In [6]:

```
import requests
from bs4 import BeautifulSoup

url = requests.get('https://en.wikipedia.org/wiki/Main_Page')
soup = BeautifulSoup(url.text, 'html.parser')
story = soup.find_all(['h1','h2','h3'])
for i in story:
    print(i)
```

```
<h1 class="firstHeading" id="firstHeading">Main Page</h1>
<h2 class="mp-h2" id="mp-tfa-h2"><span id="From_today.27s_featured_articl
e"></span><span class="mw-headline" id="From_today's_featured_article">From
today's featured article</span></h2>
<h2 class="mp-h2" id="mp-dyk-h2"><span class="mw-headline" id="Did_you_know
_...">Did you know ...</span></h2>
<h2 class="mp-h2" id="mp-itn-h2"><span class="mw-headline" id="In_the_new
s">In the news</span></h2>
<h2 class="mp-h2" id="mp-otd-h2"><span class="mw-headline" id="On_this_da
y">On this day</span></h2>
<h2 class="mp-h2" id="mp-tfp-h2"><span id="Today.27s_featured_picture"></sp
an><span class="mw-headline" id="Today's_featured_picture">Today's featured
picture</span></h2>
<h2 class="mp-h2" id="mp-other"><span class="mw-headline" id="Other_areas_o
f_Wikipedia">Other areas of Wikipedia</span></h2>
<h2 class="mp-h2" id="mp-sister"><span id="Wikipedia.27s_sister_projects">
</span><span class="mw-headline" id="Wikipedia's_sister_projects">Wikipedi
a's sister projects</span></h2>
<h2 class="mp-h2" id="mp-lang"><span class="mw-headline" id="Wikipedia_lang
uages">Wikipedia languages</span></h2>
<h2>Navigation menu</h2>
<h3 class="vector-menu-heading" id="p-personal-label">
<span>Personal tools</span>
</h3>
<h3 class="vector-menu-heading" id="p-namespaces-label">
<span>Namespaces</span>
</h3>
<h3 class="vector-menu-heading" id="p-variants-label">
<span>Variants</span>
</h3>
<h3 class="vector-menu-heading" id="p-views-label">
<span>Views</span>
</h3>
<h3 class="vector-menu-heading" id="p-cactions-label">
<span>More</span>
</h3>
<h3>
<label for="searchInput">Search</label>
</h3>
<h3 class="vector-menu-heading" id="p-navigation-label">
<span>Navigation</span>
</h3>
<h3 class="vector-menu-heading" id="p-interaction-label">
<span>Contribute</span>
</h3>
<h3 class="vector-menu-heading" id="p-tb-label">
<span>Tools</span>
```

```
</h3>
<h3 class="vector-menu-heading" id="p-coll-print_export-label">
<span>Print/export</span>
</h3>
<h3 class="vector-menu-heading" id="p-wikibase-otherprojects-label">
<span>In other projects</span>
</h3>
<h3 class="vector-menu-heading" id="p-lang-label">
<span>Languages</span>
</h3>
```

In [7]:

```
from requests import get
url = get("https://www.imdb.com/search/title/?groups=top_100&sort=user_rating
request = url.text
from bs4 import BeautifulSoup as Soup
soup_data = Soup(request,'html.parser')
soup_data.title.text
movies = soup_data.findAll('div',{'class':'lister-item mode-advanced'})
first_movie = movies[0]
first_movie.h3.a.text #Name of the movie
first_movie.find('span',{"class":"lister-item-year text-muted unbold"}).text[
first_movie.find('div',{"class":"inline-block ratings-imdb-rating"})['data-va
first_movie.find('div',{"class":"inline-block ratings-metascore"}).span.text.
first_movie.find('span',{"name":'nv'})['data-value']
Name = []
Year = []
Rating = []
Meta_score = []
Votes = []
for i in movies:
    Name.append(i.h3.a.text)
    Year.append(i.find('span',{"class":"lister-item-year text-muted unbold"})
    Rating.append(i.find('div',{"class":"inline-block ratings-imdb-rating"})[
    try:
        Meta_score.append(i.find('div',{"class":"inline-block ratings-metasco
    except:
        Meta_score.append(0)
    Votes.append(i.find('span',{"name":'nv'})['data-value'])
```

In [9]:

```
data = list(zip(Name,Year,Rating,Meta_score,Votes))
```

In [10]:

```
import pandas as pd
df = pd.DataFrame(data,columns=['Name',"Year","Rating","Metascore","Votes"])
```

In [11]:
```
df.head()
```

Out[11]:

	Name	Year	Rating	Metascore	Votes
0	The Shawshank Redemption	1994	9.3	80	2410169
1	The Godfather	1972	9.2	100	1667646
2	The Dark Knight	2008	9	84	2368918
3	The Godfather: Part II	1974	9	90	1158607
4	12 Angry Men	1957	9	96	710254

In [12]:
```
df.tail()
```

Out[12]:

	Name	Year	Rating	Metascore	Votes
45	Back to the Future	1985	8.5	87	1088153
46	Once Upon a Time in the West	1968	8.5	80	308861
47	Psycho	1960	8.5	97	618439
48	Rear Window	1954	8.5	100	454219
49	Casablanca	1942	8.5	100	532345

