In [ ]:

```

```

In [2]:

```
pip install requests beautifulsoup4 pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (2.31.0)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (4.12.2)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from r
equests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests) (3.
4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from request
s) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from request
s) (2023.7.22)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsou
p4) (2.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\programdata\anaconda3\lib\site-packages (from pan
das) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas) (202
3.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\programdata\anaconda3\lib\site-packages (from pandas) (20
23.3)
Requirement already satisfied: numpy>=1.21.0 in c:\programdata\anaconda3\lib\site-packages (from pandas) (1.2
4.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=
2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch HTML content from a URL
def get_html_content(url):
    response = requests.get(url)
    return response.text

# Function to extract header tags from HTML content
def extract_headers(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    headers = soup.find_all(['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])
    return [header.text.strip() for header in headers]

# Function to create DataFrame from a list of headers
def create_dataframe(headers):
    df = pd.DataFrame({'Headers': headers})
    return df

# Wikipedia URL
wikipedia_url = 'https://en.wikipedia.org/wiki/Main_Page'

# Fetch HTML content
html_content = get_html_content(wikipedia_url)

# Extract header tags
header_tags = extract_headers(html_content)

# Create DataFrame
df = create_dataframe(header_tags)

# Display DataFrame
print(df)
```

```
                          Headers
0                       Main Page
1            Welcome to Wikipedia
2    From today's featured article
3                   Did you know ...
4                     In the news
5                     On this day
6       From today's featured list
7         Today's featured picture
8         Other areas of Wikipedia
9      Wikipedia's sister projects
10            Wikipedia languages
```

2) Write s python program to display list of respected former presidents of India(i.e. Name , Term ofoffice) from https://presidentofindia.nic.in/former-presidents.htm and make data frame.

In [9]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch HTML content from a URL
def get_html_content(url):
    response = requests.get(url)
    return response.text

# Function to extract former presidents' information from HTML content
def extract_former_presidents_info(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the div containing the information
    president_divs = soup.find_all('div', class_='post-content')

    # Extract information for each president
    presidents_info = []
    for president_div in president_divs:
        name = president_div.find('strong').text.strip()
        term_of_office = president_div.find('p').text.strip()
        presidents_info.append({'Name': name, 'Term of Office': term_of_office})

    return presidents_info

# Function to create DataFrame from a list of presidents' information
def create_dataframe(presidents_info):
    df = pd.DataFrame(presidents_info)
    return df

# URL of the page with information about former presidents of India
presidents_url = 'https://presidentofindia.nic.in/former-presidents.htm'

# Fetch HTML content
html_content = get_html_content(presidents_url)

# Extract former presidents' information
former_presidents_info = extract_former_presidents_info(html_content)

# Create DataFrame
df = create_dataframe(former_presidents_info)

# Display DataFrame
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

6) Write a python program to scrape the details of most downloaded articles from AI in last 90 days.https://www.journals.elsevier.com/artificial-intelligence/most-downloaded-articles Scrape below mentioned details and make data framei) Paper Title ii) Authors iii) Published Date iv) Paper URL

In [11]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch HTML content from a URL
def get_html_content(url):
    response = requests.get(url)
    return response.text

# Function to extract article details from HTML content
def extract_article_details(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the container for each article
    article_containers = soup.find_all('div', class_='pod-listing')

    # Extract information for each article
    articles_info = []
    for container in article_containers:
        title = container.find('a', class_='pod-listing-header').text.strip()
        authors = container.find('div', class_='text-xs').text.strip()
        published_date = container.find('div', class_='text-xs', string='Published:').find_next('div').text.s
        paper_url = container.find('a', class_='pod-listing-header')['href'].strip()

        articles_info.append({
            'Paper Title': title,
            'Authors': authors,
            'Published Date': published_date,
            'Paper URL': paper_url
        })

    return articles_info

# Function to create DataFrame from a list of articles' information
def create_dataframe(articles_info):
    df = pd.DataFrame(articles_info)
    return df

# URL of the page with most downloaded articles
url = 'https://www.journals.elsevier.com/artificial-intelligence/most-downloaded-articles'

# Fetch HTML content
html_content = get_html_content(url)

# Extract article details
articles_info = extract_article_details(html_content)

# Create DataFrame
df = create_dataframe(a
```

```
  Cell In[11], line 49
    df = create_dataframe(a
                           ^
SyntaxError: incomplete input
```

5) Write a python program to scrape mentioned news details from https://www.cnbc.com/world/?region=world and make data framei) Headline ii) Time iii) News Link

In [12]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch HTML content from a URL
def get_html_content(url):
    response = requests.get(url)
    return response.text

# Function to extract news details from HTML content
def extract_news_details(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the container for each news article
    article_containers = soup.find_all('div', class_='Card-titleContainer')

    # Extract information for each news article
    news_info = []
    for container in article_containers:
        headline = container.find('a', class_='Card-titleLink').text.strip()
        time = container.find('time')['datetime'].strip()
        news_link = container.find('a', class_='Card-titleLink')['href'].strip()

        news_info.append({
            'Headline': headline,
            'Time': time,
            'News Link': news_link
        })

    return news_info

# Function to create DataFrame from a list of news information
def create_dataframe(news_info):
    df = pd.DataFrame(news_info)
    return df

# URL of the page with news details
url = 'https://www.cnbc.com/world/?region=world'

# Fetch HTML content
```

```
html_content = get_html_content(url)

# Extract news details
news_info = extract_news_details(html_content)

# Create DataFrame
df = create_dataframe(news_info)

# Display DataFrame
print(df)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[12], line 44
     41 html_content = get_html_content(url)
     43 # Extract news details
---> 44 news_info = extract_news_details(html_content)
     46 # Create DataFrame
     47 df = create_dataframe(news_info)

Cell In[12], line 20, in extract_news_details(html_content)
     18 news_info = []
     19 for container in article_containers:
---> 20     headline = container.find('a', class_='Card-titleLink').text.strip()
     21     time = container.find('time')['datetime'].strip()
     22     news_link = container.find('a', class_='Card-titleLink')['href'].strip()

AttributeError: 'NoneType' object has no attribute 'text'
```

3) Write a python program to scrape cricket rankings from icc-cricket.com. You have to scrape and make data framea) Top 10 ODI teams in men’s cricket along with the records for matches, points and rating. b) Top 10 ODI Batsmen along with the records of their team andrating. c) Top 10 ODI bowlers along with the records of their team andrating.

In [14]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Function to fetch HTML content from a URL
def get_html_content(url):
    response = requests.get(url)
    return response.text

# Function to extract team rankings from HTML content
def extract_team_rankings(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the table containing team rankings
    table = soup.find('table', class_='table')

    # Extract information for each team
    teams_info = []
    rows = table.find('tbody').find_all('tr')
    for row in rows[:10]:  # Extracting data for the top 10 teams
        columns = row.find_all('td')
        team_name = columns[1].text.strip()
        matches = columns[2].text.strip()
        points = columns[3].text.strip()
        rating = columns[4].text.strip()

        teams_info.append({
            'Team': team_name,
            'Matches': matches,
            'Points': points,
            'Rating': rating
        })

    return teams_info

# Function to extract player rankings from HTML content
def extract_player_rankings(html_content, category):
    soup = BeautifulSoup(html_content, 'html.parser')

    # Find the table containing player rankings
    table = soup.find('table', class_='table rankings-table')

    # Extract information for each player
    players_info = []
    rows = table.find('tbody').find_all('tr')
```

```python
    for row in rows[:10]:  # Extracting data for the top 10 players
        columns = row.find_all('td')
        player_name = columns[1].find('div', class_='rankings-block__banner--name-large')
        if player_name:
            player_name = player_name.text.strip()
        else:
            player_name = columns[1].find('a').text.strip()

        team = columns[2].find('span', class_='table-body__logo-text')
        if team:
            team = team.text.strip()
        else:
            team = columns[2].text.strip()

        rating = columns[3].text.strip()

        players_info.append({
            'Player': player_name,
            'Team': team,
            'Rating': rating
        })

    return players_info

# URL of the page with ODI team rankings
team_rankings_url = 'https://www.icc-cricket.com/rankings/team-rankings/mens/odi'

# URL of the page with ODI batsmen rankings
batsmen_rankings_url = 'https://www.icc-cricket.com/rankings/mens/player-rankings/odi/batting'

# URL of the page with ODI bowlers rankings
bowlers_rankings_url = 'https://www.icc-cricket.com/rankings/mens/player-rankings/odi/bowling'

# Fetch HTML content for team rankings
team_html_content = get_html_content(team_rankings_url)

# Extract team rankings
team_rankings_info = extract_team_rankings(team_html_content)

# Create DataFrame for team rankings
team_df = pd.DataFrame(team_rankings_info)

# Fetch HTML content for batsmen rankings
batsmen_html_content = get_html_content(batsmen_rankings_url)

# Extract batsmen rankings
batsmen_rankings_info = extract_player_rankings(batsmen_html_content, 'batting')

# Create DataFrame for batsmen rankings
batsmen_df = pd.DataFrame(batsmen_rankings_info)

# Fetch HTML content for bowlers rankings
bowlers_html_content = get_html_content(bowlers_rankings_url)

# Extract bowlers rankings
bowlers_rankings_info = extract_player_rankings(bowlers_html_content, 'bowling')

# Create DataFrame for bowlers rankings
bowlers_df = pd.DataFrame(bowlers_rankings_info)

# Display DataFrames
print("Top 10 ODI Teams:")
print(team_df)
print("\nTop 10 ODI Batsmen:")
print(batsmen_df)
print("\nTop 10 ODI Bowlers:")
print(bowlers_df)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[14], line 83
     80 team_html_content = get_html_content(team_rankings_url)
     82 # Extract team rankings
---> 83 team_rankings_info = extract_team_rankings(team_html_content)
     85 # Create DataFrame for team rankings
     86 team_df = pd.DataFrame(team_rankings_info)

Cell In[14], line 19, in extract_team_rankings(html_content)
     17 # Extract information for each team
     18 teams_info = []
---> 19 rows = table.find('tbody').find_all('tr')
     20 for row in rows[:10]:  # Extracting data for the top 10 teams
     21     columns = row.find_all('td')

AttributeError: 'NoneType' object has no attribute 'find'
```

```
In [ ]:
```