Web scraping Assignment. Prepared By: Francis Afuwah

In [1]:
```
!pip install bs4
```

```
Requirement already satisfied: bs4 in c:\anaconda\lib\site-packages (0.0.2)
Requirement already satisfied: beautifulsoup4 in c:\anaconda\lib\site-packages (from bs4) (4
.12.2)
Requirement already satisfied: soupsieve>1.2 in c:\anaconda\lib\site-packages (from beautifu
lsoup4->bs4) (2.4)
```

In [2]:
```
!pip install requests
```

```
Requirement already satisfied: requests in c:\anaconda\lib\site-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\anaconda\lib\site-packages (fr
om requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\anaconda\lib\site-packages (from requests)
 (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\anaconda\lib\site-packages (from req
uests) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda\lib\site-packages (from req
uests) (2023.11.17)
```

In [3]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Q1. Python program to display all the header tags from wikipedia.org and make data frame

In [4]:
```
def get_wikipedia_headers(url):
    # Fetch the HTML content of the Wikipedia page
    response = requests.get(url)
```

In [5]:
```
header_tags = requests.get('http://www.wikipedia.org')
```

In [6]:
```
header_tags
```

Out[6]:
```
<Response [200]>
```

In [7]:
```
def get_wikipedia_headers(url):
    # Fetch the HTML content of the Wikipedia page
    response = requests.get(url)
    if response.status_code == 200:
        # Parse the HTML content
        soup = BeautifulSoup(response.text, 'html.parser')
```

In [8]:
```
soup = BeautifulSoup("page.content")
```

In [9]:
```
soup
```

Out[9]:
```
<html><body><p>page.content</p></body></html>
```

In [10]:
```
def get_wikipedia_headers(url):
    # Fetch the HTML content of the Wikipedia page
    response = requests.get(url)

    if response.status_code == 200:
```

```
            # Parse the HTML content
            soup = BeautifulSoup(response.text, 'html.parser')

            # Extract header tags (h1, h2, h3, h4, h5, h6)
            headers = soup.find_all(['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])

            # Create a list to store the header text
            header_text = [header.text.strip() for header in headers]

            # Create a DataFrame
            df = pd.DataFrame({'Header': header_text})

            return df
        else:
            print(f"Failed to fetch Wikipedia page. Status code: {response.status_code}")
            return None

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

    # Get headers and create DataFrame
    headers_df = get_wikipedia_headers(wikipedia_url)

    if headers_df is not None:
        # Display the DataFrame
        print(headers_df)
```

```
                        Header
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       Today's featured picture
7       Other areas of Wikipedia
8    Wikipedia's sister projects
9            Wikipedia languages
```

Q5. Python program to scrape mentioned news details from https://www.cnbc.com/world/?region=world and make data frame

In [11]:
```
def scrape_cnbc_news(url):
    # Send an HTTP request to the URL
    response = requests.get(url)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the page
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find the relevant elements on the page
        headlines = soup.find_all('a', class_='Card-title-link')
```

```
            times = soup.find_all('time', class_='Card-time')
            links = soup.find_all('a', class_='Card-title-link', href=True)

            # Extract the text content from the elements
            headlines_text = [headline.text.strip() for headline in headlines]
            times_text = [time.text.strip() for time in times]
            links_text = [link['href'] for link in links]

            # Create a DataFrame using pandas
            df = pd.DataFrame({
                'Headline': headlines_text,
                'Time': times_text,
                'News Link': links_text
            })

            return df
        else:
            print(f"Error: Unable to fetch the webpage (Status Code:
    {response.status_code})")
            return None

    # URL of the CNBC World page
    url = 'https://www.cnbc.com/world/?region=world'

    # Scrape news details and create a DataFrame
    cnbc_news_df = scrape_cnbc_news(url)

    # Display the DataFrame
    if cnbc_news_df is not None:
        print(cnbc_news_df)
```

```
Empty DataFrame
Columns: [Headline, Time, News Link]
Index: []
```