In [13]:
```
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [14]:
```
## Request page source from URL
```

In [15]:
```
url = "https://www.imdb.com/india/top-rated-indian-movies/"
page = requests.get(url)
page
```

Out[15]:
```
<Response [200]>
```

In [16]:
```
## display the page source code
page.content
```

Out[16]:
```
b'\n\n\n<!DOCTYPE html>\n<html\n    xmlns:og="http://ogp.me/ns#"\n    xm
lns:fb="http://www.facebook.com/2008/fbml">\n    <head>\n        \n
<meta charset="utf-8">\n        <meta http-equiv="X-UA-Compatible" conte
nt="IE=edge">\n\n    \n    \n    \n\n    \n    \n    \n\n    <meta name
="apple-itunes-app" content="app-id=342792525, app-argument=imdb:///?src
=mdot">\n            <style>\n                body#styleguide-v2 {\n
background: no-repeat fixed center top #000;\n                }\n
</style>\n        <style>\n            body#styleguide-v2 #root {\n
box-shadow: none;\n            }\n        </style>\n\n\n\n        <scrip
t type="text/javascript">var IMDbTimer={starttime: new Date().getTime(),
pt:\'java\'};</script>\n\n<script>\n    if (typeof uet == \'function\')
{\n      uet("bb", "LoadTitle", {wb: 1});\n    }\n</script>\n  <script>
(function(t){ (t.events = t.events || {})["csm_head_pre_title"] = new Da
te().getTime(); })(IMDbTimer);</script>\n        <title>Top Rated Indian
Movies - IMDb</title>\n  <script>(function(t){ (t.events = t.events ||
{})["csm_head_post_title"] = new Date().getTime(); })(IMDbTimer);</scrip
t>\n<script>\n    if (typeof uet == \'function\') {\n      uet("be", "Lo
adTitle", {wb: 1});\n    }\n</script>\n<script>\n    if (typeof uex ==
\'function\') {\n      uex("ld", "LoadTitle", {wb: 1});\n    }\n</script
```

In [17]:
```
soup = BeautifulSoup(page.content,"html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="http://ogp.
me/ns#">
 <head>
  <meta charset="utf-8"/>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
  <meta content="app-id=342792525, app-argument=imdb:///?src=mdot" name
="apple-itunes-app"/>
  <style>
   body#styleguide-v2 {
                    background: no-repeat fixed center top #000;
                }
  </style>
  <style>
   body#styleguide-v2 #root {
                box-shadow: none;
            }
  </style>
  <script type="text/javascript">
```

In [18]:
```
#scrap movie names
scraped_movies = soup.find_all('td', class_='titleColumn')
scraped_movies
```

Out[18]:
```
[<td class="titleColumn">
      1.
      <a href="/title/tt0093603/" title="Mani Ratnam (dir.), Kamal Haas
an, Saranya Ponvannan">Nayakan</a>
 <span class="secondaryInfo">(1987)</span>
 </td>,
 <td class="titleColumn">
      2.
      <a href="/title/tt0048473/" title="Satyajit Ray (dir.), Kanu Bann
erjee, Karuna Bannerjee">Pather Panchali</a>
 <span class="secondaryInfo">(1955)</span>
 </td>,
 <td class="titleColumn">
      3.
      <a href="/title/tt8176054/" title="Mari Selvaraj (dir.), Kathir,
Anandhi">Pariyerum Perumal</a>
 <span class="secondaryInfo">(2018)</span>
 </td>,
 <td class="titleColumn">
```

In [19]:
```
# parse movie names
movies = []
for movie in scraped_movies:
    movie = movie.get_text().replace('\n',"")
    movie = movie.strip(" ")
    movies.append(movie)
movies
```

Out[19]:
```
['1.      Nayakan(1987)',
 '2.      Pather Panchali(1955)',
 '3.      Pariyerum Perumal(2018)',
 '4.      Anbe Sivam(2003)',
 '5.      Golmaal(1979)',
 '6.      C/o Kancharapalem(2018)',
 '7.      Apur Sansar(1959)',
 '8.      Manichitrathazhu(1993)',
 '9.      Kireedam(1989)',
 '10.      Natsamrat(2016)',
 '11.      96(2018)',
 '12.      Thevar Magan(1992)',
 '13.      Black Friday(2004)',
 '14.      Kumbalangi Nights(2019)',
 '15.      Drishyam 2(2021)',
 '16.      Visaaranai(2015)',
 '17.      3 Idiots(2009)',
 '18.      Taare Zameen Par(2007)',
 '19.      Jersey(2019)',
```

In [20]:
```
# scrap rating for movies
scraped_ratings = soup.find_all('td',class_="ratingColumn imdbRating")
scraped_ratings
# parse ratings
ratings = []
for rating in scraped_ratings:
    rating = rating.get_text().replace('\n','')
    ratings.append(rating)
ratings
```

Out[20]:
```
['8.5',
 '8.5',
 '8.5',
 '8.5',
 '8.5',
 '8.5',
 '8.5',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.4',
 '8.3',
 '8.3',
 '8.3',
```

In [21]:
```
## store the scraped Data
data = pd.DataFrame()
data['Movie Names'] = movies
data['Ratings'] = ratings
data.head()
```

Out[21]:

	Movie Names	Ratings
0	1. Nayakan(1987)	8.5
1	2. Pather Panchali(1955)	8.5
2	3. Pariyerum Perumal(2018)	8.5
3	4. Anbe Sivam(2003)	8.5
4	5. Golmaal(1979)	8.5

In [22]:
```
data.tail()
```

Out[22]:

	Movie Names	Ratings
245	246. Kuch Kuch Hota Hai(1998)	7.5
246	247. Jodhaa Akbar(2008)	7.5
247	248. Trapped(2016)	7.5
248	249. Parmanu: The Story of Pokhran(2018)	7.5
249	250. Delhi Belly(2011)	7.5

In [23]:
```
# Imports
import requests
from bs4 import BeautifulSoup
import pandas as pd
url = "https://bookpage.com/"
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')
print(soup)
```

```
<!DOCTYPE html>

<html lang="en" xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="h
ttp://opengraphprotocol.org/schema/">
<head>
<title>BookPage | Discover your next great book!</title>
<meta content="Find expert book recommendations for the best books to re
ad in all genres." name="description"/>
<meta content="books and literature, writing and writers" name="keyword
s"/>
<link href="https://bookpage.com/" rel="canonical"/>
<meta content="BookPage.com" property="og:site_name"/>
<meta content="Discover your next great book!" property="og:title"/>
<meta content="Find expert book recommendations for the best books to re
ad in all genres." property="og:description"/>
<meta content="website" property="og:type"/>
<meta content="https://bookpage.com/" property="og:url"/>
<meta content="//www.bookpage.com/default_image.jpg" property="og:imag
e"/>
<meta content="summary" name="twitter:card"/>
```

In [24]:
```
title = soup.find_all('h4',class_='italic')
print(title)

for x in soup.find_all('h4',class_='italic'):
    print(x.text)
```

```
[<h4 class="italic">
<a href="/interviews/26437-leah-johnson-ya">Leah Johnson</a>
</h4>, <h4 class="italic">
<a href="/behind-the-book/26415-embracing-practice-that-doesnt-make-perfect
-nonfiction">Embracing a practice that doesn’t make perfect</a>
</h4>, <h4 class="italic">
<a href="/reviews/26396-gabi-snyder-listen-childrens">Listen</a>
</h4>, <h4 class="italic">
<a href="/reviews/26434-leah-johnson-rise-to-sun-ya"> <span style="font-sty
le:normal;">★ </span>Rise to the Sun</a>
</h4>, <h4 class="italic">
<a href="/reviews/26376-brandon-taylor-filthy-animals-fiction">Filthy Anima
ls</a>
</h4>, <h4 class="italic">
<a href="/reviews/26423-kate-moore-woman-they-could-not-silence-history">Th
e Woman They Could Not Silence</a>
</h4>]

Leah Johnson


Embracing a practice that doesn’t make perfect


Listen


 ★ Rise to the Sun


Filthy Animals


The Woman They Could Not Silence
```

In [25]:

```
# import web grabbing client and
# HTML parser
from urllib.request import urlopen as uReq
from bs4 import BeautifulSoup as soup

# variale to store website link as string
myurl = 'http://books.toscrape.com/index.html'

# grab website and store in variable uclient
uClient = uReq(myurl)

# read and close HTML
page_html = uClient.read()
uClient.close()

# call BeautifulSoup for parsing
page_soup = soup(page_html, "html.parser")

# grabs all the products under list tag
bookshelf = page_soup.findAll(
    "li", {"class": "col-xs-6 col-sm-4 col-md-3 col-lg-3"})

# create csv file of all products
filename = ("Books.csv")
f = open(filename, "w")

headers = "Book title, Price\n"
f.write(headers)

for books in bookshelf:

    # collect title of all books
    book_title = books.h3.a["title"]

    # collect book price of all books
    book_price = books.findAll("p", {"class": "price_color"})
    price = book_price[0].text.strip()

    print("Title of the book :" + book_title)
    print("Price of the book :" + price)

    f.write(book_title + "," + price+"\n")

f.close()
```

```
Title of the book :A Light in the Attic
Price of the book :£51.77
Title of the book :Tipping the Velvet
Price of the book :£53.74
Title of the book :Soumission
Price of the book :£50.10
Title of the book :Sharp Objects
Price of the book :£47.82
Title of the book :Sapiens: A Brief History of Humankind
Price of the book :£54.23
Title of the book :The Requiem Red
```

