Assignment 1

July 9, 2023

[1]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

[20]:
```
page=requests.get('https://en.wikipedia.org/wiki/Main_Page')
page
```

[20]:
```
<Response [200]>
```

[124]:
```
soup=BeautifulSoup(page.content,"html.parser")
```

[22]:
```
header_tags = soup.find_all(["h1", "h2", "h3"])
```

[23]:
```
header_texts = [tag.text.strip() for tag in header_tags]
```

[24]:
```
df = pd.DataFrame({"Headers": header_texts})
df
```

[24]:

	Headers
0	Main Page
1	Welcome to Wikipedia
2	From today's featured article
3	Did you knowÂă...
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

[29]:
```
def scrape_odi_rankings():
    page=requests.get("https://www.icc-cricket.com/rankings/mens/team-rankings/
 ↪odi")
    page
    soup = BeautifulSoup(page.text, 'html.parser')
    soup
```

```
    table = soup.find('table', class_='table')
    rows = table.find_all('tr')

    data = []

    for row in rows[1:11]:
        cells = row.find_all('td')
        rank = cells[0].text.strip()
        team = cells[1].text.strip()
        matches = cells[2].text.strip()
        points = cells[3].text.strip()
        rating = cells[4].text.strip()

        data.append([rank, team, matches, points, rating])


    df = pd.DataFrame(data, columns=['Rank', 'Team', 'Matches', 'Points',␣
 ↪'Rating'])
    return df

odi_rankings_df = scrape_odi_rankings()
odi_rankings_df
```

[29]:

	Rank	Team	Matches	Points	Rating
0	1	Australia\nAUS	23	2,714	118
1	2	Pakistan\nPAK	20	2,316	116
2	3	India\nIND	33	3,807	115
3	4	New Zealand\nNZ	27	2,806	104
4	5	England\nENG	24	2,426	101
5	6	South Africa\nSA	19	1,910	101
6	7	Bangladesh\nBAN	25	2,451	98
7	8	Sri Lanka\nSL	28	2,378	85
8	9	Afghanistan\nAFG	13	1,067	82
9	10	West Indies\nWI	32	2,201	69

[30]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

page =requests.get("https://www.icc-cricket.com/rankings/mens/player-rankings/
 ↪odi/batting")
page
soup = BeautifulSoup(response.content, "html.parser")
soup

table = soup.find("table", class_="table rankings-table")
```

```
player_names = []
teams = []
ratings = []

rows = table.find_all("tr", class_="table-body")
for row in rows[:10]:  # Limit to the top 10 batsmen
    columns = row.find_all("td")
    player_name = columns[1].find("a").text.strip()
    team = columns[2].text.strip()
    rating = columns[3].text.strip()
    player_names.append(player_name)
    teams.append(team)
    ratings.append(rating)

df = pd.DataFrame({"Player": player_names, "Team": teams, "Rating": ratings})
df
```

[30]:

	Player	Team	Rating
0	Rassie van der Dussen	SA	777
1	Fakhar Zaman	PAK	755
2	Imam-ul-Haq	PAK	745
3	Shubman Gill	IND	738
4	Harry Tector	IRE	726
5	David Warner	AUS	726
6	Virat Kohli	IND	719
7	Quinton de Kock	SA	718
8	Rohit Sharma	IND	707
9	Steve Smith	AUS	702

[31]:
```
page = requests.get("https://www.icc-cricket.com/rankings/mens/player-rankings/
↪odi/bowling")
page
soup = BeautifulSoup(response.content, "html.parser")

table = soup.find("table", class_="table rankings-table")

player_names = []
teams = []
ratings = []

rows = table.find_all("tr", class_="table-body")
for row in rows[:10]:  # Limit to the top 10 bowlers
    columns = row.find_all("td")
    player_name = columns[1].find("a").text.strip()
    team = columns[2].text.strip()
    rating = columns[3].text.strip()
    player_names.append(player_name)
```

