In [2]:
```
import requests
```

In [3]:
```
pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in e:\data science files\lib
\site-packages (4.12.2)
Requirement already satisfied: soupsieve>1.2 in e:\data science files\lib
\site-packages (from beautifulsoup4) (2.4)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
from bs4 import BeautifulSoup
import pandas as pd
```

In [18]:
```
def scrape_imdb_top_100_indian_movies(url):
        response = requests.get(url)
if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        movies_data = [https://www.imdb.com/list/ls056092300/]

        for movie_tag in soup.find_all('div', class_='lister-item-content'):
            name = movie_tag.find('a').text.strip()
            rating = movie_tag.find('span', class_='ipl-rating-star__rating'
            year = movie_tag.find('span', class_='lister-item-year').text.st

            movies_data.append({
                'Name': name,
                'Rating': rating,
                'Year of Release': year
            })

        return movies_data

else:
        print(f"Failed to fetch data from {https://www.imdb.com/list/ls05609
        return None
```

```
  Cell In[18], line 5
    movies_data = [https://www.imdb.com/list/ls056092300/]
                          ^
SyntaxError: invalid syntax
```

In [20]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_imdb_top_100_indian_movies(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        movies_data = []

        for movie_tag in soup.find_all('div', class_='lister-item-content'):
            name = movie_tag.find('a').text.strip()
            rating = movie_tag.find('span', class_='ipl-rating-star__rating'
            year = movie_tag.find('span', class_='lister-item-year').text.st

            movies_data.append({
                'Name': name,
                'Rating': rating,
                'Year of Release': year
            })

        return movies_data
    else:
        print(f"Failed to fetch data from {url}")
        return None

url = 'https://www.imdb.com/list/ls056092300/'

movies_data = scrape_imdb_top_100_indian_movies(url)

if movies_data:
    df = pd.DataFrame(movies_data)
    print(df)
else:
    print("Data scraping unsuccessful.")
```

```
                                Name Rating Year of Release
0                    Ship of Theseus      8            2012
1                             Iruvar    8.4            1997
2                    Kaagaz Ke Phool    7.8            1959
3   Lagaan: Once Upon a Time in India    8.1            2001
4                    Pather Panchali    8.2            1955
..                               ...    ...             ...
95                       Apur Sansar    8.4            1959
96                       Kanchivaram    8.2            2008
97                   Monsoon Wedding    7.3            2001
98                             Black    8.1            2005
99                           Deewaar      8            1975

[100 rows x 3 columns]
```

In [74]:
```
# Write a python program to scrape product name, price and discounts from ht
def scrape_peachmode_products(url):
    # Send a GET request to the URL
    response = requests.get(url)

        if response.

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

            products = []
        for product_card in soup.find_all('div', class_='product-grid-item')
            product_name = product_card.find('h4', class_='product-title').t
            product_price = product_card.find('span', class_='product-price'
            product_discount = product_card.find('span', class_='product-dis

            products.append({
                'name': product_name,
                'price': product_price,
                'discount': product_discount,
            })

        return products

    else:

        print(f"Error: Unable to fetch data. Status code: {response.status_c
        return None

if __name__ == "__main__":

            print(f"Name: {product['name']}")
            print(f"Price: {product['price']}")
            print(f"Discount: {product['discount']}")
    else:
        print("No data to display.")
```

```
  File <tokenize>:34
    else:
    ^
IndentationError: unindent does not match any outer indentation level
```

In [78]:
```
# Write a python program to scrape product name, price and discounts from ht
def scrape_peachmode_products(url):
    # Send a GET request to the URL
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        products = []
        for card in soup.find_all('div', class_='grid'):
            name = product_card.find('h4', class_='title').text.strip()
            price = product_card.find('span', class_='price').text.strip()
            discount = product_card.find('span', class_='discount').text.str

            products.append({
                'name': p_name,
                'price': p_price,
                'discount': p_discount,
            })

        return products

    else:

        print(f"Error: Unable to fetch data. Status code: {response.status_c
        return None

if __name__ == "__main__":

    target_url = "https://peachmode.com/search?q=bags"

    scraped_data = scrape_peachmode_products(target_url)

    if scraped_data:
        for index, product in enumerate(scraped_data, start=1):
            print(f"\nProduct {index}:")
            print(f"Name: {product['name']}")
            print(f"Price: {product['price']}")
            print(f"Discount: {product['discount']}")
    else:
        print("No data to display.")
```

```
No data to display.
```

In [4]:
```
#Write a python program to scrape cricket rankings from icc-cricket.com. You

def scrape_odi_teams():

    teams_data = []

    for team in soup.find_all('tr', class_='rankings-block__banner')[:1]:
        name = team.find('span', class_='u-hide-phablet').text.strip()
        match = team.find('td', class_='rankings-block__banner--matches').te
        point = team.find('td', class_='rankings-block__banner--points').tex
        rating = team.find('td', class_='rankings-block__banner--rating').te

        teams_data.append({'Team': name, 'Matches': match, 'Points': point,

    for team in soup.find_all('tr', class_='table-body')[:9]:
        name = team.find('span', class_='u-hide-phablet').text.strip()
        match = team.find_all('td')[2].text.strip()
        point = team.find_all('td')[3].text.strip()
        rating = team.find_all('td')[4].text.strip()

        teams_data.append({'Team': name, 'Matches': match, 'Points': point,

    df = pd.DataFrame(teams_data)
    print("top ten ODI teams in men's cricket:")
    print df
```

In [4]:
```
pip install youtube_dl
import requests
from bs4 import BeautifulSoup
import pandas as pd
def scrape_odi_teams():
    url = "https://www.icc-cricket.com/rankings/team-rankings/mens/odi"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    teams_data = []

    for team in soup.find_all('tr', class_='rankings-block__banner')[:1]:
        name = team.find('span', class_='u-hide-phablet').text.strip()
        match = team.find('td', class_='rankings-block__banner--matches').te
        point = team.find('td', class_='rankings-block__banner--points').tex
        rating = team.find('td', class_='rankings-block__banner--rating').te

        teams_data.append({'Team': name, 'Matches': match, 'Points': point,

    for team in soup.find_all('tr', class_='table-body')[:9]:
        name = team.find('span', class_='u-hide-phablet').text.strip()
        match = team.find_all('td')[2].text.strip()
        point = team.find_all('td')[3].text.strip()
        rating = team.find_all('td')[4].text.strip()

        teams_data.append({'Team': name, 'Matches': match, 'Points': point,

    df = pd.DataFrame(teams_data)
    print("Top ten ODI teams in men's cricket:")
    print(df)

if __name__ == "__main__":
    scrape_odi_teams()
```

```
  Cell In[4], line 1
    pip install youtube_dl
        ^
SyntaxError: invalid syntax
```

In [ ]:

```
#Write a python program to scrape details of all the posts from https://www.

import requests
from bs4 import BeautifulSoup
import youtube_dl

def scrape_patreon_post(url):
    # Fetch HTML content
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Extract post details
    heading = soup.find('h1', class_='post-meta-title').text.strip()
    date = soup.find('time', class_='post-meta-published').text.strip()
    content = soup.find('div', class_='post-content').text.strip()

    # Extract YouTube video link
    youtube_link = soup.find('a', class_='oembed-link')['href']

    # Get YouTube video details using youtube_dl
    video_info = get_youtube_video_info(youtube_link)

    # Extract likes from video details
    likes = video_info.get('like_count', 'N/A')

    # Print the details
    print(f"Heading: {heading}")
    print(f"Date: {date}")
    print(f"Content: {content}")
    print(f"Likes: {likes}")

def get_youtube_video_info(youtube_link):
    ydl_opts = {
        'quiet': True,
        'extract_flat': True,
    }

    with youtube_dl.YoutubeDL(ydl_opts) as ydl:
        info_dict = ydl.extract_info(youtube_link, download=False)
        return info_dict

if __name__ == "__main__":
    patreon_url = "https://www.patreon.com/coreyms"
    scrape_patreon_post(patreon_url)
```

In [2]:
```
#Write a python program to scrape details of all the posts from https://www.

import requests
from bs4 import BeautifulSoup

def scrape_house_details(locality):
    url = f"https://www.nobroker.in/property/sale/{locality}/mumbai"
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        houses = soup.find_all('div', class_='card')

        for house in houses:
            title = house.find('h2', class_='heading-6').text.strip()
            location = house.find('div', class_='nb__2CMjv').text.strip()
            area = house.find('div', class_='nb__3oNyC').text.strip()
            emi = house.find('div', class_='font-semi-bold heading-6').text.
            price = house.find('div', class_='heading-7').text.strip()

            print(f"Title: {title}")
            print(f"Location: {location}")
            print(f"Area: {area}")
            print(f"EMI: {emi}")
            print(f"Price: {price}")
            print("-" * 50)

    else:
        print(f"Failed to retrieve data for {locality}. Status code: {respor

if __name__ == "__main__":
    localities = ["indiranagar", "jayanagar", "rajajinagar"]

    for locality in localities:
        print(f"Scraping house details for {locality}")
        scrape_house_details(locality)
```

```
Scraping house details for indiranagar
Scraping house details for jayanagar
Scraping house details for rajajinagar
```

In [4]:
```
# Write a python program to scrape first 10 product details which include pr

import requests
from bs4 import BeautifulSoup

def scrape_product_details(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        products = soup.find_all('div', class_='productCardBox')

        for i, product in enumerate(products[:10], start=1):
            product_name_tag = product.find('h3', class_='product-title')
            price_tag = product.find('span', class_='discounted-price')
            image_tag = product.find('img')

            # Check if elements exist before accessing attributes
            if product_name_tag and price_tag and image_tag:
                product_name = product_name_tag.text.strip()
                price = price_tag.text.strip()
                image_url = image_tag['src']

                print(f"Product {i}:")
                print(f"Name: {product_name}")
                print(f"Price: {price}")
                print(f"Image URL: {image_url}")
                print("-" * 50)
            else:
                print(f"Skipping product {i} due to missing details.")
    else:
        print(f"Failed to retrieve data. Status code: {response.status_code}

if __name__ == "__main__":
    url = "https://www.bewakoof.com/bestseller?sort=popular"
    scrape_product_details(url)
```

```
Skipping product 1 due to missing details.
Skipping product 2 due to missing details.
Skipping product 3 due to missing details.
Skipping product 4 due to missing details.
Skipping product 5 due to missing details.
Skipping product 6 due to missing details.
Skipping product 7 due to missing details.
Skipping product 8 due to missing details.
Skipping product 9 due to missing details.
Skipping product 10 due to missing details.
```

In [3]:
```
#Q1-Please visit https://www.cnbc.com/world/?region=world and scrap-

import requests
from bs4 import BeautifulSoup

def scrape_cnbc_world_news(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        # Scraping headings, dates, and news links
        headings = [heading.text.strip() for heading in soup.find_all('h3',
        dates = [date.text.strip() for date in soup.find_all('div', class_='
        news_links = [link['href'] for link in soup.select('.Card-title a[hr

        # Print the scraped data
        for i, (heading, date, link) in enumerate(zip(headings, dates, news_
            print(f"News {i}:")
            print(f"Heading: {heading}")
            print(f"Date: {date}")
            print(f"News Link: {link}")
            print("-" * 50)

    else:
        print(f"Failed to retrieve data. Status code: {response.status_code}

if __name__ == "__main__":
    url = "https://www.cnbc.com/world/?region=world"
    scrape_cnbc_world_news(url)
```

In [1]:
```
#Q2-Please visit https://www.keaipublishing.com/en/journals/artificial-intel

import requests
from bs4 import BeautifulSoup

def scrape_most_downloaded_articles(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        # Scraping paper titles, dates, and authors
        titles = [title.text.strip() for title in soup.find_all('h5', class_
        dates = [date.text.strip() for date in soup.find_all('span', class_=
        authors = [author.text.strip() for author in soup.find_all('span', c

        # Print the scraped data
        for i, (title, date, author) in enumerate(zip(titles, dates, authors
            print(f"Article {i}:")
            print(f"Paper Title: {title}")
            print(f"Date: {date}")
            print(f"Author: {author}")
            print("-" * 50)

    else:
        print(f"Failed to retrieve data. Status code: {response.status_code}

if __name__ == "__main__":
    url = "https://www.keaipublishing.com/en/journals/artificial-intelligenc
    scrape_most_downloaded_articles(url)
```

In [2]:
```
import requests

url = "http://localhost:8888/notebooks/Assignment-March%2024-Copy1.ipynb"
response = requests.get(url)

with open("downloaded_file.txt", "wb") as f:
    f.write(response.content)
```

In [ ]:
```
```