```
Price of the book :£22.65
Title of the book :The Dirty Little Secrets of Getting Your Dream Job
Price of the book :£33.34
Title of the book :The Coming Woman: A Novel Based on the Life of the Infam
ous Feminist, Victoria Woodhull
Price of the book :£17.93
Title of the book :The Boys in the Boat: Nine Americans and Their Epic Ques
t for Gold at the 1936 Berlin Olympics
Price of the book :£22.60
Title of the book :The Black Maria
Price of the book :£52.15
Title of the book :Starving Hearts (Triangular Trade Trilogy, #1)
Price of the book :£13.99
Title of the book :Shakespeare's Sonnets
Price of the book :£20.66
Title of the book :Set Me Free
Price of the book :£17.46
Title of the book :Scott Pilgrim's Precious Little Life (Scott Pilgrim #1)
Price of the book :£52.29
Title of the book :Rip it Up and Start Again
Price of the book :£35.02
Title of the book :Our Band Could Be Your Life: Scenes from the American In
die Underground, 1981-1991
Price of the book :£57.25
Title of the book :Olio
Price of the book :£23.88
Title of the book :Mesaerion: The Best Science Fiction Stories 1800-1849
Price of the book :£37.59
Title of the book :Libertarianism for Beginners
Price of the book :£51.33
Title of the book :It's Only the Himalayas
Price of the book :£45.17
```

In [26]:
```
## Request page source from URL
```

In [27]:
```
url = "https://www.icc-cricket.com/rankings/mens/player-rankings/odi/batting"
page = requests.get(url)
page
```

Out[27]: <Response [200]>

In [28]:
```
## display the page source code
page.content

soup = BeautifulSoup(page.content,"html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" name="twitter:title"/>
  <meta content="website" property="og:type"/>
  <meta content="summary_large_image" property="twitter:card"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="description"/>
  <meta content="@icc" property="twitter:site"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="twitter:description"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
fault-thumbnail.jpg" name="twitter:image"/>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" property="og:title"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
```

In [29]:
```
#scrap men names
scraped_mens = soup.find_all('td', class_='table-body__cell rankings-table__n
scraped_mens
```

Out[29]:
```
[<td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/164">Virat Kohli</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/107">Rohit Sharma</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/226">Ross Taylor</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/167">Aaron Finch</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/506">Jonny Bairstow</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/3801">Fakhar Zaman</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
```

In [30]:
```
# parse men names
mens = []
for men in scraped_mens:
    men = men.get_text().replace('\n',"")
    men = men.strip(" ")
    mens.append(men)
mens
```

Out[30]:
```
['Virat Kohli',
 'Rohit Sharma',
 'Ross Taylor',
 'Aaron Finch',
 'Jonny Bairstow',
 'Fakhar Zaman',
 'Francois du Plessis',
 'David Warner',
 'Shai Hope',
 'Quinton de Kock',
 'Kane Williamson',
 'Imam-ul-Haq',
 'Mushfiqur Rahim',
 'Joe Root',
 'Steve Smith',
 'Martin Guptill',
 'Shikhar Dhawan',
 'Jason Roy',
 'Paul Stirling',
 'Glenn Maxwell',
 'Rassie van der Dussen',
 'Usman Khawaja',
 'Ben Stokes',
 'Tamim Iqbal',
 'Eoin Morgan',
 'Lokesh Rahul',
 'Alex Carey',
 'David Miller',
 'Shimron Hetmyer',
 'Shakib Al Hasan',
 'Jos Buttler',
 'Tom Latham',
 'Nicholas Pooran',
 'Haris Sohail',
 'Mahmudullah',
 'Henry Nicholls',
 'Aqib Ilyas',
 'Kyle Coetzer',
 'Brendan Taylor',
 'Angelo Mathews',
 'Kusal Perera',
 'Hardik Pandya',
 'Sean Williams',
 'Kedar Jadhav',
 'Evin Lewis',
 'Rahmat Shah',
 'Danushka Gunathilaka',
```

```
'Mohammad Hafeez',
'Kusal Mendis',
'Andrew Balbirnie',
'Sikandar Raza',
'Litton Das',
'Imad Wasim',
'Soumya Sarkar',
'Marcus Stoinis',
'Mohammad Shahzad',
'Niroshan Dickwella',
'Sarfaraz Ahmed',
'Calum MacLeod',
'Colin de Grandhomme',
'Najibullah Zadran',
'Marnus Labuschagne',
'Chris Gayle',
'Hashmatullah Shaidi',
'Avishka Fernando',
'Jimmy Neesham',
'Shaun Marsh',
'Mohammad Nabi',
'Mitchell Marsh',
"Kevin O'Brien",
'Peter Handscomb',
'Shreyas Iyer',
'Asghar Afghan',
'Richard Berrington',
'Muhammad Usman',
'Colin Munro',
'Craig Ervine',
'Mitchell Santner',
'William Porterfield',
'Heinrich Klaasen',
'Aiden Markram',
'Andile Phehlukwayo',
'Lahiru Thirimanne',
'Dhananjaya de Silva',
'Jason Holder',
'Temba Bavuma',
'Assad Vala',
'Ravindra Jadeja',
'Sabbir Rahman',
'Mithun Ali',
'Rishabh Pant',
'Rashid Khan',
'George Munsey',
'Dinesh Chandimal',
'Rovman Powell',
'Tony Ura',
'Gulbadin Naib',
'Chris Woakes',
'Sam Billings']
```

In [31]:
```
# scrap rating for mens
scraped_ratings = soup.find_all('td',class_="table-body__cell rating")
scraped_ratings
# parse ratings
ratings = []
for rating in scraped_ratings:
    rating = rating.get_text().replace('\n','')
    ratings.append(rating)
ratings
```

Out[31]:
```
['857',
 '825',
 '801',
 '791',
 '785',
 '778',
 '778',
 '773',
 '773',
 '756',
 '754',
 '751',
 '734',
 '723',
 '707',
 '707',
 '706',
 '689',
 '687',
 '684',
 '680',
 '675',
 '668',
 '667',
 '655',
 '646',
 '644',
 '640',
 '633',
 '632',
 '627',
 '624',
 '614',
 '603',
 '582',
 '581',
 '578',
 '573',
 '571',
 '567',
 '566',
 '561',
 '561',
 '560',
 '559',
```

