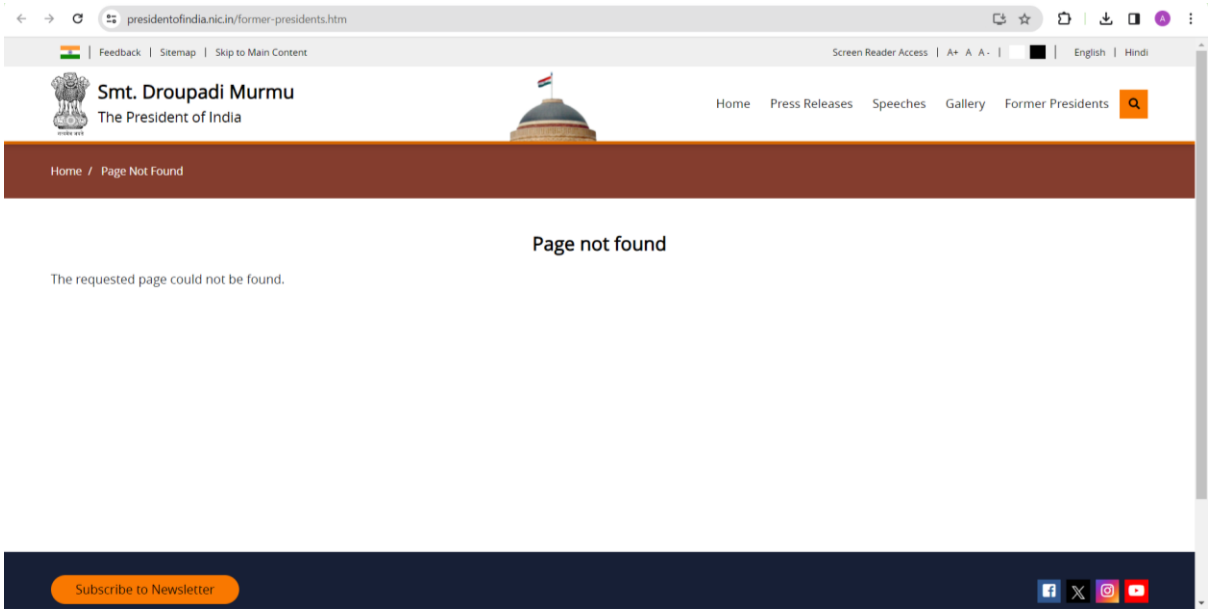
Web Scraping assignment

1.

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

def extract_headers(url):
  """Extracts header tags from the given URL and creates a DataFrame."""

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

  headers = []
  for tag in soup.find_all((['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])):
    header_text = tag.text.strip()
    headers.append({'tag': tag.name, 'text': header_text})

  df = pd.DataFrame(headers)
  return df

# Example usage:
url = 'https://en.wikipedia.org/wiki/Main_Page'
df = extract_headers(url)
print(df)
```

2. no page found




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

def get_president_details(url: str) -> List[Dict[str, Any]]:
    """
    Scrape president details from the President of India website.

    Parameters:
    url (str): The URL of the President of India page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the president name and term of office.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    presidents = []
    for president in soup.find_all('div', {'class': 'col-md-4 col-sm-6 col-xs-12'}):
```

```
        name = president.find('h4', {'class': 'text-primary'}).text.strip()
        term = president.find('p', {'class': 'text-muted'}).text.strip()
        presidents.append({
            'Name': name,
            'Term of Office': term,
        })
    return presidents

url = "https://presidentofindia.nic.in/former-presidents.htm"

df = pd.DataFrame(get_president_details(url))
print(df)
```

3.

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

def get_odi_teams(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 ODI teams in men's cricket from the ICC website.