```
    teams.append(team)
    ratings.append(rating)

df = pd.DataFrame({"Player": player_names, "Team": teams, "Rating": ratings})
df
```

[31]:

	Player	Team	Rating
0	Rassie van der Dussen	SA	777
1	Fakhar Zaman	PAK	755
2	Imam-ul-Haq	PAK	745
3	Shubman Gill	IND	738
4	Harry Tector	IRE	726
5	David Warner	AUS	726
6	Virat Kohli	IND	719
7	Quinton de Kock	SA	718
8	Rohit Sharma	IND	707
9	Steve Smith	AUS	702

[33]:
```
page =requests.get("https://www.icc-cricket.com/rankings/womens/team-rankings/
 ↪odi")
page
soup = BeautifulSoup(response.content, "html.parser")

table = soup.find("table", class_="table")

team_names = []
matches = []
points = []
ratings = []

rows = table.find_all("tr", class_="table-body")
for row in rows[:10]:
    columns = row.find_all("td")
    team_name = columns[1].text.strip()
    match = columns[2].text.strip()
    point = columns[3].text.strip()
    rating = columns[4].text.strip()
    team_names.append(team_name)
    matches.append(match)
    points.append(point)
    ratings.append(rating)

df = pd.DataFrame({"Team": team_names, "Matches": matches, "Points": points,␣
 ↪"Rating": ratings})
df
```

[33]:

	Team	Matches	Points	Rating
0	England\nENG	28	3,342	119
1	South Africa\nSA	26	3,098	119
2	India\nIND	27	2,820	104
3	New Zealand\nNZ	28	2,688	96
4	West Indies\nWI	29	2,743	95
5	Bangladesh\nBAN	14	977	70
6	Sri Lanka\nSL	12	820	68
7	Thailand\nTHA	12	806	67
8	Pakistan\nPAK	27	1,678	62
9	Ireland\nIRE	16	605	38

[40]:

```
page=requests.get("https://www.icc-cricket.com/rankings/womens/player-rankings/
 ↪odi/batting")
page

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

table = soup.find("table", class_="table rankings-table")

player_names = []
teams = []
ratings = []

rows = table.find_all("tr", class_="table-body")
for row in rows[:10]:
    columns = row.find_all("td")
    player_name = columns[1].find("a").text.strip()
    team = columns[2].text.strip()
    rating = columns[3].text.strip()
    player_names.append(player_name)
    teams.append(team)
    ratings.append(rating)

df = pd.DataFrame({"Player": player_names, "Team": teams, "Rating": ratings})
df
```

[40]:

	Player	Team	Rating
0	Beth Mooney	AUS	754
1	Laura Wolvaardt	SA	732
2	Natalie Sciver	ENG	731
3	Meg Lanning	AUS	717
4	Harmanpreet Kaur	IND	716
5	Smriti Mandhana	IND	714
6	Ellyse Perry	AUS	626
7	Stafanie Taylor	WI	618
8	Tammy Beaumont	ENG	595

```
9       Amelia Kerr   NZ    591
```

[42]:
```
page=requests.get("https://www.icc-cricket.com/rankings/womens/player-rankings/
 ↪odi/all-rounder")
page
soup = BeautifulSoup(response.content, "html.parser")

table = soup.find("table", class_="table rankings-table")

player_names = []
teams = []
ratings = []

rows = table.find_all("tr", class_="table-body")
for row in rows[:10]:
    columns = row.find_all("td")
    player_name = columns[1].find("a").text.strip()
    team = columns[2].text.strip()
    rating = columns[3].text.strip()
    player_names.append(player_name)
    teams.append(team)
    ratings.append(rating)

df = pd.DataFrame({"Player": player_names, "Team": teams, "Rating": ratings})
df
```

[42]:

	Player	Team	Rating
0	Natalie Sciver	ENG	371
1	Ellyse Perry	AUS	366
2	Marizanne Kapp	SA	349
3	Amelia Kerr	NZ	328
4	Deepti Sharma	IND	322
5	Ashleigh Gardner	AUS	292
6	Jess Jonassen	AUS	250
7	Sophie Devine	NZ	233
8	Nida Dar	PAK	232
9	Sophie Ecclestone	ENG	205

[61]:
```
page=requests.get ("https://www.cnbc.com/world/?region=world")
page
soup = BeautifulSoup(response.content, "html.parser")

headline_tags = soup.find_all("a", class_="Card-title")

headlines = [tag.text.strip() for tag in headline_tags]

df = pd.DataFrame({"Headline": headlines})
```

```
df
```

[61]:

	Headline
0	Earnings playbook: JPMorgan Chase and Delta Ai...
1	June inflation data will be closely watched by...
2	The Fed has rolled out a new index for gauging...
3	Biogen shares fall after AlzheimerâĂŹs drug appr...
4	These stocks that are about to turn a profit s...
5	Earnings playbook: JPMorgan Chase and Delta Ai...
6	The Fed rolls out a new economic index that co...
7	Micron and Pfizer are the most oversold S&P 50...
8	Active funds are powering JPMorgan's ETFs past...
9	Ukraine reports advances near eastern city of ...
10	Wagner leader Prigozhin in St. Petersburg, Bel...
11	Zelenskyy warns of provocations at nuclear pla...
12	Wagner's Prigozhin reportedly resurfaces; NATO...
13	Russia's Medvedev cites risks of nuclear war; ...
14	'We are in uncharted territory': World records...
15	Bad news for nervous flyers: Turbulence is get...
16	World registers hottest day since records bega...
17	Millions of workers face up to challenge of he...
18	El NiÃśo has officially begun. UN says phenomen...
19	Southeast Asia's IPO market is an investor fav...
20	We the see biggest opportunities in Indonesia,...
21	Singapore pledged billions to fight climate ch...
22	V3 Gourmet explains why it chose Singapore for...
23	Revenues of 'warung' operators grew after they...
24	The best places to eat in Andalusia âĂŤ from a c...
25	What a 250,000 euro penthouse renovation in Ba...
26	Serving 'lunch' before midnight âĂŤ and other wa...
27	These workers take âĂŸhush trips.âĂŹ HereâĂŹs how th...
28	Michelin Guide adds 17 food stalls in Singapor...
29	28-year-old social media manager in Norway is ...
30	10 rules of ikigai, from authors of the Japane...
31	5 low-stress summer jobs that are hiring right...
32	Americans need to earn over $230,000 a year to...
33	Bill Gates says Warren Buffett taught him how ...

[64]:
```
page=requests.get ("https://www.cnbc.com/world/?region=world")
page
soup = BeautifulSoup(response.content, "html.parser")

time_tags = soup.find_all("time")

times = [tag.text.strip() for tag in time_tags]

df = pd.DataFrame({"Time": times})
```

```
df
```

[64]:

	Time
0	47 Min Ago
1	47 Min Ago
2	47 Min Ago
3	1 Hour Ago
4	2 Hours Ago
5	2 Hours Ago
6	3 Hours Ago
7	4 Hours Ago
8	4 Hours Ago
9	10 Hours Ago
10	21 Hours Ago
11	July 8, 2023
12	July 8, 2023
13	July 8, 2023
14	July 8, 2023
15	July 8, 2023
16	July 8, 2023
17	July 8, 2023
18	July 8, 2023
19	July 8, 2023
20	July 8, 2023
21	July 8, 2023
22	July 7, 2023
23	July 7, 2023
24	July 7, 2023
25	July 7, 2023
26	July 7, 2023
27	July 7, 2023
28	July 7, 2023
29	July 7, 2023

[67]:
```
page= requests.get("https://www.cnbc.com/world/?region=world")
page
soup = BeautifulSoup(response.content, "html.parser")

link_tags = soup.find_all("a", class_="Card-title")

news_links = [tag['href'] for tag in link_tags]

df = pd.DataFrame({"News Link": news_links})
df
```

[67]:

	News Link
0	https://www.cnbc.com/2023/07/09/earnings-playb...