```
 '559',
 '554',
 '546',
 '542',
 '537',
 '536',
 '535',
 '533',
 '529',
 '525',
 '519',
 '518',
 '512',
 '508',
 '507',
 '506',
 '506',
 '504',
 '498',
 '495',
 '495',
 '495',
 '494',
 '486',
 '484',
 '483',
 '479',
 '479',
 '479',
 '471',
 '468',
 '464',
 '460',
 '460',
 '457',
 '454',
 '450',
 '449',
 '448',
 '445',
 '444',
 '443',
 '440',
 '437',
 '433',
 '432',
 '432',
 '430',
 '429',
 '426',
 '415',
 '415',
 '411',
 '409']
```

In [32]:
```
# scrap team for mens
scraped_teams = soup.find_all('span',class_="table-body__logo-text")
scraped_teams
# parse teams
teams = []
for team in scraped_teams:
    team = team.get_text().replace('\n','')
    teams.append(team)
teams
```

Out[32]:
```
['IND',
 'IND',
 'NZ',
 'AUS',
 'ENG',
 'PAK',
 'SA',
 'AUS',
 'WI',
 'SA',
 'NZ',
 'PAK',
 'BAN',
 'ENG',
 'AUS',
 'NZ',
 'IND',
 'ENG',
 'IRE',
 'AUS',
 'SA',
 'AUS',
 'ENG',
 'BAN',
 'ENG',
 'IND',
 'AUS',
 'SA',
 'WI',
 'BAN',
 'ENG',
 'NZ',
 'WI',
 'PAK',
 'BAN',
 'NZ',
 'OMA',
 'SCO',
 'ZIM',
 'SL',
 'SL',
 'IND',
 'ZIM',
 'IND',
 'WI',
```

```
'AFG',
'SL',
'PAK',
'SL',
'IRE',
'ZIM',
'BAN',
'PAK',
'BAN',
'AUS',
'AFG',
'SL',
'PAK',
'SCO',
'NZ',
'AFG',
'AUS',
'WI',
'AFG',
'SL',
'NZ',
'AUS',
'AFG',
'AUS',
'IRE',
'AUS',
'IND',
'AFG',
'SCO',
'UAE',
'NZ',
'ZIM',
'NZ',
'IRE',
'SA',
'SA',
'SA',
'SL',
'SL',
'WI',
'SA',
'PNG',
'IND',
'BAN',
'BAN',
'IND',
'AFG',
'SCO',
'SL',
'WI',
'PNG',
'AFG',
'ENG',
'ENG']
```

In [33]:
```
import pandas as pd
```

In [34]:
```
## store the scraped Data
data = pd.DataFrame()
data['Mens'] = mens
data['Ratings'] = ratings
data['Teams'] = teams
data.head()
```

Out[34]:

	Mens	Ratings	Teams
0	Virat Kohli	857	IND
1	Rohit Sharma	825	IND
2	Ross Taylor	801	NZ
3	Aaron Finch	791	AUS
4	Jonny Bairstow	785	ENG

In [35]:
```
## Request page source from URL
```

In [36]:
```
url = "https://www.icc-cricket.com/rankings/mens/player-rankings/odi/bowling"
page = requests.get(url)
page
```

Out[36]:
```
<Response [200]>
```

In [37]:
```
## display the page source code
page.content

soup = BeautifulSoup(page.content,"html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" name="twitter:title"/>
  <meta content="website" property="og:type"/>
  <meta content="summary_large_image" property="twitter:card"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="description"/>
  <meta content="@icc" property="twitter:site"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="twitter:description"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
fault-thumbnail.jpg" name="twitter:image"/>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" property="og:title"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
```

In [42]:
```
#scrap bowler names
scraped_bowlers = soup.find_all('td', class_='table-body__cell rankings-table
scraped_bowlers
```

Out[42]:
```
[<td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/1597">Mehedi Hasan</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/4572">Mujeeb Ur Rahman</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/1505">Matt Henry</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/1124">Jasprit Bumrah</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/1664">Kagiso Rabada</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/mens/player-rankings/967">Chris Woakes</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
```

In [55]:
```
# parse bowler names
bowlers = []
for bowler in scraped_bowlers:
    bowler = bowler.get_text().replace('\n',"")
    bowler = bowler.strip(" ")
    bowlers.append(bowler)
bowlers
```

Out[55]:
```
['Mehedi Hasan',
 'Mujeeb Ur Rahman',
 'Matt Henry',
 'Jasprit Bumrah',
 'Kagiso Rabada',
 'Chris Woakes',
 'Josh Hazlewood',
 'Pat Cummins',
 'Mustafizur Rahman',
 'Mohammad Amir',
 'Shaheen Afridi',
 'Bhuvneshwar Kumar',
 'Jofra Archer',
 'Rashid Khan',
 'Adam Zampa',
 'Lachlan Ferguson',
 'Shakib Al Hasan',
 'Mitchell Starc',
 'Lungi Ngidi',
 'Mohammad Nabi',
 'Mark Wood',
 'Akila Dananjaya',
 'Yuzvendra Chahal',
 'Mitchell Santner',
 'Mohammad Shami',
 'Alzarri Joseph',
 'Kuldeep Yadav',
 'Ravindra Jadeja',
 'Sheldon Cottrell',
 'Liam Plunkett',
 'Dushmantha Chameera',
 'Andy McBrine',
 'Ahmed Raza',
 'Adil Rashid',
 'Shadab Khan',
 'Dwaine Pretorius',
 'Colin de Grandhomme',
 'Andile Phehlukwayo',
 'Tim Southee',
 'Ish Sodhi',
 'Imad Wasim',
 'Mark Watt',
 'Tendai Chatara',
 'Usman Khan',
 'Jhye Richardson',
 'Moeen Ali',
 'Rohan Mustafa',
```

```
'Mashrafe Mortaza',
'David Willey',
'Simi Singh',
'Wanindu De Silva',
'Mohammad Mohammad Saifuddin',
'Hasan Ali',
'Nuwan Pradeep',
'Tabraiz Shamsi',
'Nathan Coulter-Nile',
'Faheem Ashraf',
'Safyaan Sharif',
'Jason Holder',
'Chris Morris',
'Nathan Lyon',
'Jimmy Neesham',
'Kyle Jarvis',
'Suranga Lakmal',
'Wahab Riaz',
'Junaid Khan',
'Sikandar Raza',
'Ben Stokes',
'Kemar Roach',
'Dawlat Zadran',
'Josh Davey',
'Sean Williams',
'Anrich Nortje',
'Kane Richardson',
'Aftab Alam',
'Yasir Shah',
'Shardul Thakur',
'Jason Behrendorff',
'Dhananjaya de Silva',
'Rubel Hossain',
'Hardik Pandya',
'Craig Young',
'Gulbadin Naib',
'Barry McCarthy',
'Mitchell Marsh',
'Alasdair Evans',
'Tom Curran',
'Taskin Ahmed',
'Ashley Nurse',
'Hamid Hassan',
'Blessing Muzarabani',
'Marcus Stoinis',
'Lakshan Sandakan',
'Taijul Islam',
'Ashton Agar',
'Mohammad Nawaz',
'Donald Tiripano',
'George Dockrell',
'Bradley Wheal']
```

In [51]:
```
# scrap rating for bowlers
scraped_ratings = soup.find_all('td',class_="table-body__cell rating")
scraped_ratings
# parse ratings
ratings = []
for rating in scraped_ratings:
    rating = rating.get_text().replace('\n','')
    ratings.append(rating)
ratings
```

Out[51]:
```
['713',
 '708',
 '691',
 '690',
 '666',
 '665',
 '660',
 '646',
 '645',
 '638',
 '634',
 '632',
 '627',
 '626',
 '623',
 '619',
 '613',
 '611',
 '606',
 '596',
 '594',
 '594',
 '584',
 '583',
 '574',
 '569',
 '564',
 '559',
 '558',
 '546',
 '533',
 '533',
 '531',
 '529',
 '523',
 '515',
 '508',
 '506',
 '500',
 '496',
 '495',
 '490',
 '490',
 '489',
 '488',
```

