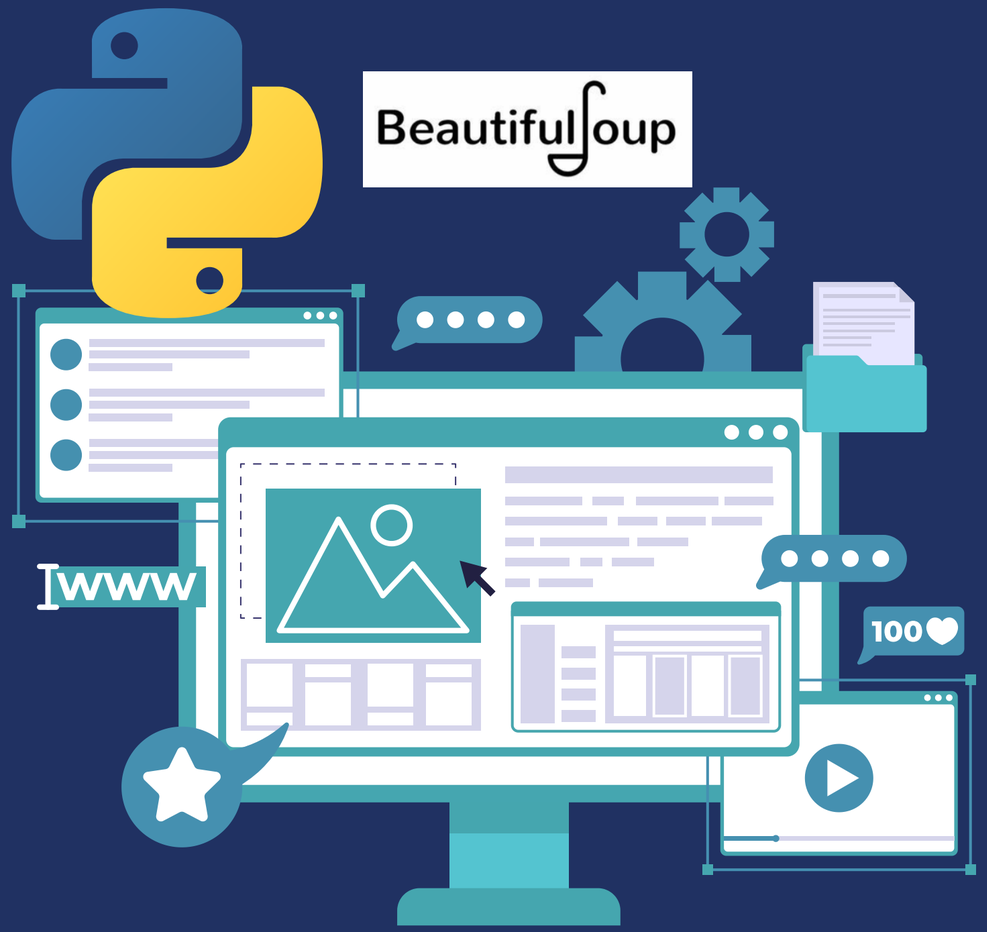
- S — ScraPy – Framework for large scale web scraping
- SE — Selenium – Browser Automation Library for Web Scraping using JavaScript
- R — rvest – R package for multi-page web scraping
- A — API (Application Programming Interface) – Direct data requests from the source
- D — DOM (Document Object Model) parsing
- P — Pattern matching text – Using regex (regular expressions) for selecting text
- M — Manually copy-pasting

BeautifulSoup vs Scapy

Features	BeautifulSoup	Scrapy
Purpose	data scraping	web scrawling and scraping
Best For	smal-medium project	for larger and complex scraping
API Scraping	Tricky	Better Support
User Authentication	Not Supported	Supported
Data Storage	Need external libraries	Provides built-in support

Thank You

For Code files refer -
https://github.com/deepanshib/WebScraping/



ADVANCED PROGRAMMING FOR BUSINESS ANALYTICS

BY: DEEPANSHI BANSAL