```
1   https://www.cnbc.com/2023/07/07/june-inflation...
2   https://www.cnbc.com/2023/07/08/the-fed-has-ro...
3   https://www.cnbc.com/2023/07/07/biogen-falls-a...
4   https://www.cnbc.com/2023/07/09/these-stocks-t...
5   https://www.cnbc.com/2023/07/09/earnings-playb...
6   https://www.cnbc.com/2023/07/08/the-fed-has-ro...
7   https://www.cnbc.com/2023/07/08/micron-and-pfi...
8   https://www.cnbc.com/2023/07/08/active-funds-a...
9   https://www.cnbc.com/2023/07/07/ukraine-war-li...
10  https://www.cnbc.com/2023/07/06/ukraine-war-li...
11  https://www.cnbc.com/2023/07/05/ukraine-war-li...
12  https://www.cnbc.com/2023/07/04/russia-ukraine...
13  https://www.cnbc.com/2023/07/03/ukraine-war-li...
14  https://www.cnbc.com/2023/07/07/climate-world-...
15  https://www.cnbc.com/2023/07/06/bad-news-for-n...
16  https://www.cnbc.com/2023/07/05/climate-crisis...
17  https://www.cnbc.com/2023/07/05/as-planet-heat...
18  https://www.cnbc.com/2023/07/04/el-nio-un-says...
19  https://www.cnbc.com/2023/07/07/southeast-asia...
20  https://www.cnbc.com/video/2023/06/26/we-the-s...
21  https://www.cnbc.com/2023/06/21/singapore-clim...
22  https://www.cnbc.com/video/2023/06/19/v3-gourm...
23  https://www.cnbc.com/video/2023/06/12/bukalapa...
24  https://www.cnbc.com/2023/07/06/the-best-place...
25  https://www.cnbc.com/2023/06/26/real-estate-in...
26  https://www.cnbc.com/2023/06/22/how-to-reduce-...
27  https://www.cnbc.com/2023/06/20/hush-trips-her...
28  https://www.cnbc.com/2023/06/16/michelin-guide...
29  https://www.cnbc.com/2023/07/09/how-a-norway-2...
30  https://www.cnbc.com/2023/07/09/10-rules-of-ik...
31  https://www.cnbc.com/2023/07/09/5-low-stress-s...
32  https://www.cnbc.com/2023/07/09/salary-america...
33  https://www.cnbc.com/2023/07/09/bill-gates-say...
```

[71]:
```
pages = requests.get("https://www.journals.elsevier.com/artificial-intelligence/
 ↪most-downloaded-articles")
page
soup = BeautifulSoup(response.content, "html.parser")

article_tags = soup.find_all("h2", class_="js-article-title")

article_titles = [tag.text.strip() for tag in article_tags]

df = pd.DataFrame({"Paper Title": article_titles})
df
```

[71]:
```
Empty DataFrame
Columns: [Paper Title]
Index: []
```

[72]:
```
page = requests.get("https://www.journals.elsevier.com/artificial-intelligence/
 ↪most-downloaded-articles")
page
soup = BeautifulSoup(response.content, "html.parser")

article_tags = soup.find_all("h2", class_="js-article-title")

authors_list = []

for article in article_tags:
    parent_div = article.find_next("div", class_="text-xs")

    author_tags = parent_div.find_all("span", class_="author")

    authors = [author.text.strip() for author in author_tags]

    authors_list.append(authors)

flattened_authors = [author for sublist in authors_list for author in sublist]

df = pd.DataFrame({"Authors": flattened_authors})
df
```

[72]:
```
Empty DataFrame
Columns: [Authors]
Index: []
```

[75]:
```
page =requests.get ("https://www.journals.elsevier.com/artificial-intelligence/
 ↪most-downloaded-articles")
page
soup = BeautifulSoup(response.content, "html.parser")

article_tags = soup.find_all("h2", class_="js-article-title")

published_dates = []

for article in article_tags:
    date_tag = article.find_next("span",␣
 ↪class_="js-article-history-date-published")
    published_date = date_tag.text.strip()
    published_dates.append(published_date)

df = pd.DataFrame({"Published Date": published_dates})
```

```
df
```

[75]:
```
Empty DataFrame
Columns: [Published Date]
Index: []
```

[77]:
```
page =requests.get ("https://www.journals.elsevier.com/artificial-intelligence/
↪most-downloaded-articles")
page
soup = BeautifulSoup(response.content, "html.parser")

article_tags = soup.find_all("h2", class_="js-article-title")

paper_urls = []

for article in article_tags:
    link_tag = article.find_next("a", class_="result-list-title-link")
    paper_url = link_tag["href"]
    paper_urls.append(paper_url)

df = pd.DataFrame({"Paper URL": paper_urls})
df
```

[77]:
```
Empty DataFrame
Columns: [Paper URL]
Index: []
```

[79]:
```
page = requests.get ("https://www.dineout.co.in/delhi-restaurants")
page
soup = BeautifulSoup(response.content, "html.parser")

restaurant_tags = soup.find_all("a", class_="restnt-name ellipsis")

restaurant_names = [tag.text.strip() for tag in restaurant_tags]

df = pd.DataFrame({"Restaurant Name": restaurant_names})
df
```

[79]:

	Restaurant Name
0	Local
1	Tamasha
2	Ministry Of Beer
3	Station Bar
4	My Bar Square
5	Warehouse Cafe
6	Openhouse Cafe
7	Connaught Club House

```
8                Unplugged Courtyard
9                            Berco's
10             The Junkyard Cafe
11                 The G.T. Road
12              Lord of the Drinks
13                    38 Barracks
14  Ardor 2.1 Restaurant and Lounge
15                             QBA
16                    Cafe High 5
17                    Dasaprakash
18              The Imperial Spice
19              Cafe Delhi Heights
20       Somewhere Restaurant & Bar
```

[80]:
```
page = requests.get ("https://www.dineout.co.in/delhi-restaurants")
page
soup = BeautifulSoup(response.content, "html.parser")

cuisine_tags = soup.find_all("span", class_="double-line-ellipsis")

cuisine_types = [tag.text.strip() for tag in cuisine_tags]

df = pd.DataFrame({"Cuisine": cuisine_types})
df
```

[80]:
```
                                              Cuisine
0   âĆź 2,000 for 2 (approx) | North Indian, Asian, ...
1   âĆź 2,000 for 2 (approx) | Continental, Asian, I...
2   âĆź 3,000 for 2 (approx) | North Indian, Contine...
3   âĆź 1,100 for 2 (approx) | Italian, Chinese, Nor...
4   âĆź 2,000 for 2 (approx) | Finger Food, Chinese,...
5   âĆź 2,500 for 2 (approx) | North Indian, Chinese...
6   âĆź 2,000 for 2 (approx) | North Indian, Asian, ...
7   âĆź 1,800 for 2 (approx) | North Indian, Contine...
8   âĆź 3,300 for 2 (approx) | North Indian, Italian...
9              âĆź 1,300 for 2 (approx) | Chinese, Thai
10  âĆź 2,100 for 2 (approx) | North Indian, Contine...
11             âĆź 2,000 for 2 (approx) | North Indian
12  âĆź 2,500 for 2 (approx) | Chinese, North Indian...
13  âĆź 2,700 for 2 (approx) | North Indian, Chinese...
14  âĆź 2,000 for 2 (approx) | North Indian, Chinese...
15  âĆź 2,100 for 2 (approx) | North Indian, Contine...
16  âĆź 1,700 for 2 (approx) | North Indian, Contine...
17  âĆź 800 for 2 (approx) | North Indian, South Ind...
18  âĆź 3,000 for 2 (approx) | North Indian, Chinese...
19  âĆź 1,300 for 2 (approx) | Continental, North In...
20  âĆź 1,000 for 2 (approx) | North Indian, Contine...
```

[90]:
```
page =requests.get ("https://www.dineout.co.in/delhi-restaurants")
page
soup = BeautifulSoup(response.content, "html.parser")