```
'486',
'477',
'476',
'475',
'474',
'473',
'473',
'473',
'473',
'470',
'470',
'469',
'467',
'465',
'462',
'462',
'458',
'458',
'456',
'454',
'449',
'443',
'442',
'431',
'427',
'427',
'426',
'425',
'422',
'422',
'418',
'411',
'410',
'410',
'406',
'403',
'403',
'399',
'397',
'395',
'394',
'391',
'391',
'388',
'387',
'387',
'383',
'381',
'379',
'377',
'376',
'373',
'370',
'368']
```

In [52]:
```
# scrap team for bowlers
scraped_teams = soup.find_all('span',class_="table-body__logo-text")
scraped_teams
# parse teams
teams = []
for team in scraped_teams:
    team = team.get_text().replace('\n','')
    teams.append(team)
teams
```

Out[52]:
```
['BAN',
 'AFG',
 'NZ',
 'IND',
 'SA',
 'ENG',
 'AUS',
 'AUS',
 'BAN',
 'PAK',
 'PAK',
 'IND',
 'ENG',
 'AFG',
 'AUS',
 'NZ',
 'BAN',
 'AUS',
 'SA',
 'AFG',
 'ENG',
 'SL',
 'IND',
 'NZ',
 'IND',
 'WI',
 'IND',
 'IND',
 'WI',
 'ENG',
 'SL',
 'IRE',
 'UAE',
 'ENG',
 'PAK',
 'SA',
 'NZ',
 'SA',
 'NZ',
 '',
 'PAK',
 'SCO',
 'ZIM',
 'PAK',
 'AUS',
```

```
'ENG',
'UAE',
'BAN',
'ENG',
'IRE',
'SL',
'BAN',
'PAK',
'SL',
'SA',
'AUS',
'PAK',
'SCO',
'WI',
'SA',
'AUS',
'NZ',
'ZIM',
'SL',
'PAK',
'PAK',
'ZIM',
'ENG',
'WI',
'AFG',
'SCO',
'ZIM',
'SA',
'AUS',
'AFG',
'PAK',
'IND',
'AUS',
'SL',
'BAN',
'IND',
'IRE',
'AFG',
'IRE',
'AUS',
'SCO',
'ENG',
'BAN',
'WI',
'AFG',
'ZIM',
'AUS',
'SL',
'BAN',
'AUS',
'PAK',
'ZIM',
'IRE',
'SCO']
```

In [ ]:
```
import pandas as pd
```

In [57]:
```
## store the scraped Data
data = pd.DataFrame()
data['Bowlers'] = bowler
data['Ratings'] = ratings
data['Teams'] = teams
data.head()
```

Out[57]:

	Bowlers	Ratings	Teams
0	NaN	713	BAN
1	NaN	708	AFG
2	NaN	691	NZ
3	NaN	690	IND
4	NaN	666	SA

In [58]:
```
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [59]:
```
## Request page source from URL
```

In [20]:
```
url = "https://www.icc-cricket.com/rankings/womens/player-rankings/odi/all-ro
page = requests.get(url)
page
```

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

In [61]:
```
## display the page source code
page.content
soup = BeautifulSoup(page.content,"html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta content="ICC Ranking for ODI teams International Cricket Counci
l" name="twitter:title"/>
  <meta content="website" property="og:type"/>
  <meta content="summary_large_image" property="twitter:card"/>
  <meta content="Official International Cricket Council rankings for tes
t match cricket teams. Discover latest ICC rankings table, predict upcom
ing matches, see points and ratings for all teams." name="description"/>
  <meta content="@icc" property="twitter:site"/>
  <meta content="Official International Cricket Council rankings for tes
t match cricket teams. Discover latest ICC rankings table, predict upcom
ing matches, see points and ratings for all teams." name="twitter:descri
ption"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
fault-thumbnail.jpg" name="twitter:image"/>
  <meta content="ICC Ranking for ODI teams International Cricket Counci
l" property="og:title"/>
```

In [24]:
```
#scrap women names
scraped_womens = soup.find_all('td', class_='table-body__cell rankings-table_
scraped_womens
```

Out[24]: []

In [25]:
```
# parse women names
womens = []
for women in scraped_womens:
    women = match.get_text().replace('\n',"")
    women = match.strip(" ")
    womens.append(women)
womens
```

Out[25]: []

In [70]:
```
# scrap rating for matches
scraped_ratings = soup.find_all('td',class_="table-body__cell u-text-right ra
scraped_ratings
# parse ratings
ratings = []
for rating in scraped_ratings:
    rating = rating.get_text().replace('\n','')
    ratings.append(rating)
ratings
```

Out[70]: ['118', '117', '111', '93', '85', '73', '61', '47', '13']

In [29]:
```
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [30]:
```
## Request page source from URL
```

In [31]:
```
url = "https://www.icc-cricket.com/rankings/womens/player-rankings/odi/all-ro
page = requests.get(url)
page
## display teh page source code
page.content
soup = BeautifulSoup(page.content,"html.parser")
print(soup.prettify())
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" name="twitter:title"/>
  <meta content="website" property="og:type"/>
  <meta content="summary_large_image" property="twitter:card"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="description"/>
  <meta content="@icc" property="twitter:site"/>
  <meta content="Official ICC Cricket website - live matches, scores, ne
ws, highlights, commentary, rankings, videos and fixtures from the Inter
national Cricket Council." name="twitter:description"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
fault-thumbnail.jpg" name="twitter:image"/>
  <meta content="Live Cricket Scores &amp;amp; News International Cricket Co
uncil" property="og:title"/>
  <meta content="https://www.icc-cricket.com/resources/ver/i/elements/de
fault-thumbnail.jpg" property="og:image"/>
```

In [32]:
```
#scrap women names
scraped_womens = soup.find_all('td', class_='table-body__cell rankings-table_
scraped_womens
```

Out[32]:
```
[<td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/471">Ellyse Perry</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/573">Stafanie Taylor</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/1800">Natalie Sciver</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/3192">Deepti Sharma</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/468">Jess Jonassen</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/4128">Ashleigh Gardner</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/430">Dane van Niekerk</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/478">Sophie Devine</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/3756">Amelia Kerr</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/547">Katherine Brunt</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/1813">Shikha Pandey</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/562">Jhulan Goswami</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/552">Heather Knight</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/1830">Rumana Ahmed</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/3126">Hayley Matthews</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/559">Harmanpreet Kaur</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
 <a href="/rankings/womens/player-rankings/600">Chamari Athapaththu</a>
 </td>,
 <td class="table-body__cell rankings-table__name name">
```

```
<a href="/rankings/womens/player-rankings/428">Sune Luus</a>
</td>,
<td class="table-body__cell rankings-table__name name">
<a href="/rankings/womens/player-rankings/1814">Poonam Yadav</a>
</td>]
```

In [33]:

```
# parse women names
womens = []
for women in scraped_womens:
    women = women.get_text().replace('\n',"")
    women = women.strip(" ")
    womens.append(women)
womens
```

Out[33]:

```
['Ellyse Perry',
 'Stafanie Taylor',
 'Natalie Sciver',
 'Deepti Sharma',
 'Jess Jonassen',
 'Ashleigh Gardner',
 'Dane van Niekerk',
 'Sophie Devine',
 'Amelia Kerr',
 'Katherine Brunt',
 'Shikha Pandey',
 'Jhulan Goswami',
 'Heather Knight',
 'Rumana Ahmed',
 'Hayley Matthews',
 'Harmanpreet Kaur',
 'Chamari Athapaththu',
 'Sune Luus',
 'Poonam Yadav']
```

In [34]:
```
# scrap rating for womens
scraped_ratings = soup.find_all('td',class_="table-body__cell rating")
scraped_ratings
# parse ratings
ratings = []
for rating in scraped_ratings:
    rating = rating.get_text().replace('\n','')
    ratings.append(rating)
ratings
```

Out[34]:
```
['418',
 '410',
 '349',
 '343',
 '307',
 '252',
 '243',
 '242',
 '236',
 '236',
 '204',
 '200',
 '189',
 '179',
 '176',
 '167',
 '164',
 '153',
 '151']
```

In [35]:
```
# scrap team for womens
scraped_teams = soup.find_all('span',class_="table-body__logo-text")
scraped_teams
# parse teams
teams = []
for team in scraped_teams:
    team = team.get_text().replace('\n','')
    teams.append(team)
teams
```

Out[35]:
```
['AUS',
 'WI',
 'ENG',
 'IND',
 'AUS',
 'AUS',
 'SA',
 'NZ',
 'NZ',
 'ENG',
 'IND',
 'IND',
 'ENG',
 'BAN',
 'WI',
 'IND',
 'SL',
 'SA',
 'IND']
```

In [36]:
```
## store the scraped Data
data = pd.DataFrame()
data['Womens'] = womens
data['Ratings'] = ratings
data['Teams'] = teams
data.head()
```

Out[36]:

	Womens	Ratings	Teams
0	Ellyse Perry	418	AUS
1	Stafanie Taylor	410	WI
2	Natalie Sciver	349	ENG
3	Deepti Sharma	343	IND
4	Jess Jonassen	307	AUS

In [37]:
```
data.tail()
```

Out[37]:

	Womens	Ratings	Teams
14	Hayley Matthews	176	WI
15	Harmanpreet Kaur	167	IND
16	Chamari Athapaththu	164	SL
17	Sune Luus	153	SA
18	Poonam Yadav	151	IND

In [38]:
```
from bs4 import BeautifulSoup
import requests
import pandas as pd
```

In [39]:
```
HEADERS = ({'User-Agent':
            'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
            'Accept-Language': 'en-US, en;q=0.5'})
```

In [40]:
```
URL = "https://www.amazon.in/s?k=mobile+phone+under+20000&crid=3R49O4WKN9ABY&
Webpage = requests.get(URL, headers=HEADERS)
```

In [42]:
```
soup = BeautifulSoup(Webpage.content, "lxml")
```

In [44]:

```
from bs4 import BeautifulSoup
import requests

# Function to extract Product Title
def get_title(soup):

    try:
        # Outer Tag Object
        title = soup.find("span", attrs={"id":'productTitle'})

        # Inner NavigableString Object
        title_value = title.string

        # Title as a string value
        title_string = title_value.strip()

        # # Printing types of values for efficient understanding
        # print(type(title))
        # print(type(title_value))
        # print(type(title_string))
        # print()

    except AttributeError:
        title_string = ""

    return title_string

# Function to extract Product Price
def get_price(soup):

    try:
        price = soup.find("span", attrs={'id':'a-price-whole'}).string.strip(

    except AttributeError:
        price = ""

    return price

# Function to extract Product Rating
def get_rating(soup):

    try:
        rating = soup.find("i", attrs={'class':'a-icon a-icon-star a-star-4-5

    except AttributeError:

        try:
            rating = soup.find("span", attrs={'class':'a-icon-alt'}).string.s
        except:
            rating = ""

    return rating

# Function to extract Number of User Reviews
def get_review_count(soup):
    try:
```

```
        review_count = soup.find("span", attrs={'id':'acrCustomerReviewText'}

    except AttributeError:
        review_count = ""

    return review_count

# Function to extract  Availability Status
def get_availability(soup):
    try:
        available = soup.find("div", attrs={'id':'availability'})
        available = available.find("span").string.strip()

    except AttributeError:
        available = ""

    return available

if __name__ == '__main__':

    # Headers for request
    HEADERS = ({'User-Agent':
                'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, l
                'Accept-Language': 'en-US, en;q=0.5'})

    # The webpage URL
    URL = "https://www.amazon.in/Green-Storage-Additional-Exchange-Offers/dp/

    # HTTP Request
    webpage = requests.get(URL, headers=HEADERS)

    # Soup Object containing all data
    soup = BeautifulSoup(webpage.content, "lxml")

    # Function calls to display all necessary product information
    print("Product Title =", get_title(soup))
    print("Product Price =", get_price(soup))
    print("Product Rating =", get_rating(soup))
    print("Number of Product Reviews =", get_review_count(soup))
    print("Availability =", get_availability(soup))
    print()
    print()
```

```
Product Title = OPPO A31 (Lake Green, 4GB RAM, 64GB Storage) with No Cost E
MI/Additional Exchange Offers
Product Price =
Product Rating = 4.2 out of 5 stars
Number of Product Reviews = 17,341 ratings
Availability = In stock.
```

In [45]:

```
from bs4 import BeautifulSoup
import requests

# Function to extract Product Title
def get_title(soup):

    try:
        # Outer Tag Object
        title = soup.find("span", attrs={"id":'productTitle'})

        # Inner NavigatableString Object
        title_value = title.string

        # Title as a string value
        title_string = title_value.strip()

        # # Printing types of values for efficient understanding
        # print(type(title))
        # print(type(title_value))
        # print(type(title_string))
        # print()

    except AttributeError:
        title_string = ""

    return title_string

# Function to extract Product Price
def get_price(soup):

    try:
        price = soup.find("span", attrs={'id':'priceblock_dealprice'}).string

    except AttributeError:

        try:
            # If there is some deal price
            price = soup.find("span", attrs={'id':'priceblock_dealprice'}).st

        except:
            price = ""

    return price

# Function to extract Product Rating
def get_rating(soup):

    try:
        rating = soup.find("i", attrs={'class':'a-icon a-icon-star a-star-4-5

    except AttributeError:

        try:
            rating = soup.find("span", attrs={'class':'a-icon-alt'}).string.s
        except:
            rating = ""
```

```
    return rating

# Function to extract Number of User Reviews
def get_review_count(soup):
    try:
        review_count = soup.find("span", attrs={'id':'acrCustomerReviewText'}

    except AttributeError:
        review_count = ""

    return review_count

# Function to extract Availability Status
def get_availability(soup):
    try:
        available = soup.find("div", attrs={'id':'availability'})
        available = available.find("span").string.strip()

    except AttributeError:
        available = "Not Available"

    return available


if __name__ == '__main__':

    # Headers for request
    HEADERS = ({'User-Agent':
                'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, l
                'Accept-Language': 'en-US'})