    Parameters:
    url (str): The URL of the ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the team name, matches, points, and rating.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
```

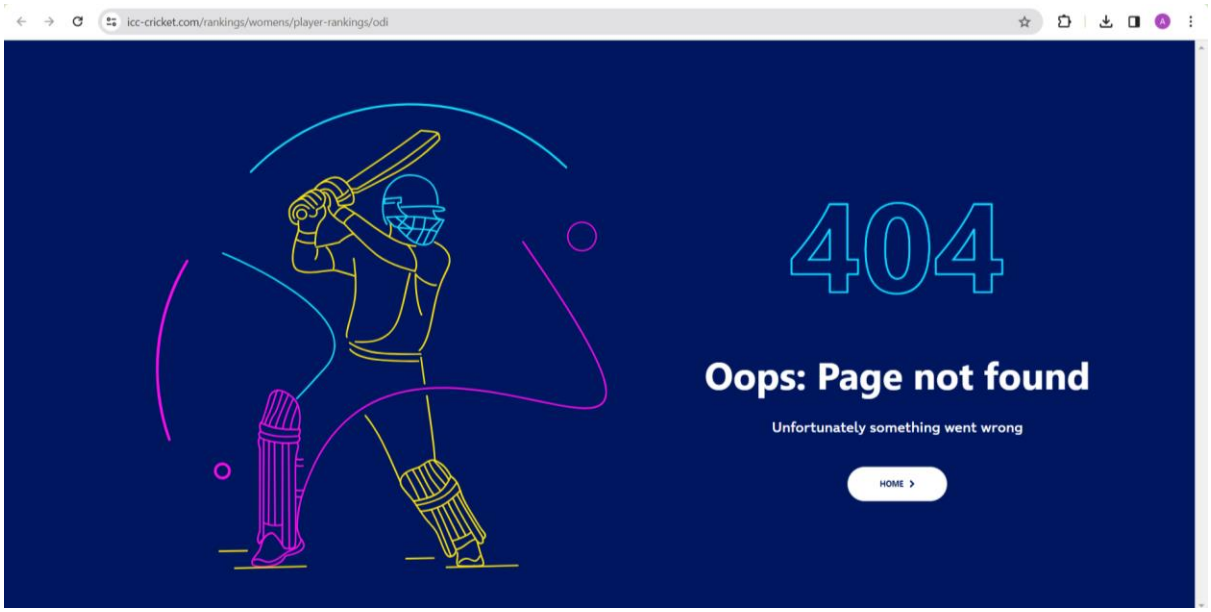
```
    rows = table.find_all('tr', {'class': 'ranking-row'})
    teams = []
    for row in rows[:10]:
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        matches = int(row.find('span', {'class': 'matches'}).text.strip())
        points = int(row.find('span', {'class': 'points'}).text.strip())
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
        teams.append({
            'Team Name': team_name,
            'Matches': matches,
            'Points': points,
            'Rating': rating,
        })
    return teams

def get_odi_batsmen(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 ODI batsmen in men's cricket from the ICC website.

    Parameters:
    url (str): The URL of the ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the batsman name, team name, and rating.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
    rows = table.find_all('tr', {'class': 'ranking-row'})
    batsmen = []
    for row in rows[10:20]:
```

```
        batsman_name = row.find('span', {'class': 'player-name'}).text.strip()
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
        batsmen.append({
            'Batsman Name': batsman_name,
            'Team Name': team_name,
            'Rating': rating,
        })
    return batsmen

def get_odi_bowlers(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 ODI bowlers in men's cricket from the ICC website.