Q6. 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 frame

In [12]:
```
def scrape_elsevier_most_downloaded(url):
    # Send an HTTP request to the URL
    response = requests.get(url)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the page
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find the relevant elements on the page
        articles = soup.find all('li', class ='js-article-list-item')
```

```
            # Initialize lists to store extracted information
            titles = []
            authors_list = []
            published_dates = []
            paper_urls = []

            # Extract information from each article
            for article in articles:
                title = article.find('a', class_='js-article-title')
                authors = article.find('div', class_='js-authors-list')
                published_date = article.find('span', class_='js-article-date')
                paper_url = title['href'] if title else None

                # Extract text content from elements
                title_text = title.text.strip() if title else None
                authors_text = authors.text.strip() if authors else None
                published_date_text = published_date.text.strip() if published_date else
    None

                # Append extracted information to lists
                titles.append(title_text)
                authors_list.append(authors_text)
                published_dates.append(published_date_text)
                paper_urls.append(paper_url)

            # Create a DataFrame using pandas
            df = pd.DataFrame({
                'Paper Title': titles,
                'Authors': authors_list,
                'Published Date': published_dates,
                'Paper URL': paper_urls
            })

            return df
        else:
            print(f"Error: Unable to fetch the webpage (Status Code:
    {response.status_code})")
            return None

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

    # Scrape details and create a DataFrame
    elsevier_df = scrape_elsevier_most_downloaded(url)

    # Display the DataFrame
    if elsevier_df is not None:
        print(elsevier_df)
```

```
Empty DataFrame
Columns: [Paper Title, Authors, Published Date, Paper URL]
Index: []
```

Q7. Python program to scrape mentioned details from

dineout.co.in and make data frame

In [13]:
```
def scrape_dineout_details(url):
    # Send an HTTP request to the URL
    response = requests.get(url)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the page
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find the relevant elements on the page
        restaurants = soup.find_all('div', class_='restnt-card')

        # Initialize lists to store extracted information
        restaurant_names = []
        cuisines = []
        locations = []
        ratings = []
        image_urls = []

        # Extract information from each restaurant
        for restaurant in restaurants:
            name = restaurant.find('h4', class_='restnt-card-main-title')
            cuisine = restaurant.find('p', class_='restnt-card-cuisines')
            location = restaurant.find('p', class_='restnt-card-location')
            rating = restaurant.find('span', class_='restnt-rating')
            image = restaurant.find('img', class_='restnt-card-img')

            # Extract text content from elements
            name_text = name.text.strip() if name else None
            cuisine_text = cuisine.text.strip() if cuisine else None
            location_text = location.text.strip() if location else None
            rating_text = rating.text.strip() if rating else None
            image_url = image['data-src'] if image else None

            # Append extracted information to lists
            restaurant_names.append(name_text)
            cuisines.append(cuisine_text)
            locations.append(location_text)
            ratings.append(rating_text)
            image_urls.append(image_url)

        # Create a DataFrame using pandas
        df = pd.DataFrame({
            'Restaurant Name': restaurant_names,
            'Cuisine': cuisines,
            'Location': locations,
            'Ratings': ratings,
            'Image URL': image_urls
        })

        return df
    else:
        print(f"Error: Unable to fetch the webpage (Status Code:
{response.status_code})")
```

```
        return None

# URL of the Dineout webpage
url = 'https://www.dineout.co.in/delhi-restaurants'

# Scrape details and dataFrame
dineout_df = scrape_dineout_details(url)

# Display DataFrame
if dineout_df is not None:
    print(dineout_df)
```

```
Empty DataFrame
Columns: [Restaurant Name, Cuisine, Location, Ratings, Image URL]
Index: []
```

Q3.Python program to scrape cricket rankings from icc-cricket.com. You have to scrape and make data frame

In [14]:
```
def scrape_odi_teams():
    url = 'https://www.icc-cricket.com/rankings/mens/team-rankings/odi'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    teams_data = []
    for team in soup.select('.rankings-block__banner, .table-body'):
        name = team.select_one('.u-hide-phablet').get_text(strip=True)
        matches = team.select_one('.rankings-block__banner--matches, .table-
body__cell.u-center-text').get_text(strip=True)
        points = team.select_one('.rankings-block__banner--points, .table-
body__cell.u-center-text').get_text(strip=True)
        rating = team.select_one('.rankings-block__banner--rating, .table-
body__cell.u-text-right').get_text(strip=True)

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

    df_teams = pd.DataFrame(teams_data[:10])  # Top 10 teams
    return df_teams

def scrape_odi_batsmen():
    url = 'https://www.icc-cricket.com/rankings/mens/player-rankings/odi/batting'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    batsmen_data = []
    for player in soup.select('.rankings-block__banner, .table-body'):
        name = player.select_one('.rankings-block__banner--name, .table-
body__cell.name a').get_text(strip=True)
        team = player.select_one('.rankings-block__banner--nationality, .table-
body__cell.nationality-logo span').get_text(strip=True)
        rating = player.select_one('.rankings-block__banner--rating, .table-
body__cell.u-text-right').get_text(strip=True)

        batsmen_data.append({'Batsman': name, 'Team': team, 'Rating': rating})
```

```
        df_batsmen = pd.DataFrame(batsmen_data[:10])  # Top 10 batsmen
        return df_batsmen

    def scrape_odi_bowlers():
        url = 'https://www.icc-cricket.com/rankings/mens/player-rankings/odi/bowling'
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        bowlers_data = []
        for player in soup.select('.rankings-block__banner, .table-body'):
            name = player.select_one('.rankings-block__banner--name, .table-
    body__cell.name a').get_text(strip=True)
            team = player.select_one('.rankings-block__banner--nationality, .table-
    body__cell.nationality-logo span').get_text(strip=True)
            rating = player.select_one('.rankings-block__banner--rating, .table-
    body__cell.u-text-right').get_text(strip=True)

            bowlers_data.append({'Bowler': name, 'Team': team, 'Rating': rating})

        df_bowlers = pd.DataFrame(bowlers_data[:10])  # Top 10 bowlers
        return df_bowlers

    if __name__ == "__main__":
        # Scraping and displaying the data
        df_teams = scrape_odi_teams()
        print("Top 10 ODI Teams:")
        print(df_teams)

        df_batsmen = scrape_odi_batsmen()
        print("\nTop 10 ODI Batsmen:")
        print(df_batsmen)

        df_bowlers = scrape_odi_bowlers()
        print("\nTop 10 ODI Bowlers:")
        print(df_bowlers)
```

```
Top 10 ODI Teams:
Empty DataFrame
Columns: []
Index: []

Top 10 ODI Batsmen:
Empty DataFrame
Columns: []
Index: []

Top 10 ODI Bowlers:
Empty DataFrame
Columns: []
Index: []
```

Q4. Python program to scrape cricket rankings from icc-cricket.com. You have to scrape and make data frame

In [15]:
```
def get_icc_data(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    return soup

def scrape_odi_team_rankings():
    url = 'https://www.icc-cricket.com/rankings/womens/team-rankings/odi'
    soup = get_icc_data(url)

    teams_data = []
    for team in soup.find_all('tr', class_='table-body'):
        team_info = team.find_all('td')
        team_name = team_info[1].text.strip()
        matches = int(team_info[2].text)
        points = int(team_info[3].text)
        rating = int(team_info[4].text)

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

    df = pd.DataFrame(teams_data[:10])  # Selecting top 10 teams
    return df

def scrape_odi_batting_rankings():
    url = 'https://www.icc-cricket.com/rankings/womens/player-rankings/odi/batting'
    soup = get_icc_data(url)

    players_data = []
    for player in soup.find_all('tr', class_='table-body'):
        player_info = player.find_all('td')
        player_name = player_info[1].text.strip()
        team_name = player_info[2].text.strip()
        rating = int(player_info[3].text)

        players_data.append({
            'Player': player_name,
            'Team': team_name,
            'Rating': rating
        })

    df = pd.DataFrame(players_data[:10])  # Selecting top 10 players
    return df

def scrape_odi_allrounder_rankings():
    url = 'https://www.icc-cricket.com/rankings/womens/player-rankings/odi/all-
rounder'
    soup = get_icc_data(url)

    allrounders_data = []
    for allrounder in soup.find_all('tr', class_='table-body'):
        allrounder_info = allrounder.find_all('td')
        allrounder_name = allrounder_info[1].text.strip()
        team_name = allrounder_info[2].text.strip()
```

```
            rating = int(allrounder_info[3].text)

            allrounders_data.append({
                'Player': allrounder_name,
                'Team': team_name,
                'Rating': rating
            })

        df = pd.DataFrame(allrounders_data[:10])  # Selecting top 10 all-rounders
        return df

    if __name__ == "__main__":
        odi_teams_df = scrape_odi_team_rankings()
        odi_batting_df = scrape_odi_batting_rankings()
        odi_allrounder_df = scrape_odi_allrounder_rankings()

        print("Top 10 ODI Teams in Women's Cricket:")
        print(odi_teams_df)

        print("\nTop 10 Women's ODI Batting Players:")
        print(odi_batting_df)

        print("\nTop 10 Women's ODI All-rounders:")
        print(odi_allrounder_df)
```

```
Top 10 ODI Teams in Women's Cricket:
Empty DataFrame
Columns: []
Index: []

Top 10 Women's ODI Batting Players:
Empty DataFrame
Columns: []
Index: []

Top 10 Women's ODI All-rounders:
Empty DataFrame
Columns: []
Index: []
```