location_tags = soup.find_all("div", class_="restnt-loc ellipsis")

locations = [tag.text.strip() for tag in location_tags]

df = pd.DataFrame({"Location": locations})
df
```

[90]:

	Location
0	Scindia House,Connaught Place, Central Delhi
1	Connaught Place, Central Delhi
2	M-Block,Connaught Place, Central Delhi
3	F-Block,Connaught Place, Central Delhi
4	Connaught Place, Central Delhi
5	Connaught Place, Central Delhi
6	Connaught Place, Central Delhi
7	Connaught Place, Central Delhi
8	Connaught Place, Central Delhi
9	Connaught Place, Central Delhi
10	Connaught Place, Central Delhi
11	M-Block,Connaught Place, Central Delhi
12	Connaught Place, Central Delhi
13	M-Block,Connaught Place, Central Delhi
14	Connaught Place, Central Delhi
15	Connaught Place, Central Delhi
16	Connaught Place, Central Delhi
17	Connaught Place, Central Delhi
18	M-Block,Connaught Place, Central Delhi
19	Janpath, Central Delhi
20	Connaught Place, Central Delhi

[94]:
```
page =requests.get ("https://www.dineout.co.in/delhi-restaurants")
page
soup = BeautifulSoup(response.content, "html.parser")

rating_tags = soup.find_all("div", class_="restnt-rating rating-4")

ratings = [tag.text.strip() for tag in rating_tags]

df = pd.DataFrame({"Rating": ratings})
df
```

[94]:

	Rating
0	4

```
1     4.2
2       4
3       4
4     3.9
5     4.1
6     4.1
7     4.2
8       4
9     4.3
10    4.1
11    4.3
12    4.2
13    4.3
14    4.1
15    4.2
16      4
17    4.2
18    4.4
19    4.3
20    4.1
```

[123]:
```
pages =requests.get ("https://www.dineout.co.in/delhi-restaurants")
pages
soup = BeautifulSoup(response.content, "html.parser")
images = []
for i in soup.find_all("img",class_="no-img"):
    images.append(i.get('data-src'))
images
df = pd.DataFrame({"images": images})
df
```

[123]:

	images
0	https://im1.dineout.co.in/images/uploads/resta...
1	https://im1.dineout.co.in/images/uploads/resta...
2	https://im1.dineout.co.in/images/uploads/resta...
3	https://im1.dineout.co.in/images/uploads/resta...
4	https://im1.dineout.co.in/images/uploads/resta...
5	https://im1.dineout.co.in/images/uploads/resta...
6	https://im1.dineout.co.in/images/uploads/resta...
7	https://im1.dineout.co.in/images/uploads/resta...
8	https://im1.dineout.co.in/images/uploads/resta...
9	https://im1.dineout.co.in/images/uploads/resta...
10	https://im1.dineout.co.in/images/uploads/resta...
11	https://im1.dineout.co.in/images/uploads/resta...
12	https://im1.dineout.co.in/images/uploads/resta...
13	https://im1.dineout.co.in/images/uploads/resta...
14	https://im1.dineout.co.in/images/uploads/resta...

```
15  https://im1.dineout.co.in/images/uploads/resta...
16  https://im1.dineout.co.in/images/uploads/resta...
17  https://im1.dineout.co.in/images/uploads/resta...
18  https://im1.dineout.co.in/images/uploads/resta...
19  https://im1.dineout.co.in/images/uploads/resta...
20  https://im1.dineout.co.in/images/uploads/resta...
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: