1.

1. Import the necessary libraries:

import requests

from bs4 import BeautifulSoup

1. Send a GET request to the Wikipedia URL and retrieve the HTML content:

url = "https://en.wikipedia.org/wiki/List\_of\_most-viewed\_YouTube\_videos"

response = requests.get(url)

content = response.content

1. Parse the HTML content using BeautifulSoup:

soup = BeautifulSoup(content, "html.parser")

1. Find the table containing the video details:

table = soup.find("table", class\_="wikitable sortable")

1. Iterate over the rows of the table and extract the desired details (Rank, Artist, Upload date, Views):

for row in table.find\_all("tr")[1:]:

cells = row.find\_all("td")

rank = cells[0].text.strip()

artist = cells[2].text.strip()

upload\_date = cells[4].text.strip()

views = cells[1].text.strip()

# Do something with the extracted details (e.g., print them)

print(f"Rank: {rank}")

print(f"Artist: {artist}")

print(f"Upload Date: {upload\_date}")

print(f"Views: {views}")

2.

import requests

from bs4 import BeautifulSoup

# Send a GET request to the international fixtures page

url = "https://www.bcci.tv/"

response = requests.get(url)

# Parse the HTML content

soup = BeautifulSoup(response.content, "html.parser")

# Find the elements containing the fixture details

fixture\_elements = soup.find\_all("div", class\_="fixture\_\_format-strip")

# Extract the details and store them in a data structure

fixtures = []

for element in fixture\_elements:

series = element.find("span", class\_="fixture\_\_format-strip--suffix").text.strip()

place = element.find("p", class\_="fixture\_\_additional-info").text.strip()

date = element.find("span", class\_="fixture\_\_datetime").text.strip()

time = element.find("span", class\_="fixture\_\_time").text.strip()

fixture = {

"Series": series,

"Place": place,

"Date": date,

"Time": time

}

fixtures.append(fixture)

# Print the extracted details

for fixture in fixtures:

print(fixture)

3.

1. Import the necessary libraries:

import requests

from bs4 import BeautifulSoup

1. Send a GET request to the website's homepage and extract the HTML content:

url = "http://statisticstimes.com/"

response = requests.get(url)

html\_content = response.content

1. Create a BeautifulSoup object to parse the HTML content:

soup = BeautifulSoup(html\_content, 'html.parser')

1. Find the link to the "Economy" page and extract its URL:

economy\_link = soup.find('a', text='Economy').get('href')

economy\_url = url + economy\_link

1. Send another GET request to the "Economy" page and extract its HTML content:

economy\_response = requests.get(economy\_url)

economy\_html\_content = economy\_response.content

1. Create another BeautifulSoup object to parse the HTML content of the "Economy" page:

economy\_soup = BeautifulSoup(economy\_html\_content, 'html.parser')

1. Find the table containing the State-wise GDP details and extract the required information:

gdp\_table = economy\_soup.find('table', {'id': 'table\_id'})

rows = gdp\_table.find\_all('tr')

for row in rows:

data = row.find\_all('td')

if len(data) > 0:

rank = data[0].text.strip()

state = data[1].text.strip()

gdp\_18\_19 = data[2].text.strip()

gdp\_19\_20 = data[3].text.strip()

share\_18\_19 = data[4].text.strip()

gdp\_billion = data[5].text.strip()

4.

1. Import the necessary libraries:

from bs4 import BeautifulSoup

import requests

1. Send a GET request to the GitHub homepage and parse the HTML content:

url = "https://github.com/"

response = requests.get(url)

soup = BeautifulSoup(response.content, "html.parser")

1. Find the link to the trending repositories page:

explore\_menu = soup.find("nav", class\_="UnderlineNav-body")

trending\_link = explore\_menu.find("a", href="/trending")

trending\_url = "https://github.com" + trending\_link["href"]

1. Send another GET request to the trending repositories page and parse the HTML content:

trending\_response = requests.get(trending\_url)

trending\_soup = BeautifulSoup(trending\_response.content, "html.parser")

1. Find the list of repositories and iterate over each repository to extract the details:

repositories = trending\_soup.find\_all("article", class\_="Box-row")

for repo in repositories:

# Repository title

title = repo.find("h1").text.strip()

# Repository description

description = repo.find("p", class\_="col-9").text.strip()

# Contributors count

contributors = repo.find("a", href=lambda href: href and "/contributors" in href)

contributors\_count = contributors.text.strip() if contributors else "0"

# Language used

language = repo.find("span", itemprop="programmingLanguage")

language\_used = language.text.strip() if language else "Not specified"

# Print or store the details as per your requirement

print("Title:", title)

print("Description:", description)

print("Contributors count:", contributors\_count)

print("Language used:", language\_used)

print()

5.

1. Import the necessary libraries:

import requests

from bs4 import BeautifulSoup

1. Send a GET request to the billboard.com homepage and retrieve the HTML content:

url = "https://www.billboard.com/"

response = requests.get(url)

html\_content = response.content

1. Parse the HTML content using BeautifulSoup:

soup = BeautifulSoup(html\_content, "html.parser")

1. Find the link to the Hot 100 page:

charts\_link = soup.find("a", text="Charts")

hot100\_link = charts\_link["href"]

1. Send another GET request to the Hot 100 page and retrieve the HTML content:

hot100\_url = url + hot100\_link

hot100\_response = requests.get(hot100\_url)

hot100\_html\_content = hot100\_response.content

1. Parse the Hot 100 HTML content using BeautifulSoup:

hot100\_soup = BeautifulSoup(hot100\_html\_content, "html.parser")

1. Find the table containing the top 100 songs:

table = hot100\_soup.find("table", class\_="chart-list")

1. Iterate over the rows of the table and extract the desired details:

for row in table.find\_all("tr"):

columns = row.find\_all("td")

if len(columns) == 5:

song\_name = columns[1].text.strip()

artist\_name = columns[2].text.strip()

last\_week\_rank = columns[3].text.strip()

peak\_rank = columns[4].text.strip()

weeks\_on\_board = columns[5].text.strip()

# Do something with the extracted details (e.g., store them in a list or write to a file)

6.

import requests

from bs4 import BeautifulSoup

# Send a GET request to the URL

url = "https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-time-fifty-shades-grey-compare"

response = requests.get(url)

# Parse the HTML content

soup = BeautifulSoup(response.content, 'html.parser')

# Find the relevant HTML elements and extract the data

novels = []

table = soup.find('table')

rows = table.find\_all('tr')[1:] # Exclude the header row

for row in rows:

columns = row.find\_all('td')

book\_name = columns[1].text.strip()

author\_name = columns[2].text.strip()

volumes\_sold = columns[3].text.strip()

publisher = columns[4].text.strip()

genre = columns[5].text.strip()

novel = {

'Book Name': book\_name,

'Author Name': author\_name,

'Volumes Sold': volumes\_sold,

'Publisher': publisher,

'Genre': genre

}

novels.append(novel)

# Print the scraped data

for novel in novels:

print(novel)

7.

import requests

from bs4 import BeautifulSoup

url = "https://www.imdb.com/list/ls095964455/"

response = requests.get(url)

soup = BeautifulSoup(response.content, "html.parser")

series\_list = soup.find\_all("div", class\_="lister-item-content")

for series in series\_list:

name = series.find("h3").find("a").text.strip()

year\_span = series.find("span", class\_="lister-item-year").text.strip("()")

genre = series.find("span", class\_="genre").text.strip()

runtime = series.find("span", class\_="runtime").text.strip()

rating = series.find("span", class\_="ipl-rating-star\_\_rating").text.strip()

votes = series.find("span", attrs={"name": "nv"}).text.strip()

print("Name:", name)

print("Year Span:", year\_span)

print("Genre:", genre)

print("Run Time:", runtime)

print("Ratings:", rating)

print("Votes:", votes)

print()

8.

import requests

from bs4 import BeautifulSoup

# Send a GET request to the UCI machine learning repositories website

url = "https://archive.ics.uci.edu/"

response = requests.get(url)

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.content, "html.parser")

# Find the link to the "Show All Dataset" page

show\_all\_link = soup.find("a", href="ml/datasets.php")

# Construct the URL for the "Show All Dataset" page

show\_all\_url = url + show\_all\_link["href"]

# Send another GET request to the "Show All Dataset" page

show\_all\_response = requests.get(show\_all\_url)

# Parse the HTML content of the "Show All Dataset" page

show\_all\_soup = BeautifulSoup(show\_all\_response.content, "html.parser")

# Find the table containing the dataset details

dataset\_table = show\_all\_soup.find("table", class\_="table")

# Extract the details from the table rows

dataset\_details = []

for row in dataset\_table.find\_all("tr")[1:]:

columns = row.find\_all("td")

dataset\_name = columns[0].text.strip()

data\_type = columns[1].text.strip()

task = columns[2].text.strip()

attribute\_type = columns[3].text.strip()

num\_instances = columns[4].text.strip()

num\_attributes = columns[5].text.strip()

year = columns[6].text.strip()

dataset\_details.append((dataset\_name, data\_type, task, attribute\_type, num\_instances, num\_attributes, year))

# Print the dataset details

for dataset in dataset\_details:

print("Dataset Name:", dataset[0])

print("Data Type:", dataset[1])

print("Task:", dataset[2])

print("Attribute Type:", dataset[3])

print("No of Instances:", dataset[4])

print("No of Attributes:", dataset[5])

print("Year:", dataset[6])

print()