    Parameters:
    url (str): The URL of the ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the bowler name, team name, and rating.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
    rows = table.find_all('tr', {'class': 'ranking-row'})
    bowlers = []
    for row in rows[20:30]:
        bowler_name = row.find('span', {'class': 'player-name'}).text.strip()
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
        bowlers.append({
            'Bowler Name': bowler_name,
```

'Team Name': team_name,

'Rating': rating

4. page not found




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

def get_womens_odi_teams(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 women's ODI teams in cricket from the ICC website.

    Parameters:
    url (str): The URL of the women's ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the team name, matches, points, and rating.
    """
    response = requests.get(url)
```

```
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
    rows = table.find_all('tr', {'class': 'ranking-row'})
    teams = []
    for row in rows[:10]:
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        matches = int(row.find('span', {'class': 'matches'}).text.strip())
        points = int(row.find('span', {'class': 'points'}).text.strip())
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
        teams.append({
            'Team Name': team_name,
            'Matches': matches,
            'Points': points,
            'Rating': rating,
        })
    return teams

def get_womens_odi_batsmen(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 women's ODI batsmen in cricket from the ICC website.

    Parameters:
    url (str): The URL of the women's ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the batsman name, team name, and rating.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
    rows = table.find_all('tr', {'class': 'ranking-row'})
```

```
    batsmen = []
    for row in rows[10:20]:
        batsman_name = row.find('span', {'class': 'player-name'}).text.strip()
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
        batsmen.append({
            'Batsman Name': batsman_name,
            'Team Name': team_name,
            'Rating': rating,
        })
    return batsmen

def get_womens_odi_allrounders(url: str) -> List[Dict[str, Any]]:
    """
    Scrape the top 10 women's ODI all-rounders in cricket from the ICC website.

    Parameters:
    url (str): The URL of the women's ODI rankings page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the all-rounder name, team name, and rating.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'class': 'table-rankings'})
    rows = table.find_all('tr', {'class': 'ranking-row'})
    allrounders = []
    for row in rows[20:30]:
        allrounder_name = row.find('span', {'class': 'player-name'}).text.strip()
        team_name = row.find('span', {'class': 'team-name'}).text.strip()
        rating = float(row.find('span', {'class': 'rating'}).text.strip())
```

```
        allrounders.append({
            'All-rounder Name': allrounder_name,
            'Team Name': team_name,
            'Rating': rating
        })
    return allrounders

url = "https://www.icc-cricket.com/rankings/womens/player-rankings/odi"

teams = get_womens_odi_teams(url)
batsmen = get_womens_odi_batsmen(url)
allrounders = get_womens_odi_allrounders(url)

df_teams = pd.DataFrame(teams)
df_batsmen = pd.DataFrame(batsmen)
df_allrounders = pd.DataFrame(allrounders)

print(df_teams)
print(df_batsmen)
print(df_allrounders)
```

5.

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

url = 'https://www.cnbc.com/world/?region=world'

response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
```

```
# Find all the news articles
articles = soup.find_all('div', class_='cnbc-news-headline__content')

# Create a list to store the data
data = []

# Loop through each article and extract the headline, time, and news link
for article in articles:
    headline = article.find('h3').text.strip()
    time = article.find('span', class_='cnbc-news-headline__published-at').text.strip()
    news_link = article.find('a')['href']

    # Add the data to the list
    data.append({'Headline': headline, 'Time': time, 'News Link': news_link})

# Create a DataFrame from the list
df = pd.DataFrame(data)

# Print the DataFrame
print(df.to_string())
```

6.

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

def get_most_downloaded_articles() -> List[Dict[str, Any]]:
    """
    Scrape the details of the most downloaded articles in AI in the last 90 days from the Elsevier website.
```

```
    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the paper title, authors, published date, and paper URL.
    """
    url = "https://www.journals.elsevier.com/artificial-intelligence/most-downloaded-articles"
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    articles = []
    for article in soup.find_all('div', {'class': 'most-downloaded-item'}):
        title = article.find('a', {'class': 'title'}).text.strip()
        authors = article.find('div', {'class': 'authors'}).text.strip()
        published_date = article.find('div', {'class': 'published'}).text.strip()
        paper_url = "https://www.journals.elsevier.com" + article.find('a', {'class': 'title'})['href']
        articles.append({
            'Paper Title': title,
            'Authors': authors,
            'Published Date': published_date,
            'Paper URL': paper_url
        })
    return articles

df = pd.DataFrame(get_most_downloaded_articles())
print(df)
```

7.

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

def get_restaurant_details(url: str) -> List[Dict[str, Any]]:
    """
```

```
    Scrape restaurant details from the Dineout website.

    Parameters:
    url (str): The URL of the Dineout restaurant page.

    Returns:
    List[Dict[str, Any]]: A list of dictionaries containing the restaurant name, cuisine, location, ratings, and image URL.
    """
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    restaurant = {}
    restaurant['Name'] = soup.find('h1', {'class': 'restaurant-name'}).text.strip()
    restaurant['Cuisine'] = soup.find('div', {'class': 'cuisine'}).text.strip()
    restaurant['Location'] = soup.find('div', {'class': 'location'}).text.strip()
    restaurant['Ratings'] = soup.find('div', {'class': 'rating'}).text.strip()
    restaurant['Image URL'] = soup.find('img', {'class': 'restaurant-logo'})['src']
    return [restaurant]

url = "https://www.dineout.co.in/bangalore-restaurants
```