    # The webpage URL
    URL ="https://www.amazon.in/s?k=mobile+phone+under+20000&crid=2JJIQQMZ8VI

    # HTTP Request
    webpage = requests.get(URL, headers=HEADERS)

    # Soup Object containing all data
    soup = BeautifulSoup(webpage.content, "lxml")

    # Fetch links as List of Tag Objects
    links = soup.find_all("a", attrs={'class':'a-link-normal sg-col-inner'})

    # Store the links
    links_list = []

    # Loop for extracting links from Tag Objects
    for link in links:
        links_list.append(link.get('href'))


    # Loop for extracting product details from each link
    for link in links_list:

        new_webpage = requests.get("https://www.amazon.com" + link, headers=H
```

```
        new_soup = BeautifulSoup(new_webpage.content, "lxml")

        # Function calls to display all necessary product information
        print("Product Title =", get_title(new_soup))
        print("Product Price =", get_price(new_soup))
        print("Product Rating =", get_rating(new_soup))
        print("Number of Product Reviews =", get_review_count(new_soup))
        print("Availability =", get_availability(new_soup))
        print()
        print()
```

In [46]:

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

In [47]:

```
page = requests.get("http://forecast.weather.gov/MapClick.php?lat=37.7772&lon
soup = BeautifulSoup(page.content, 'html.parser')
seven_day = soup.find(id="seven-day-forecast")
forecast_items = seven_day.find_all(class_="tombstone-container")
tonight = forecast_items[0]
print(tonight.prettify())
```

```
<div class="tombstone-container">
 <p class="period-name">
  Overnight
  <br/>
  <br/>
 </p>
 <p>
  <img alt="Overnight: Mostly cloudy, with a low around 56. West wind aroun
d 14 mph, with gusts as high as 18 mph. " class="forecast-icon" src="newima
ges/medium/nbkn.png" title="Overnight: Mostly cloudy, with a low around 56.
West wind around 14 mph, with gusts as high as 18 mph. "/>
 </p>
 <p class="short-desc">
  Mostly Cloudy
 </p>
 <p class="temp temp-low">
  Low: 56 °F
 </p>
</div>
```

In [48]:
```
period = tonight.find(class_="period-name").get_text()
short_desc = tonight.find(class_="short-desc").get_text()
temp = tonight.find(class_="temp").get_text()

print(period)
print(short_desc)
print(temp)
```

```
Overnight
Mostly Cloudy
Low: 56 °F
```

In [49]:
```
img = tonight.find("img")
desc = img['title']

print(desc)
```

```
Overnight: Mostly cloudy, with a low around 56. West wind around 14 mph, wi
th gusts as high as 18 mph.
```

In [50]:
```
period_tags = seven_day.select(".tombstone-container .period-name")
periods = [pt.get_text() for pt in period_tags]
periods
```

Out[50]:
```
['Overnight',
 'Friday',
 'FridayNight',
 'Saturday',
 'SaturdayNight',
 'Sunday',
 'SundayNight',
 'Monday',
 'MondayNight']
```

In [51]:
```
short_descs = [sd.get_text() for sd in seven_day.select(".tombstone-container
temps = [t.get_text() for t in seven_day.select(".tombstone-container .temp")
descs = [d["title"] for d in seven_day.select(".tombstone-container img")]
print(short_descs)
print(temps)
print(descs)
```

```
['Mostly Cloudy', 'Mostly Cloudythen Sunnyand Breezy', 'IncreasingClouds an
dBreezy thenCloudy', 'Partly Sunnythen Sunnyand Breezy', 'Mostly Clearand B
reezythen MostlyCloudy', 'Mostly Sunnyand Breezy', 'Partly Cloudyand Breezy
then MostlyCloudy', 'Mostly Sunny', 'Partly Cloudy']
['Low: 56 °F', 'High: 68 °F', 'Low: 57 °F', 'High: 72 °F', 'Low: 58 °F', 'H
igh: 70 °F', 'Low: 58 °F', 'High: 70 °F', 'Low: 58 °F']
['Overnight: Mostly cloudy, with a low around 56. West wind around 14 mph,
with gusts as high as 18 mph. ', 'Friday: Cloudy through mid morning, then
gradual clearing, with a high near 68. Breezy, with a west wind 10 to 15 mp
h increasing to 19 to 24 mph in the afternoon. Winds could gust as high as
32 mph. ', 'Friday Night: Increasing clouds, with a low around 57. Breezy,
with a west wind 21 to 26 mph decreasing to 11 to 16 mph after midnight. Wi
nds could gust as high as 33 mph. ', 'Saturday: Mostly cloudy through mid m
orning, then gradual clearing, with a high near 72. Breezy, with a west sou
thwest wind 7 to 12 mph increasing to 20 to 25 mph in the afternoon. Winds
could gust as high as 32 mph. ', 'Saturday Night: Partly cloudy, with a low
around 58. Breezy, with a west southwest wind 21 to 26 mph decreasing to 11
to 16 mph in the evening. Winds could gust as high as 34 mph. ', 'Sunday: M
ostly sunny, with a high near 70. Breezy. ', 'Sunday Night: Mostly cloudy,
with a low around 58. Breezy. ', 'Monday: Mostly sunny, with a high near 7
0.', 'Monday Night: Partly cloudy, with a low around 58.']
```

In [52]:
```
weather = pd.DataFrame({"period":periods,"short_desc": short_descs,"temp":tem
weather
```

Out[52]:

	period	short_desc	temp	desc
0	Overnight	Mostly Cloudy	Low: 56 °F	Overnight: Mostly cloudy, with a low around 56...
1	Friday	Mostly Cloudythen Sunnyand Breezy	High: 68 °F	Friday: Cloudy through mid morning, then gradu...
2	FridayNight	IncreasingClouds andBreezy thenCloudy	Low: 57 °F	Friday Night: Increasing clouds, with a low ar...
3	Saturday	Partly Sunnythen Sunnyand Breezy	High: 72 °F	Saturday: Mostly cloudy through mid morning, t...
4	SaturdayNight	Mostly Clearand Breezythen MostlyCloudy	Low: 58 °F	Saturday Night: Partly cloudy, with a low arou...
5	Sunday	Mostly Sunnyand Breezy	High: 70 °F	Sunday: Mostly sunny, with a high near 70. Bre...
6	SundayNight	Partly Cloudyand Breezythen MostlyCloudy	Low: 58 °F	Sunday Night: Mostly cloudy, with a low around...
7	Monday	Mostly Sunny	High: 70 °F	Monday: Mostly sunny, with a high near 70.
8	MondayNight	Partly Cloudy	Low: 58 °F	Monday Night: Partly cloudy, with a low around...

In [53]:
```
import requests
URL = "https://internshala.com/fresher-jobs"
page = requests.get(URL)
from bs4 import BeautifulSoup as soup
page.status_code
bsobj = soup(page.content,'lxml')
bsobj
```

Out[53]:
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:fb="https://www.faceboo
k.com/2008/fbml" xmlns:og="http://ogp.me/ns#">
<head>
<meta content="IE=9" http-equiv="X-UA-Compatible"/>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0 user-scalable=0" na
me="viewport"/>
<meta content="272234782795210" property="fb:app_id"/>
<meta content="article" property="og:type"/>
<meta content="1200" property="og:image:width"/>
<meta content="630" property="og:image:height"/>
<meta content="@Internshala" name="twitter:site"/>
<meta content="summary_large_image" name="twitter:card"/>
<meta content="@internshala" name="twitter:creator"/>
<meta content="#ffffff" name="theme-color"/>
<meta content="#ffffff" name="msapplication-navbutton-color"/>
<meta content="telephone=no" name="format-detection"/>
<link as="font" crossorigin="" href="/static/fonts/Inter.woff2?v=3.11" r
```

In [54]:
```
title = []
for header in bsobj.findAll('div',{'class':'heading_4_5 profile'}):
    title.append(header.text.strip())

title
```

Out[54]:
```
['Recruiter',
 'Associate Software Developer',
 'Business Development Executive',
 'Software Developer',
 'Management Trainee - Assistant Category Manager',
 'Account Executive - Field Sales',
 'Software Engineer',
 'Associate Software Developer',
 'Social Media Strategist And Copywriter',
 'Level 1 Technical Support Engineer',
 'Associate Software Developer',
 'Accounting Assistant',
 'Software Engineer',
 'Business Development Manager',
 'Business Development Executive',
 'Product Developer - Mathematics',
 'Sales Development Representative',
 'Full Stack Engineer',
 'Junior ReactJS Developer',
 'Marketing Manager',
 'Product Marketing Associate',
 'Administration Associate',
 'Associate Software Developer',
 'Project Coordinator',
 'Web Developer',
 'Software Engineer Trainee',
 'Business Development Specialist (Sales & Marketing)',
 'Business Development Executive',
 'Associate Front End Developer',
 'Corporate Sales Executive',
 'Associate Software Developer (Full-Stack - React And Node)',
 'Customer Relationship Specialist',
 'Associate Editor Engagement',
 'Management Consultant Associate',
 'Web Analytics Developer',
 'Junior Social Media Marketing Manager',
 'Junior Social Media Marketing Associate',
 'Business Development Manager (Digital Marketing Sales)',
 'Business Development Executive',
 'Accountant']
```

In [55]:
```
company = []
for name in bsobj.findAll('div',{'class':'heading_6 company_name'}):
    company.append(name.text.strip())

company
```

Out[55]:
```
['Radish Consultants Private Limited',
 'Labellerr By Tensor Matics Private Limited',
 'UFaber Edutech',
 'Aptigenz Solutions Private Limited',
 'Noise',
 'RoaDo',
 'CrewKarma',
 'IQGateway',
 'Internshala',
 'Mithi Software Technologies Private Limited',
 'White Blink',
 'Wono Inc',
 'Wono Inc',
 'Wono Inc',
 'Emertxe Information Technologies',
 'Open Door Education',
 'Content Beta',
 'Softway Solutions Private Limited',
 'DeepThought Edutech Ventures Private Limited',
 'Saltoro Coffee Roasters',
 'WebMOBI',
 'Global Sun',
 'Medico Healthcare Services & Technologies',
 'Edupixels',
 'Manufac Analytics Private Limited',
 'Swabhav Techlabs',
 'Claraeon Learning Private Limited',
 'Picostone',
 'AIMonk Labs Technology Limited',
 '369 Zoss Waters',
 'SleekSky LLC',
 'InPhase Power Technologies',
 'Cactus Communications Private Limited',
 'StrategyCo.Global',
 'DataVinci Private Limited',
 'The Test Tribe',
 'Glu Studios',
 'Graygraph Technologies LLC',
 'BookLeaf Publishing',
 'Ravi Ladia & Co']
```

In [56]:
```
import pandas as pd
```

In [57]:
```
data = pd.DataFrame()
data['Title'] = title
data['comapny'] = company
data.head()
```

Out[57]:

	Title	comapny
0	Recruiter	Radish Consultants Private Limited
1	Associate Software Developer	Labellerr By Tensor Matics Private Limited
2	Business Development Executive	UFaber Edutech
3	Software Developer	Aptigenz Solutions Private Limited
4	Management Trainee - Assistant Category Manager	Noise

In [59]:
```
import requests
URL = "https://www.nobroker.in/property/sale/bangalore/Electronic%20City?type
page = requests.get(URL)
from bs4 import BeautifulSoup as soup
page.status_code
bsobj = soup(page.content,'lxml')
bsobj
```

Out[59]:
```
<!DOCTYPE html>
<html lang="en"><head>
<meta content="794951570520699" property="fb:pages"/>
<link href="https://www.nobroker.in" rel="canonical"/>
<link href="//assets.nobroker.in/static/img/favicon.png" id="favicon" re
l="shortcut icon"/>
<link href="https://images.nobroker.in/static/img/fav64.png" rel="apple-
touch-icon"/>
<meta charset="utf-8"/><meta content="text/html; charset=utf-8" http-equ
iv="Content-Type"/>
<meta content="app-id=com.nobroker.app&amp;referrer=utm_source%3Dnobroke
r%26utm_medium%3DmobileWeb" name="google-play-app"/>
<meta content="app-id=1200507100, app-argument=nobrokerapp://" name="app
le-itunes-app"/>
<meta content="#fd3752" name="theme-color"/>
<meta content="4 BHK  flats for sale in Electronic City, 4 BHK  apartmen
ts for sale in Electronic City,  houses for sale in Electronic City" nam
e="keywords"/>
<meta content="4 BHK  Houses, Apartments for Sale in Electronic City, Ba
```

In [60]:
```
title = []
for header in bsobj.findAll('a',{'class':'nb__3CnI6'}):
    title.append(header.text.strip())

title
```

Out[60]:
```
['4 BHK In Independent House  For Sale  In Naganathapura',
 '4 BHK For Sale  In Daadys Garden In Electronic City',
 '4 BHK In Independent House  For Sale  In Sarjapura',
 '4 BHK Flat  For Sale  In Electronic City',
 '4 BHK In Independent House  For Sale  In Electronic City,',
 '4 BHK In Independent House  For Sale  In Electronic City Phase Ii',
 '4 BHK In Independent House  For Sale  In Electronic City',
 '4 BHK Apartment  For Sale  In Gopalan Gardenia In Electronic City',
 '4 BHK Flat  For Sale  In Electronic City',
 '4 BHK Flat  For Sale  In Heena Enclave In Electronic City']
```

In [61]:
```
area = []
for header in bsobj.findAll('div',{'class':'nb__3oNyC'}):
    area.append(header.text.strip())

area
```

Out[61]:
```
['1,500 sqft',
 '2,600 sqft',
 '1,100 sqft',
 '1,400 sqft',
 '2,500 sqft',
 '1,500 sqft',
 '2,400 sqft',
 '2,650 sqft',
 '1,120 sqft',
 '2,350 sqft']
```

In [70]:
```
import pandas as pd
```

In [71]:
```
data = pd.DataFrame()
data['Title'] = title
data['area'] = area
data.head()
```

Out[71]:

	Title	area
0	4 BHK In Independent House For Sale In Nagan...	1,500 sqft
1	4 BHK For Sale In Daadys Garden In Electronic...	2,600 sqft
2	4 BHK In Independent House For Sale In Sarja...	1,100 sqft
3	4 BHK Flat For Sale In Electronic City	1,400 sqft
4	4 BHK In Independent House For Sale In Elect...	2,500 sqft

In [ ]: