1) Write a python program to display all the header tags from wikipedia.org.

In [1]:

```
# importing the required libraries
from bs4 import BeautifulSoup
import requests
```

In [27]:

```
page=requests.get('https://www.wikipedia.org')
```

In [28]:

```
page
```

Out[28]:

```
<Response [200]>
```

In [29]:

```
soup=BeautifulSoup(page.content)
soup
```

```
vg>
</div>
<div class="banner__topbar">
We ask you, humbly, to help.
</div>
<div class="banner__main">
<div class="banner__text">
If everyone reading this donated <span class="banner__amount">$2.75</span
>, we could keep Wikipedia thriving for years.
</div>
<a class="banner__button banner__button--progressive" href="#">
Donate now
</a>
</div>
</div>
<div class="banner" id="portalBanner_enIN_2022a_txt_hiSalutation">
<div class="banner__close" tabindex="0">
<svg aria-labelledby="banner__close-icon" class="banner__close-icon" heigh
t="16" viewbox="0 0 16 16" width="16" xmlns="http://www.w3.org/2000/svg"><
title id="banner__close-icon">Close</title><g fill="none" fill-rule="eveno
```

In [ ]:

```
#class="firstHeading mw-first-heading"
```

In [30]:

```
first_heading=soup.find('div', class_="central-featured-lang lang1")
first_heading
```

Out[30]:

```
<div class="central-featured-lang lang1" dir="ltr" lang="en">
<a class="link-box" data-slogan="The Free Encyclopedia" href="//en.wikipedi
a.org/" id="js-link-box-en" title="English — Wikipedia — The Free Encycloped
ia">
<strong>English</strong>
<small><bdi dir="ltr">6 458 000+</bdi> <span>articles</span></small>
</a>
</div>
```

In [23]:

```
first_heading.text
```

Out[23]:

```
'\n\nEnglish\n6\xa0458\xa0000+ articles\n\n'
```

In [48]:

```
first_heading.text.split()
```

Out[48]:

```
['English', '6', '458', '000+', 'articles']
```

In [25]:

```
header  =[] # empty list for store the

for i in soup.find_all('div',class_="central-featured-lang lang1"):
    header.append(i.text)

header
```

Out[25]:

```
['\n\nEnglish\n6\xa0458\xa0000+ articles\n\n']
```

In [26]:

```
# Making data frame

import pandas as pd
df=pd.DataFrame({'Titles':header})
df
```

Out[26]:

	Titles
0	\n\nEnglish\n6 458 000+ articles\n\n

2) Write a python program to display IMDB's Top rated 100 movies' data (i.e. name, rating, year of release) and make data frame.

In [ ]:

```
```

In [31]:

```
page=requests.get('https://www.imdb.com/chart/top/?ref_=nv_mv_250')
page
```

Out[31]:

```
<Response [200]>
```

In [37]:

```
soup=BeautifulSoup(page.content)
soup
```

```
                var startTimeInt = +new Date();
                var roboto = new FontFace('Roboto',
                    'url(https://m.media-amazon.com/images/G/01/IMDb/cm9ib
3Rv.woff2)',
                    { style:'normal', weight: 400 });
                var robotoMedium = new FontFace('Roboto',
                    'url(https://m.media-amazon.com/images/G/01/IMDb/cm9ib
3RvTWVk.woff2)',
                    { style:'normal', weight: 500 });
                var robotoBold = new FontFace('Roboto',
                    'url(https://m.media-amazon.com/images/G/01/IMDb/cm9ib
3RvQm9sZA.woff2)',
                    { style:'normal', weight: 600 });
                var robotoLoaded = roboto.load();
                var robotoMediumLoaded = robotoMedium.load();
                var robotoBoldLoaded = robotoBold.load();

                win.Promise.all([robotoLoaded, robotoMediumLoaded, robotoB
oldLoaded]).then(function() {
                    var loadTimeInt = +new Date();
```

In [51]:

```
Movie_name=soup.find('td', class_="titleColumn")
Movie_name
```

Out[51]:

```
<td class="titleColumn">
      1.
      <a href="/title/tt0111161/" title="Frank Darabont (dir.), Tim Robbins,
Morgan Freeman">The Shawshank Redemption</a>
<span class="secondaryInfo">(1994)</span>
</td>
```

In [52]:

```
Movie_name.text
```

Out[52]:

```
'\n      1.\n      The Shawshank Redemption\n(1994)\n'
```

In [57]:

```
Movie_name.text.split()
```

Out[57]:

```
['1.', 'The', 'Shawshank', 'Redemption', '(1994)']
```

In [ ]:

```
#class="secondaryInfo"
```

In [59]:

```
year=soup.find('span',class_="secondaryInfo")
year.text
```

Out[59]:

```
'(1994)'
```

In [ ]:

```
#class="ratingColumn imdbRating"
```

In [60]:

```
rating=soup.find('td',class_="ratingColumn imdbRating")
rating.text
```

Out[60]:

```
'\n9.2\n'
```

In [64]:

```
rating.text.split()
```

Out[64]:

```
['9.2']
```

In [86]:

```
Movie_name= []  # empty list

for i in soup.find_all('td',class_="titleColumn"):
    Movie_name.append(i.text)
Movie_name
```

Out[86]:

```
[]
```

In [66]:

```
year= []  # empty list

for i in soup.find_all('span',class_="secondaryInfo"):
    year.append(i.text)
year
```

Out[66]:

```
['(1994)',
 '(1972)',
 '(2008)',
 '(1974)',
 '(1957)',
 '(1993)',
 '(2003)',
 '(1994)',
 '(2001)',
 '(1966)',
 '(1994)',
 '(1999)',
 '(2010)',
 '(2002)',
 '(1980)',
 '(1999)',
 '(1990)',
 '(1975)',
```

In [68]:

```
rating= []  # empty list

for i in soup.find_all('td',class_="ratingColumn imdbRating"):
    rating.append(i.text)
rating
```

Out[68]:

```
['\n9.2\n',
 '\n9.2\n',
 '\n9.0\n',
 '\n9.0\n',
 '\n8.9\n',
 '\n8.9\n',
 '\n8.9\n',
 '\n8.9\n',
 '\n8.8\n',
 '\n8.8\n',
 '\n8.8\n',
 '\n8.7\n',
 '\n8.7\n',
 '\n8.7\n',
 '\n8.7\n',
 '\n8.7\n',
 '\n8.7\n',
 '\n8.6\n',
```

In [74]:

```
import pandas as pd
df=pd.DataFrame({'MOVIE NAME':Movie_name,'Rating':rating,'YEAR OF RELEASE':year,})
df
```

Out[74]:

	MOVIE NAME	Rating	YEAR OF RELEASE
0	\n 1.\n The Shawshank Redemption\n(1...	\n9.2\n	(1994)
1	\n 2.\n The Godfather\n(1972)\n	\n9.2\n	(1972)
2	\n 3.\n The Dark Knight\n(2008)\n	\n9.0\n	(2008)
3	\n 4.\n The Godfather Part II\n(1974)\n	\n9.0\n	(1974)
4	\n 5.\n 12 Angry Men\n(1957)\n	\n8.9\n	(1957)
...	...	...	...
245	\n 246.\n Aladdin\n(1992)\n	\n8.0\n	(1992)
246	\n 247.\n Gandhi\n(1982)\n	\n8.0\n	(1982)
247	\n 248.\n Jai Bhim\n(2021)\n	\n8.0\n	(2021)
248	\n 249.\n The Help\n(2011)\n	\n8.0\n	(2011)
249	\n 250.\n Beauty and the Beast\n(199...	\n8.0\n	(1991)

250 rows × 3 columns

3. Write a python program to display IMDB's Top rated

100 Indian movies' data (i.e. name, rating, year of release) and make data frame.

In [76]:

```
page=requests.get('https://www.imdb.com/list/ls009997493/')
page
```

Out[76]:

```
<Response [200]>
```

In [77]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[77]:

```
<!DOCTYPE html>
<html xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="http://ogp.m
e/ns#">
<head>
<meta charset="utf-8"/>
<meta content="IE=edge" http-equiv="X-UA-Compatible"/>
<script type="text/javascript">var IMDbTimer={starttime: new Date().getTim
e(),pt:'java'};</script>
<script>
    if (typeof uet == 'function') {
      uet("bb", "LoadTitle", {wb: 1});
    }
</script>
<script>(function(t){ (t.events = t.events || {})["csm_head_pre_title"] =
new Date().getTime(); })(IMDbTimer);</script>
<title>IMDB Top 100 Hindi Movies - IMDb</title>
<script>(function(t){ (t.events = t.events || {})["csm_head_post_title"] =
new Date().getTime(); })(IMDbTimer);</script>
```

In [79]:

```
Movie_name=soup.find('h3', class_="lister-item-header")
Movie_name
```

Out[79]:

```
<h3 class="lister-item-header">
<span class="lister-item-index unbold text-primary">1.</span>
<a href="/title/tt0405508/">Rang De Basanti</a>
<span class="lister-item-year text-muted unbold">(2006)</span>
</h3>
```

In [80]:

```
Movie_name.text
```

Out[80]:

```
'\n1.\nRang De Basanti\n(2006)\n'
```

In [82]:

```
year =soup.find('span',class_="lister-item-year text-muted unbold")
year.text
```

Out[82]:

```
'(2006)'
```

In [ ]:

```
#class="ipl-rating-star__rating"
```

In [100]:

```
rating =soup.find('div',class_="ipl-rating-star small")
rating.text
```

Out[100]:

```
'\n\n\n\n\n\n\n\n8.1\n'
```

In [87]:

```
Movie_name= []  # empty list

for i in soup.find_all('h3', class_="lister-item-header"):
    Movie_name.append(i.text)
Movie_name
```

Out[87]:

```
['\n1.\nRang De Basanti\n(2006)\n',
 '\n2.\n3 Idiots\n(2009)\n',
 '\n3.\nTaare Zameen Par\n(2007)\n',
 '\n4.\nDil Chahta Hai\n(2001)\n',
 '\n5.\nSwades: We, the People\n(2004)\n',
 '\n6.\nLagaan: Once Upon a Time in India\n(2001)\n',
 '\n7.\nGangs of Wasseypur\n(2012)\n',
 '\n8.\nBarfi!\n(2012)\n',
 '\n9.\nAnand\n(1971)\n',
 '\n10.\nMunna Bhai M.B.B.S.\n(2003)\n',
 '\n11.\nA Wednesday\n(2008)\n',
 '\n12.\nAndaz Apna Apna\n(1994)\n',
 '\n13.\nSholay\n(1975)\n',
 '\n14.\nBhaag Milkha Bhaag\n(2013)\n',
 '\n15.\nHera Pheri\n(2000)\n',
 '\n16.\nUdaan\n(2010)\n',
 '\n17.\nKahaani\n(2012)\n',
 '\n18.\nBlack\n(2005)\n',
 '\n19.\nChak De! India\n(2007)\n',
 '\n20.\nKhosla Ka Ghosla!\n(2006)\n',
 '\n21.\nJo Jeeta Wohi Sikandar\n(1992)\n',
 '\n22.\nZindagi Na Milegi Dobara\n(2011)\n',
 '\n23.\nPaan Singh Tomar\n(2012)\n',
 '\n24.\nDilwale Dulhania Le Jayenge\n(1995)\n',
 '\n25.\nOmkara\n(2006)\n',
 '\n26.\nLage Raho Munna Bhai\n(2006)\n',
 '\n27.\nIqbal\n(2005)\n',
 '\n28.\nThe Lunchbox\n(2013)\n',
 '\n29.\nBlack Friday\n(2004)\n',
 '\n30.\nCompany\n(2002)\n',
 '\n31.\nGolmaal\n(1979)\n',
 '\n32.\nDev.D\n(2009)\n',
 '\n33.\nJaane Bhi Do Yaaro\n(1983)\n',
 '\n34.\nOMG: Oh My God!\n(2012)\n',
 '\n35.\nMughal-E-Azam\n(1960)\n',
 '\n36.\nGulaal\n(2009)\n',
 '\n37.\nDor\n(2006)\n',
 '\n38.\nJab We Met\n(2007)\n',
 '\n39.\nPyaasa\n(1957)\n',
 '\n40.\nThe Legend of Bhagat Singh\n(2002)\n',
 '\n41.\nMasoom\n(1983)\n',
 '\n42.\nSalaam Bombay!\n(1988)\n',
 '\n43.\nSatya\n(1998)\n',
 '\n44.\nVicky Donor\n(2012)\n',
 '\n45.\nLakshya\n(2004)\n',
 '\n46.\nVaastav: The Reality\n(1999)\n',
 '\n47.\nKal Ho Naa Ho\n(2003)\n',
 '\n48.\nOye Lucky! Lucky Oye!\n(2008)\n',
 '\n49.\nSarfarosh\n(1999)\n',
 '\n50.\nGangaajal\n(2003)\n',
 '\n51.\nAngoor\n(1982)\n',
 '\n52.\nMadras Cafe\n(2013)\n',
 '\n53.\nEnglish Vinglish\n(2012)\n',
 '\n54.\nChupke Chupke\n(1975)\n',
 '\n55.\nJohnny Gaddaar\n(2007)\n',
 '\n56.\nMaqbool\n(2003)\n',
 '\n57.\nHazaaron Khwaishein Aisi\n(2003)\n',
 '\n58.\nRock On!!\n(2008)\n',
 '\n59.\nDon\n(1978)\n',
 '\n60.\nChhoti Si Baat\n(1976)\n',
 '\n61.\nGuide\n(1965)\n',
 '\n62.\nRaanjhanaa\n(2013)\n',
 '\n63.\nDeewaar\n(1975)\n',
 '\n64.\nSpecial Chabbis\n(2013)\n',
 '\n65.\nPadosan\n(1968)\n',
 '\n66.\nMumbai Meri Jaan\n(2008)\n',
 '\n67.\nAb Tak Chhappan\n(2004)\n',
 '\n68.\nKai po che!\n(2013)\n',
 '\n69.\nAwaara\n(1951)\n',
 '\n70.\nShree 420\n(1955)\n',
 '\n71.\nEarth\n(1999)\n',
 '\n72.\nGunda\n(1998)\n',
 '\n73.\nParinda\n(1989)\n',
 '\n74.\nDasvidaniya\n(2008)\n',
 '\n75.\nHey Ram\n(2000)\n',
 '\n76.\nPinjar: Beyond Boundaries...\n(2003)\n',
 '\n77.\nSocha Na Tha\n(2005)\n',
 '\n78.\nGuru\n(2007)\n',
 '\n79.\nBawarchi\n(1972)\n',
 '\n80.\nManorama: Six Feet Under\n(2007)\n',
 '\n81.\nMr. India\n(1987)\n',
 '\n82.\nAamir\n(2008)\n',
 '\n83.\nZakhm\n(1998)\n',
 '\n84.\nWater\n(I) (2005)\n',
 '\n85.\nStanley Ka Dabba\n(2011)\n',
 '\n86.\nAgneepath\n(1990)\n',
 '\n87.\nMy Name Is Khan\n(2010)\n',
 '\n88.\nQayamat Se Qayamat Tak\n(1988)\n',
 '\n89.\n3 Deewarein\n(2003)\n',
 '\n90.\nAbhimaan\n(1973)\n',
 '\n91.\nSarkar\n(2005)\n',
 '\n92.\nBheja Fry\n(2007)\n',
 '\n93.\nMother India\n(1957)\n',
 '\n94.\nJaane Tu... Ya Jaane Na\n(2008)\n',
 '\n95.\nDelhi Belly\n(2011)\n',
 '\n96.\nWake Up Sid\n(2009)\n',
 '\n97.\nRangeela\n(1995)\n',
 '\n98.\nShatranj Ke Khilari\n(1977)\n',
 '\n99.\nPyaar Ka Punchnama\n(2011)\n',
 '\n100.\nEk Hasina Thi\n(2004)\n']
```

In [88]:

```
year= []  # empty list

for i in soup.find_all('span',class_="lister-item-year text-muted unbold"):
    year.append(i.text)
year
```

Out[88]:

```
['(2006)',
 '(2009)',
 '(2007)',
 '(2001)',
 '(2004)',
 '(2001)',
 '(2012)',
 '(2012)',
 '(1971)',
 '(2003)',
 '(2008)',
 '(1994)',
 '(1975)',
 '(2013)',
 '(2000)',
 '(2010)',
 '(2012)',
 '(2005)',
 '(2007)',
 '(2006)',
 '(1992)',
 '(2011)',
 '(2012)',
 '(1995)',
 '(2006)',
 '(2006)',
 '(2005)',
 '(2013)',
 '(2004)',
 '(2002)',
 '(1979)',
 '(2009)',
 '(1983)',
 '(2012)',
 '(1960)',
 '(2009)',
 '(2006)',
 '(2007)',
 '(1957)',
 '(2002)',
 '(1983)',
 '(1988)',
 '(1998)',
 '(2012)',
 '(2004)',
 '(1999)',
 '(2003)',
 '(2008)',
 '(1999)',
 '(2003)',
 '(1982)',
 '(2013)',
 '(2012)',
 '(1975)',
 '(2007)',
 '(2003)',
 '(2003)',
 '(2008)',
 '(1978)',
 '(1976)',
 '(1965)',
 '(2013)',
 '(1975)',
 '(2013)',
 '(1968)',
 '(2008)',
 '(2004)',
 '(2013)',
 '(1951)',
 '(1955)',
 '(1999)',
 '(1998)',
 '(1989)',
 '(2008)',
 '(2000)',
 '(2003)',
 '(2005)',
 '(2007)',
 '(1972)',
 '(2007)',
 '(1987)',
 '(2008)',
 '(1998)',
 '(I) (2005)',
 '(2011)',
 '(1990)',
 '(2010)',
 '(1988)',
 '(2003)',
 '(1973)',
 '(2005)',
 '(2007)',
 '(1957)',
 '(2008)',
 '(2011)',
 '(2009)',
 '(1995)',
 '(1977)',
 '(2011)',
 '(2004)']
```

In [102]:

```
rating= []  # empty list

for i in soup.find_all('div',class_="ipl-rating-star small"):
    rating.append(i.text)
rating
```

Out[102]:

```
['\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.4\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.2\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.2\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n8.2\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.2\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n8.4\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8.5\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n8.4\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n8.3\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n8\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.3\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.4\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n8.1\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.7\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.9\n',
 '\n\n\n\n\n\n\n\n7.4\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.8\n',
 '\n\n\n\n\n\n\n\n7.4\n',
 '\n\n\n\n\n\n\n\n7.5\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.4\n',
 '\n\n\n\n\n\n\n\n7.5\n',
 '\n\n\n\n\n\n\n\n7.6\n',
 '\n\n\n\n\n\n\n\n7.5\n']
```

In [103]:

```
import pandas as pd
df=pd.DataFrame({'MOVIE NAME':Movie_name,'Rating':rating,'YEAR OF RELEASE':year})
df
```

Out[103]:

	MOVIE NAME	Rating	YEAR OF RELEASE
0	\n1.\nRang De Basanti\n(2006)\n	\n\n\n\n\n\n\n8.1\n	(2006)
1	\n2.\n3 Idiots\n(2009)\n	\n\n\n\n\n\n\n8.4\n	(2009)
2	\n3.\nTaare Zameen Par\n(2007)\n	\n\n\n\n\n\n\n8.3\n	(2007)
3	\n4.\nDil Chahta Hai\n(2001)\n	\n\n\n\n\n\n\n8.1\n	(2001)
4	\n5.\nSwades: We, the People\n(2004)\n	\n\n\n\n\n\n\n8.1\n	(2004)
...	...	...	...
95	\n96.\nWake Up Sid\n(2009)\n	\n\n\n\n\n\n\n7.6\n	(2009)
96	\n97.\nRangeela\n(1995)\n	\n\n\n\n\n\n\n7.4\n	(1995)
97	\n98.\nShatranj Ke Khilari\n(1977)\n	\n\n\n\n\n\n\n7.5\n	(1977)
98	\n99.\nPyaar Ka Punchnama\n(2011)\n	\n\n\n\n\n\n\n7.6\n	(2011)
99	\n100.\nEk Hasina Thi\n(2004)\n	\n\n\n\n\n\n\n7.5\n	(2004)

100 rows × 3 columns

4) Write s python program to display list of respected former presidents of India(i.e. Name , Term of office) from https://presidentofindia.nic.in/former-presidents.htm (https://presidentofindia.nic.in/former-presidents.htm)

In [115]:

```
page=requests.get('https://presidentofindia.nic.in/former-presidents.htm')
```

In [116]:

```
page
```

Out[116]:

```
<Response [200]>
```

In [117]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[117]:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://ww
w.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1"><title>
        Former Presidents - The President of India
</title><meta content="text/html; charset=utf-8" http-equiv="Content-Typ
e"/>
<!--<meta http-equiv="Content-Style-Type" content="text/css" /><meta http-
equiv="Content-Script-Type" content="type" />-->
<meta content="telephone=no" name="format-detection"/><meta content="IE=Em
ulateIE10" http-equiv="X-UA-Compatible"/>
<!-- Start Favicon -->
<link href="favicon.ico" rel="shortcut icon" type="image/x-icon"/><link hr
ef="js/panorama_viewer.css" rel="stylesheet" type="text/css"/>
<!-- Start Viewport -->
<!--<meta name="viewport" content="width=device-width, initial-scale=1, ma
ximum-scale=1, user-scalable=no" />-->
<!-- Start IE CSS -->
```

In [118]:

```
president=soup.find('div', class_="presidentListing")
president
```

Out[118]:

```
<div class="presidentListing">
<h3>Shri Pranab Mukherjee (1935-2020)</h3>
<p><span class="terms">Term of Office:</span> 25 July, 2012 to 25 July, 2017
</p>
<p><a href="http://pranabmukherjee.nic.in" target="_blank">http://pranabmukh
erjee.nic.in</a></p>
</div>
```

In [119]:

```
president.text
```

Out[119]:

```
'\nShri Pranab Mukherjee (1935-2020)\nTerm of Office: 25 July, 2012 to 25 Ju
ly, 2017 \nhttp://pranabmukherjee.nic.in\n'
```

In [122]:

```
term=soup.find('span', class_="terms")
term
```

Out[122]:

```
<span class="terms">Term of Office:</span>
```

In [123]:

```
term.text
```

Out[123]:

```
'Term of Office:'
```

In [124]:

```
president  =[] # empty list for store the

for i in soup.find_all('div', class_="presidentListing"):
    president.append(i.text)

president
```

Out[124]:

```
['\nShri Pranab Mukherjee (1935-2020)\nTerm of Office: 25 July, 2012 to 25 J
uly, 2017 \nhttp://pranabmukherjee.nic.in\n',
 '\nSmt Pratibha Devisingh Patil (birth - 1934)\nTerm of Office: 25 July, 20
07 to 25 July, 2012 \nhttp://pratibhapatil.nic.in\n',
 '\nDR. A.P.J. Abdul Kalam (1931-2015)\nTerm of Office: 25 July, 2002 to 25
July, 2007 \nhttp://abdulkalam.nic.in\n',
 '\nShri K. R. Narayanan (1920 - 2005)\nTerm of Office: 25 July, 1997 to 25
July, 2002 \n',
 '\nDr Shankar Dayal Sharma (1918-1999)\nTerm of Office: 25 July, 1992 to 25
July, 1997 \n',
 '\nShri R Venkataraman (1910-2009)\nTerm of Office: 25 July, 1987 to 25 Jul
y, 1992 \n',
 '\nGiani Zail Singh (1916-1994)\nTerm of Office: 25 July, 1982 to 25 July,
1987 \n',
 '\nShri Neelam Sanjiva Reddy (1913-1996)\nTerm of Office: 25 July, 1977 to
25 July, 1982 \n',
 '\nDr. Fakhruddin Ali Ahmed (1905-1977)\nTerm of Office: 24 August, 1974 to
11 February, 1977\n',
 '\nShri Varahagiri Venkata Giri (1894-1980)\nTerm of Office: 3 May, 1969 to
20 July, 1969 and 24 August, 1969 to 24 August, 1974\n',
 '\nDr. Zakir Husain (1897-1969)\nTerm of Office: 13 May, 1967 to 3 May, 196
9\n',
 '\nDr. Sarvepalli Radhakrishnan (1888-1975)\nTerm of Office: 13 May, 1962 t
o 13 May, 1967\n',
 '\nDr. Rajendra Prasad (1884-1963) \nTerm of Office: 26 January, 1950 to 13
May, 1962\n']
```

In [125]:

```
term  =[] # empty list for store the

for i in soup.find_all('span', class_="terms"):
    term.append(i.text)

term
```

Out[125]:

```
['Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:',
 'Term of Office:']
```

In [128]:

```
import pandas as pd
df=pd.DataFrame({'NAME':president,'TERM OF OFFICE':term,})
df
```

Out[128]:

	NAME	TERM OF OFFICE
0	\nShri Pranab Mukherjee (1935-2020)\nTerm of O...	Term of Office:
1	\nSmt Pratibha Devisingh Patil (birth - 1934)\...	Term of Office:
2	\nDR. A.P.J. Abdul Kalam (1931-2015)\nTerm of ...	Term of Office:
3	\nShri K. R. Narayanan (1920 - 2005)\nTerm of ...	Term of Office:
4	\nDr Shankar Dayal Sharma (1918-1999)\nTerm of...	Term of Office:
5	\nShri R Venkataraman (1910-2009)\nTerm of Off...	Term of Office:
6	\nGiani Zail Singh (1916-1994)\nTerm of Office...	Term of Office:
7	\nShri Neelam Sanjiva Reddy (1913-1996)\nTerm ...	Term of Office:
8	\nDr. Fakhruddin Ali Ahmed (1905-1977)\nTerm o...	Term of Office:
9	\nShri Varahagiri Venkata Giri (1894-1980)\nTe...	Term of Office:
10	\nDr. Zakir Husain (1897-1969)\nTerm of Office...	Term of Office:
11	\nDr. Sarvepalli Radhakrishnan (1888-1975)\nTe...	Term of Office:
12	\nDr. Rajendra Prasad (1884-1963) \nTerm of Of...	Term of Office:

5) Write a python program to scrape cricket rankings from icc-cricket.com. You have to scrape:

a) Top 10 ODI teams in men’s cricket along with the records for matches, points and rating.

b) Top 10 ODI Batsmen along with the records of their team and rating.

c) Top 10 ODI bowlers along with the records of their team and rating.

a) Top 10 ODI teams in men’s cricket along with the records for matches, points and rating.

In [254]:

```
page=requests.get('https://www.icc-cricket.com/rankings/mens/team-rankings/odi')
page
```

Out[254]:

```
<Response [200]>
```

In [255]:

```
soup=BeautifulSoup(page.content)
soup
```

```
<script>
  window.googletag = window.googletag || {cmd: []};
  googletag.cmd.push(function() {
  googletag.defineSlot('/182539303/ICC_Live_TeamODIRankings_Mobile_320x5
0', [320, 50], 'div-gpt-ad-1631699775893-0').addService(googletag.pubads
());
  googletag.pubads().enableSingleRequest();
  googletag.pubads().collapseEmptyDivs();
  googletag.enableServices();
  });
 </script>
<!-- /182539303/ICC_Live_TeamODIRankings_Mobile_320x50 -->
<div id="div-gpt-ad-1631699775893-0" style="min-width: 320px; min-height:
 50px;">
<script>
  googletag.cmd.push(function() { googletag.display('div-gpt-ad-1631699775
893-0'); });
  </script>
</div>
```

In [256]:

```
Team=soup.find('span', class_="u-hide-phablet")
Team
```

Out[256]:

```
<span class="u-hide-phablet">New Zealand</span>
```

In [257]:

```
Team.text
```

Out[257]:

```
'New Zealand'
```

In [264]:

```
matches=soup.find('td', class_="table-body__cell u-center-text")
matches
```

Out[264]:

```
<td class="table-body__cell u-center-text">22</td>
```

In [265]:

```
matches.text
```

Out[265]:

```
'22'
```

In [266]:

```
points=soup.find('td', class_="table-body__cell u-center-text")
points
```

Out[266]:

```
<td class="table-body__cell u-center-text">22</td>
```

In [267]:

```
points.text
```

Out[267]:

```
'22'
```

In [268]:

```
Rating=soup.find('td', class_="table-body__cell u-text-right rating")
Rating
```

Out[268]:

```
<td class="table-body__cell u-text-right rating">125</td>
```

In [269]:

```
Rating.text
```

Out[269]:

```
'125'
```

In [ ]:

```

```

In [ ]:

```

```

In [258]:

```
Team =[] # empty list for store the

for i in soup.find_all('span', class_="u-hide-phablet"):
    Team.append(i.text)

Team
```

Out[258]:

```
['New Zealand',
 'England',
 'Pakistan',
 'India',
 'Australia',
 'South Africa',
 'Bangladesh',
 'Sri Lanka',
 'West Indies',
 'Afghanistan',
 'Ireland',
 'Scotland',
 'UAE',
 'Netherlands',
 'Zimbabwe',
 'Oman',
 'United States',
 'Namibia',
```

In [262]:

```
Team1=Team[0:10]
Team1
```

Out[262]:

```
['New Zealand',
 'England',
 'Pakistan',
 'India',
 'Australia',
 'South Africa',
 'Bangladesh',
 'Sri Lanka',
 'West Indies',
 'Afghanistan']
```

In [270]:

```
matches =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-center-text"):
    matches.append(i.text)

matches
```

```
 '32',
 '2,306',
 '18',
 '1,238',
 '20',
 '1,083',
 '18',
 '814',
 '19',
 '724',
 '18',
 '603',
 '17',
 '539',
 '30',
 '919',
 '20',
 '544',
 '11',
```

In [278]:

```
matches1=matches[0:19:2]
matches1
```

Out[278]:

```
['22', '19', '22', '23', '19', '24', '29', '32', '18', '20']
```

In [271]:

```
point =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-center-text"):
    point.append(i.text)

point
```

...

In [277]:

```
point1=point[1:20:2]
point1
```

Out[277]:

```
['2,756',
 '2,005',
 '2,304',
 '2,325',
 '1,872',
 '2,275',
 '2,658',
 '2,306',
 '1,238',
 '1,083']
```

In [272]:

```
rating =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-text-right rating"):
    rating.append(i.text)

rating
```

Out[272]:

```
['125',
 '106',
 '105',
 '101',
 '99',
 '95',
 '92',
 '72',
 '69',
 '54',
 '45',
 '38',
 '34',
 '32',
 '31',
 '27',
 '22',
 '17',
 '6']
```

In [276]:

```
rating1=rating[0:10]
rating1
```

Out[276]:

```
['125', '106', '105', '101', '99', '95', '92', '72', '69', '54']
```

In [280]:

```
df=pd.DataFrame({'TEAM':Team1,'MATCHES':matches1,'POINT':point1,'RATING':rating1})
df
```

Out[280]:

	TEAM	MATCHES	POINT	RATING
0	New Zealand	22	2,756	125
1	England	19	2,005	106
2	Pakistan	22	2,304	105
3	India	23	2,325	101
4	Australia	19	1,872	99
5	South Africa	24	2,275	95
6	Bangladesh	29	2,658	92
7	Sri Lanka	32	2,306	72
8	West Indies	18	1,238	69
9	Afghanistan	20	1,083	54

b) Top 10 ODI Batsmen along with the records of their team and rating.¶

In [281]:

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

Out[281]:

```
<Response [200]>
```

In [282]:

```
soup=BeautifulSoup(page.content)
soup
```

```
...
```

In [283]:

```
name=soup.find('td', class_="table-body__cell rankings-table__name name")
name.text
```

Out[283]:

```
'\nImam-ul-Haq\n'
```

In [286]:

```
name.text.split()
```

Out[286]:

```
['Imam-ul-Haq']
```

In [287]:

```
team=soup.find('span', class_="table-body__logo-text")
team.text
```

Out[287]:

```
'PAK'
```

In [289]:

```

rating=soup.find('td', class_="table-body__cell rating")
rating.text
```

Out[289]:

```
'815'
```

In [290]:

```
rating  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rating"):
    rating.append(i.text)

rating
```

```
...
```

In [295]:

```
rating1=rating[0:10]
rating1
```

Out[295]:

```
['815', '811', '791', '789', '775', '769', '760', '745', '727', '725']
```

In [291]:

```
name =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rankings-table__name name"):
    name.append(i.text)

name
```

...

In [299]:

```
name1=name[0:10]
name1
```

Out[299]:

```
['\nImam-ul-Haq\n',
 '\nVirat Kohli\n',
 '\nRohit Sharma\n',
 '\nQuinton de Kock\n',
 '\nRoss Taylor\n',
 '\nRassie van der Dussen\n',
 '\nJonny Bairstow\n',
 '\nDavid Warner\n',
 '\nAaron Finch\n',
 '\nJoe Root\n']
```

In [292]:

```
team  =[] # empty list for store the

for i in soup.find_all('span', class_="table-body__logo-text"):
    team.append(i.text)

team
```

...

In [293]:

```
team1=team[0:10]
team1
```

Out[293]:

```
['PAK', 'IND', 'IND', 'SA', 'NZ', 'SA', 'ENG', 'AUS', 'AUS', 'ENG']
```

In [ ]:

In [296]:

```
df=pd.DataFrame({'NAME':name1,'TEAM':team1,'RATING':rating1,})
df
```

Out[296]:

	NAME	TEAM	RATING
0	\nImam-ul-Haq\n	PAK	815
1	\nVirat Kohli\n	IND	811
2	\nRohit Sharma\n	IND	791
3	\nQuinton de Kock\n	SA	789
4	\nRoss Taylor\n	NZ	775
5	\nRassie van der Dussen\n	SA	769
6	\nJonny Bairstow\n	ENG	760
7	\nDavid Warner\n	AUS	745
8	\nAaron Finch\n	AUS	727
9	\nJoe Root\n	ENG	725

In [ ]:

```

```

Top 10 ODI bowlers along with the records of their team and rating.

In [300]:

```
page=requests.get('https://www.icc-cricket.com/rankings/mens/player-rankings/odi/bowling')
page
```

Out[300]:

```
<Response [200]>
```

In [300]:

```
page=requests.get('https://www.icc-cricket.com/rankings/mens/player-rankings/odi/bowling')
page
```

Out[300]:

```
<Response [200]>
```

In [301]:

```
soup=BeautifulSoup(page.content)
soup
```

```
...
```

In [303]:

```
name=soup.find('td', class_="table-body__cell rankings-table__name name")
name.text
```

Out[303]:

```
'\nChris Woakes\n'
```

In [304]:

```
team=soup.find('span', class_="table-body__logo-text")
team.text
```

Out[304]:

```
'ENG'
```

In [305]:

```
rating=soup.find('td', class_="table-body__cell rating")
rating.text
```

Out[305]:

```
'686'
```

In [306]:

```
name =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rankings-table__name name"):
    name.append(i.text)

name
```

```
 '\nJhye Richardson\n',
 '\nZeeshan Maqsood\n',
 '\nDwaine Pretorius\n',
 '\nChad Soper\n',
 '\nRohan Mustafa\n',
 '\nCraig Young\n',
 '\nAndile Phehlukwayo\n',
 '\nImad Wasim\n',
 '\nHasan Ali\n',
 '\nKaleemullah\n',
 '\nTendai Chatara\n',
 '\nMohammad Mohammad Saifuddin\n',
 '\nDhananjaya de Silva\n',
 '\nHaris Rauf\n',
 '\nJimmy Neesham\n',
 '\nMoeen Ali\n',
 '\nSaqib Mahmood\n',
 '\nFred Klaassen\n',
 '\nSafyaan Sharif\n',
 '\nAshton Agar\n',
```

In [309]:

```
name1=name[0:10]
name1
```

Out[309]:

```
['\nChris Woakes\n',
 '\nMatt Henry\n',
 '\nShaheen Afridi\n',
 '\nJasprit Bumrah\n',
 '\nMujeeb Ur Rahman\n',
 '\nJosh Hazlewood\n',
 '\nMehedi Hasan\n',
 '\nMohammad Nabi\n',
 '\nShakib Al Hasan\n',
 '\nRashid Khan\n']
```

In [307]:

```
team =[] # empty list for store the

for i in soup.find_all('span', class_="table-body__logo-text"):
    team.append(i.text)

team
```

Out[307]:

```
['ENG',
 'NZ',
 'PAK',
 'IND',
 'AFG',
 'AUS',
 'BAN',
 'AFG',
 'BAN',
 'AFG',
 'IRE',
 'SA',
 'AUS',
 'AUS',
 'BAN',
 'IND',
 'AUS',
 'SA',
```

In [314]:

```
team1=team[0:10]
team1
```

Out[314]:

```
['ENG', 'NZ', 'PAK', 'IND', 'AFG', 'AUS', 'BAN', 'AFG', 'BAN', 'AFG']
```

In [308]:

```
rating  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rating"):
    rating.append(i.text)

rating
```

```
['686',
 '683',
 '681',
 '679',
 '676',
 '667',
 '661',
 '657',
 '657',
 '651',
 '646',
 '644',
 '632',
 '617',
 '614',
 '610',
 '609',
 '607',
 '604',
```

In [315]:

```
rating1=rating[0:10]
rating1
```

Out[315]:

```
['686', '683', '681', '679', '676', '667', '661', '657', '657', '651']
```

In [ ]:

In [316]:

```
df=pd.DataFrame({'NAME':name1,'TEAM':team1,'RATING':rating1,})
df
```

Out[316]:

```
                   NAME  TEAM  RATING
0       \nChris Woakes\n   ENG     686
1         \nMatt Henry\n    NZ     683
2     \nShaheen Afridi\n   PAK     681
3     \nJasprit Bumrah\n   IND     679
4  \nMujeeb Ur Rahman\n   AFG     676
5     \nJosh Hazlewood\n   AUS     667
6      \nMehedi Hasan\n   BAN     661
7    \nMohammad Nabi\n   AFG     657
8    \nShakib Al Hasan\n   BAN     657
9       \nRashid Khan\n   AFG     651
```

In [ ]:

In [ ]:

6) Write a python program to scrape cricket rankings from icc-cricket.com. You have to scrape:

a) Top 10 ODI teams in women’s cricket along with the records for matches, points and rating.

b) Top 10 women’s ODI Batting players along with the records of their team and rating.

c) Top 10 women’s ODI all-rounder along with the records of their team and rating.

In [ ]:

a) Top 10 ODI teams in women’s cricket along with the records for matches, points and rating.¶

In [317]:

```
page=requests.get('https://www.icc-cricket.com/rankings/womens/team-rankings/odi')
page
```

Out[317]:

```
<Response [200]>
```

In [318]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[318]:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta content="ICC Women's ODI Team Rankings | ICC" 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 test ma
tch cricket teams. Discover latest ICC rankings table, predict upcoming ma
tches, see points and ratings for all teams." name="description"/>
<meta content="@icc" property="twitter:site"/>
<meta content="Official International Cricket Council rankings for test ma
tch cricket teams. Discover latest ICC rankings table, predict upcoming ma
tches, see points and ratings for all teams." name="twitter:description"/>
<meta content="https://www.icc-cricket.com/resources/ver/i/elements/defaul
t-thumbnail.jpg" name="twitter:image"/>
<meta content="ICC Women's ODI Team Rankings | ICC" property="og:title"/>
<meta content="https://www.icc-cricket.com/resources/ver/i/elements/defaul
t-thumbnail.jpg" property="og:image"/>
```

In [319]:

```
team=soup.find('span', class_="u-hide-phablet")
team.text
```

Out[319]:

```
'Australia'
```

In [320]:

```
matches=soup.find('td', class_="table-body__cell u-center-text")
matches.text
```

Out[320]:

```
'32'
```

In [322]:

```
points=soup.find('td', class_="table-body__cell u-center-text")
points.text
```

Out[322]:

```
'32'
```

In [323]:

```
rating=soup.find('td', class_="table-body__cell u-text-right rating")
rating.text
```

Out[323]:

```
'123'
```

In [324]:

```
team=[] # empty list for store the

for i in soup.find_all('span', class_="u-hide-phablet"):
    team.append(i.text)

team
```

Out[324]:

```
['Australia',
 'South Africa',
 'England',
 'India',
 'New Zealand',
 'West Indies',
 'Bangladesh',
 'Pakistan',
 'Sri Lanka',
 'Ireland',
 'Zimbabwe',
 '',
 '',
 '',
 '',
 '']
```

In [330]:

```
team1=team[1:11]
team1
```

Out[330]:

```
['South Africa',
 'England',
 'India',
 'New Zealand',
 'West Indies',
 'Bangladesh',
 'Pakistan',
 'Sri Lanka',
 'Ireland',
 'Zimbabwe']
```

In [325]:

```
matches  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-center-text"):
    matches.append(i.text)

matches
```

Out[325]:

```
['32',
 '3,949',
 '30',
 '3,531',
 '29',
 '2,889',
 '31',
 '3,019',
 '30',
 '2,768',
 '12',
 '930',
 '30',
 '1,962',
 '8',
 '384',
 '8',
 '351',
 '8',
 '0']
```

In [333]:

```
matches1=matches[0:21:2]
matches1
```

Out[333]:

```
['32', '30', '29', '31', '30', '12', '30', '8', '8', '8']
```

In [326]:

```
points  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-center-text"):
    points.append(i.text)

points
```

Out[326]:

```
['32',
 '3,949',
 '30',
 '3,531',
 '29',
 '2,889',
 '31',
 '3,019',
 '30',
 '2,768',
 '12',
 '930',
 '30',
 '1,962',
 '8',
 '384',
 '8',
 '351',
 '8',
 '0']
```

In [332]:

```
points1=points[1:21:2]
points1
```

Out[332]:

```
['3,949',
 '3,531',
 '2,889',
 '3,019',
 '2,768',
 '930',
 '1,962',
 '384',
 '351',
 '0']
```

In [327]:

```
rating  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell u-text-right rating"):
    rating.append(i.text)

rating
```

Out[327]:

```
['123', '118', '100', '97', '92', '78', '65', '48', '44', '0']
```

In [ ]:

```

```

In [334]:

```
df=pd.DataFrame({'TEAM':team1,'MATCHES':matches1,'POINTS':points1,'RATING':rating})
df
```

Out[334]:

	TEAM	MATCHES	POINTS	RATING
0	South Africa	32	3,949	123
1	England	30	3,531	118
2	India	29	2,889	100
3	New Zealand	31	3,019	97
4	West Indies	30	2,768	92
5	Bangladesh	12	930	78
6	Pakistan	30	1,962	65
7	Sri Lanka	8	384	48
8	Ireland	8	351	44
9	Zimbabwe	8	0	0

In [ ]:

```

```

b) Top 10 women’s ODI Batting players along with the records of their team and rating.

In [335]:

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

Out[335]:

```
<Response [200]>
```

In [336]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[336]:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta content="ICC Women's ODI Batting | Player Rankings | ICC" name="twit
ter:title"/>
<meta content="website" property="og:type"/>
<meta content="summary_large_image" property="twitter:card"/>
<meta content="Official ICC Cricket website - live matches, scores, news,
highlights, commentary, rankings, videos and fixtures from the Internation
al Cricket Council." name="description"/>
<meta content="@icc" property="twitter:site"/>
<meta content="Official ICC Cricket website - live matches, scores, news,
highlights, commentary, rankings, videos and fixtures from the Internation
al Cricket Council." name="twitter:description"/>
<meta content="https://www.icc-cricket.com/resources/ver/i/elements/defaul
t-thumbnail.jpg" name="twitter:image"/>
<meta content="ICC Women's ODI Batting | Player Rankings | ICC" property
="og:title"/>
```

In [338]:

```
player_name=soup.find('td', class_="table-body__cell rankings-table__name name")
player_name.text
```

Out[338]:

```
'\nNatalie Sciver\n'
```

In [339]:

```
team=soup.find('span', class_="table-body__logo-text")
team.text
```

Out[339]:

```
'ENG'
```

In [340]:

```
rating=soup.find('td', class_="table-body__cell rating")
rating.text
```

Out[340]:

```
'750'
```

In [ ]:

In [341]:

```
player_name =[] # empty list for store
for i in soup.find_all('td', class_="table-body__cell rankings-table__name name"):
    player_name.append(i.text)

player_name
```

Out[341]:

```
['\nNatalie Sciver\n',
 '\nBeth Mooney\n',
 '\nLaura Wolvaardt\n',
 '\nMeg Lanning\n',
 '\nRachael Haynes\n',
 '\nAmy Satterthwaite\n',
 '\nSmriti Mandhana\n',
 '\nTammy Beaumont\n',
 '\nEllyse Perry\n',
 '\nDeandra Dottin\n',
 '\nStafanie Taylor\n',
 '\nHarmanpreet Kaur\n',
 '\nHeather Knight\n',
 '\nAmelia Kerr\n',
 '\nSophie Devine\n',
 '\nLizelle Lee\n',
 '\nSuzie Bates\n',
 '\nMignon du Preez\n',
```

In [342]:

```
player_name1=player_name[0:10]
player_name1
```

Out[342]:

```
['\nNatalie Sciver\n',
 '\nBeth Mooney\n',
 '\nLaura Wolvaardt\n',
 '\nMeg Lanning\n',
 '\nRachael Haynes\n',
 '\nAmy Satterthwaite\n',
 '\nSmriti Mandhana\n',
 '\nTammy Beaumont\n',
 '\nEllyse Perry\n',
 '\nDeandra Dottin\n']
```

In [343]:

```
team  =[] # empty list for store the

for i in soup.find_all('span', class_="table-body__logo-text"):
    team.append(i.text)

team
```

Out[343]:

```
['ENG',
 'AUS',
 'SA',
 'AUS',
 'AUS',
 'NZ',
 'IND',
 'ENG',
 'AUS',
 'WI',
 'WI',
 'IND',
 'ENG',
 'NZ',
 'NZ',
 'SA',
 'NZ',
 'SA',
```

In [344]:

```
team1=team[0:10]
team1
```

Out[344]:

```
['ENG', 'AUS', 'SA', 'AUS', 'AUS', 'NZ', 'IND', 'ENG', 'AUS', 'WI']
```

In [345]:

```
rating  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rating"):
    rating.append(i.text)

rating
```

Out[345]:

```
['750',
 '748',
 '713',
 '710',
 '701',
 '681',
 '669',
 '659',
 '642',
 '625',
 '616',
 '602',
 '601',
 '597',
 '591',
 '571',
 '568',
 '564',
```

In [346]:

```
rating1=rating[0:10]
rating1
```

Out[346]:

```
['750', '748', '713', '710', '701', '681', '669', '659', '642', '625']
```

In [347]:

```
print(len(player_name1),len(team1),len(rating1))
```

```
10 10 10
```

In [348]:

```
df=pd.DataFrame({'PLAYER NAME':player_name1,'TEAM NAME':team1,'RATING':rating1,})
df
```

Out[348]:

	PLAYER NAME	TEAM NAME	RATING
0	\nNatalie Sciver\n	ENG	750
1	\nBeth Mooney\n	AUS	748
2	\nLaura Wolvaardt\n	SA	713
3	\nMeg Lanning\n	AUS	710
4	\nRachael Haynes\n	AUS	701
5	\nAmy Satterthwaite\n	NZ	681
6	\nSmriti Mandhana\n	IND	669
7	\nTammy Beaumont\n	ENG	659
8	\nEllyse Perry\n	AUS	642
9	\nDeandra Dottin\n	WI	625

In [ ]:

```

```

c) Top 10 women’s ODI all-rounder along with the records of their team and rating.

In [349]:

```
page=requests.get('https://www.icc-cricket.com/rankings/womens/player-rankings/odi/all-roun
page
```

Out[349]:

```
<Response [200]>
```

In [350]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[350]:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta content="ICC Women's ODI All Rounder | Player Rankings | ICC" 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, news,
highlights, commentary, rankings, videos and fixtures from the Internation
al Cricket Council." name="description"/>
<meta content="@icc" property="twitter:site"/>
<meta content="Official ICC Cricket website - live matches, scores, news,
highlights, commentary, rankings, videos and fixtures from the Internation
al Cricket Council." name="twitter:description"/>
<meta content="https://www.icc-cricket.com/resources/ver/i/elements/defaul
t-thumbnail.jpg" name="twitter:image"/>
<meta content="ICC Women's ODI All Rounder | Player Rankings | ICC" proper
ty="og:title"/>
```

In [ ]:

```

```

In [351]:

```
Player_name=soup.find('td', class_="table-body__cell rankings-table__name name")
Player_name.text
```

Out[351]:

```
'\nEllyse Perry\n'
```

In [352]:

```
Team=soup.find('span', class_="table-body__logo-text")
Team.text
```

Out[352]:

```
'AUS'
```

In [353]:

```
Rating=soup.find('td', class_="table-body__cell rating")
Rating.text
```

Out[353]:

```
'374'
```

In [354]:

```
Player_name  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rankings-table__name name"):
    Player_name.append(i.text)

Player_name
```

Out[354]:

```
['\nEllyse Perry\n',
 '\nHayley Matthews\n',
 '\nMarizanne Kapp\n',
 '\nAmelia Kerr\n',
 '\nAshleigh Gardner\n',
 '\nDeepti Sharma\n',
 '\nJess Jonassen\n',
 '\nSune Luus\n',
 '\nKatherine Brunt\n',
 '\nJhulan Goswami\n',
 '\nSophie Ecclestone\n',
 '\nStafanie Taylor\n',
 '\nSophie Devine\n',
 '\nNida Dar\n',
 '\nRumana Ahmed\n',
 '\nSalma Khatun\n',
 '\nDane van Niekerk\n',
 '\nHeather Knight\n',
 '\nTahlia McGrath\n']
```

In [357]:

```
Player_name1=Player_name[0:10]
Player_name1
```

Out[357]:

```
['\nEllyse Perry\n',
 '\nHayley Matthews\n',
 '\nMarizanne Kapp\n',
 '\nAmelia Kerr\n',
 '\nAshleigh Gardner\n',
 '\nDeepti Sharma\n',
 '\nJess Jonassen\n',
 '\nSune Luus\n',
 '\nKatherine Brunt\n',
 '\nJhulan Goswami\n']
```

In [355]:

```
Team  =[] # empty list for store the

for i in soup.find_all('span', class_="table-body__logo-text"):
    Team.append(i.text)

Team
```

Out[355]:

```
['AUS',
 'WI',
 'SA',
 'NZ',
 'AUS',
 'IND',
 'AUS',
 'SA',
 'ENG',
 'IND',
 'ENG',
 'WI',
 'NZ',
 'PAK',
 'BAN',
 'BAN',
 'SA',
 'ENG',
 'AUS']
```

In [358]:

```
Team1=Team[0:10]
Team1
```

Out[358]:

```
['AUS', 'WI', 'SA', 'NZ', 'AUS', 'IND', 'AUS', 'SA', 'ENG', 'IND']
```

In [356]:

```
Rating  =[] # empty list for store the

for i in soup.find_all('td', class_="table-body__cell rating"):
    Rating.append(i.text)

Rating
```

Out[356]:

```
['374',
 '338',
 '338',
 '335',
 '269',
 '249',
 '245',
 '223',
 '221',
 '217',
 '206',
 '205',
 '202',
 '198',
 '169',
 '167',
 '159',
 '156',
 '149']
```

In [360]:

```
Rating1=Rating[0:10]
Rating1
```

Out[360]:

```
['374', '338', '338', '335', '269', '249', '245', '223', '221', '217']
```

In [ ]:

In [361]:

```
df=pd.DataFrame({'PLAYER NAME':Player_name1,'TEAM':Team1,'RATING':Rating1})
df
```

Out[361]:

	PLAYER NAME	TEAM	RATING
0	\nEllyse Perry\n	AUS	374
1	\nHayley Matthews\n	WI	338
2	\nMarizanne Kapp\n	SA	338
3	\nAmelia Kerr\n	NZ	335
4	\nAshleigh Gardner\n	AUS	269
5	\nDeepti Sharma\n	IND	249
6	\nJess Jonassen\n	AUS	245
7	\nSune Luus\n	SA	223
8	\nKatherine Brunt\n	ENG	221
9	\nJhulan Goswami\n	IND	217

In [ ]:

In [ ]:

In [ ]:

In [ ]:

7) Write a python program to scrape mentioned news details from https://www.cnbc.com/world/?region=world (https://www.cnbc.com/world/?region=world) :

i) Headline

ii) Time

iii) News Link

In [242]:

```
page=requests.get('https://www.cnbc.com/world/?region=world')
page
```

Out[242]:

```
<Response [200]>
```

In [243]:

```
soup=BeautifulSoup(page.content)
soup
```

Out[243]:

```
<!DOCTYPE html>
<html itemscope="" itemtype="https://schema.org/WebPage" lang="en" prefix
="og=https://ogp.me/ns#"><head><link as="font" crossorigin="anonymous" hre
f="https://static-redesign.cnbcfm.com/dist/icomoon.ttf" rel="preload" type
="font/ttf"/><link as="font" crossorigin="anonymous" href="https://static-
redesign.cnbcfm.com/dist/351C86_0_0.woff2" rel="preload" type="font/woff
2"/><link as="font" crossorigin="anonymous" href="https://static-redesign.
cnbcfm.com/dist/351C86_1_0.woff2" rel="preload" type="font/woff2"/><link a
s="font" crossorigin="anonymous" href="https://static-redesign.cnbcfm.com/
dist/351C86_2_0.woff2" rel="preload" type="font/woff2"/><link as="font" cr
ossorigin="anonymous" href="https://static-redesign.cnbcfm.com/dist/351C86
_3_0.woff2" rel="preload" type="font/woff2"/><link as="font" crossorigin
="anonymous" href="https://static-redesign.cnbcfm.com/dist/351C86_4_0.woff
2" rel="preload" type="font/woff2"/><link as="font" crossorigin="anonymou
s" href="https://static-redesign.cnbcfm.com/dist/LyonText-Bold-Web.woff2"
rel="preload" type="font/woff2"/><link as="font" crossorigin="anonymous" h
ref="https://static-redesign.cnbcfm.com/dist/LyonText-Regular-Web.woff2" r
el="preload" type="font/woff2"/><meta content="telephone=no" name="format-
```

In [244]:

```
time=soup.find('span', class_="LatestNews-wrapper")
time
```

Out[244]:

```
<span class="LatestNews-wrapper"><time class="LatestNews-timestamp">13 Min A
go</time></span>
```

In [245]:

```
time.text
```

Out[245]:

```
'13 Min Ago'
```

In [246]:

```
head_line=soup.find('a', class_="LatestNews-headline")
head_line.text
```

Out[246]:

```
'NATO countries gather as Ukraine war overshadows Europe; Russian attack on
shopping mall kills 15'
```

In [247]:

```
head_line  =[] # empty list for store the

for i in soup.find_all('a', class_="LatestNews-headline"):
    head_line.append(i.text)

head_line
```

Out[247]:

```
['NATO countries gather as Ukraine war overshadows Europe; Russian attack on
shopping mall kills 15',
 'Why one stock brokerage is bullish on Reliance Industries and Infosys',
 'UK Covid cases are on the rise, but a return to restrictions looks unlikel
y',
 "Hydrogen-powered trains to be used in Germany's capital region",
 "As Klarna and Affirm falter, new 'buy now, pay later' startups steal spotl
ight",
 "G-7's infrastructure plan offers an alternative to China's BRI in a 'delib
erate way'",
 "Total shutdown of Russian gas pipelines to Europe 'is not inconceivable'",
 'European markets head for negative open as global investor confidence dip
s',
 "Fund manager says this stock is so cheap it makes 'no sense'",
 'Investors could do ‘a lot worse’ than FedEx here, Jim Cramer says',
 'This fund manager is beating the market — and he has 4 tips for investor
s',
 'Asia-Pacific stocks mixed as investors weigh economic concerns',
 "Cramer's lightning round: I like Belden over Encore Wire",
 "U.S. may lose silicon wafer factory if CHIPS Act isn't funded, Raimondo sa
ys",
 'Mixed inflation data might be necessary for a soft landing, Jim Cramer say
s',
 'Only 20% of U.S. workers in office three days or more: IBM CEO',
 'Stock index futures slip following a losing day Monday ',
 'JetBlue ups offer for Spirit Airlines ahead of shareholder vote on Frontie
r bid',
 'CVS capping purchases of Plan B pills to ensure consistent supply',
 'Best trades on CNBC Monday: Pros say these stocks have stabilized',
 "Avoidable or inevitable recession? Billionaire David Rubenstein's view",
 'Gay Connecticut justice zings Thomas on same-sex marriage ruling repeal id
ea',
 'JPMorgan keeps dividend unchanged as rivals including Morgan Stanley hike
payout',
 'Pelosi says Democrats are mulling plans to protect abortion access after r
uling',
 "Renewable energy vs. fossil fuels? It's a false choice, says John Doerr",
 "Nike earnings top Wall Street's expectations",
 'Alabama abortion clinic cancels 100 appointments after Roe v. Wade is over
turned',
 "American Airlines' Envoy offers pilots triple pay to pick up trips in Jul
y",
 'Gender gaps in financial knowledge, retirement readiness persist, research
shows',
 'Abysmal manufacturing surveys are the latest signs of a major economic slo
wdown']
```

In [248]:

```
time  =[] # empty list for store the

for i in soup.find_all('span', class_="LatestNews-wrapper"):
    time.append(i.text)

time
```

Out[248]:

```
['13 Min Ago',
 '19 Min Ago',
 '43 Min Ago',
 '47 Min Ago',
 '47 Min Ago',
 '52 Min Ago',
 '1 Hour Ago',
 '2 Hours Ago',
 '5 Hours Ago',
 '6 Hours Ago',
 '6 Hours Ago',
 '6 Hours Ago',
 '7 Hours Ago',
 '7 Hours Ago',
 '8 Hours Ago',
 '8 Hours Ago',
 '8 Hours Ago',
 '8 Hours Ago',
 '9 Hours Ago',
 '9 Hours Ago',
 '9 Hours Ago',
 '9 Hours Ago',
 '9 Hours Ago',
 '10 Hours Ago',
 '10 Hours Ago',
 '10 Hours Ago',
 '10 Hours Ago',
 '10 Hours Ago',
 '11 Hours Ago',
 '11 Hours Ago']
```

In [250]:

```
df=pd.DataFrame({'HEADLINE':head_line,'TIME': time,})
df
```

Out[250]:

```
                                            HEADLINE          TIME
 0   NATO countries gather as Ukraine war overshado...    13 Min Ago
 1     Why one stock brokerage is bullish on Reliance...    19 Min Ago
 2        UK Covid cases are on the rise, but a return t...    43 Min Ago
 3    Hydrogen-powered trains to be used in Germany'...    47 Min Ago
 4       As Klarna and Affirm falter, new 'buy now, pay...    47 Min Ago
 5           G-7's infrastructure plan offers an alternativ...    52 Min Ago
 6       Total shutdown of Russian gas pipelines to Eur...    1 Hour Ago
 7    European markets head for negative open as glo...   2 Hours Ago
 8     Fund manager says this stock is so cheap it ma...   5 Hours Ago
 9       Investors could do ‘a lot worse’ than FedEx he...   6 Hours Ago
10    This fund manager is beating the market — and ...   6 Hours Ago
11      Asia-Pacific stocks mixed as investors weigh e...   6 Hours Ago
12        Cramer's lightning round: I like Belden over E...   7 Hours Ago
13        U.S. may lose silicon wafer factory if CHIPS A...   7 Hours Ago
14        Mixed inflation data might be necessary for a ...   8 Hours Ago
15       Only 20% of U.S. workers in office three days ...   8 Hours Ago
16          Stock index futures slip following a losing da...   8 Hours Ago
17            JetBlue ups offer for Spirit Airlines ahead of...   8 Hours Ago
18       CVS capping purchases of Plan B pills to ensur...   9 Hours Ago
19  Best trades on CNBC Monday: Pros say these sto...   9 Hours Ago
20          Avoidable or inevitable recession? Billionaire...   9 Hours Ago
21    Gay Connecticut justice zings Thomas on same-s...   9 Hours Ago
22     JPMorgan keeps dividend unchanged as rivals in...   9 Hours Ago
23       Pelosi says Democrats are mulling plans to pro...  10 Hours Ago
24          Renewable energy vs. fossil fuels? It's a fals...  10 Hours Ago
25             Nike earnings top Wall Street's expectations  10 Hours Ago
26     Alabama abortion clinic cancels 100 appointmen...  10 Hours Ago
27           American Airlines' Envoy offers pilots triple ...  10 Hours Ago
28       Gender gaps in financial knowledge, retirement...  11 Hours Ago
29      Abysmal manufacturing surveys are the latest s...  11 Hours Ago
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

8) Write a python program to scrape the details of most downloaded articles from AI in last 90 days.

https://www.journals.elsevier.com/artificial-intelligence/most-downloaded-articles (https://www.journals.elsevier.com/artificial-intelligence/most-downloaded-articles)

Scrape below mentioned details :

i) Paper Title

ii) Authors

iii) Published Date

iv) Paper URL

In [234]:

```
page=requests.get('https://www.journals.elsevier.com/artificial-intelligence/most-downloade
page
```

Out[234]:

```
<Response [200]>
```

In [223]:

```
soup=BeautifulSoup(page.content)
soup
```

```
lsevier.com/__data/cover_img/505601.gif" name="og:image:secure_url"/><meta
content="journals.elsevier.com/artificial-intelligence/most-downloaded-art
icles" name="og:url"/><meta content="website" property="og:type"/><meta co
ntent="Most Downloaded Articles - Artificial Intelligence - Journal - Else
vier" name="twitter:title"/><meta content="The journal of Artificial Intel
ligence (AIJ)  welcomes papers on broad aspects of AI that constitute adva
nces in the overall field including, but not limited …" name="twitter:desc
ription"/><meta content="https://www.elsevier.com/__data/cover_img/505601.
gif" name="twitter:image"/><meta content="summary" name="twitter:card"/><s

cript type="application/ld+json">{&quot;@context&quot;:&quot;https://schem
a.org&quot;,&quot;@type&quot;:&quot;Periodical&quot;,&quot;issn&quot;:&quo
t;0004-3702&quot;,&quot;name&quot;:&quot;Artificial Intelligence&quot;,&qu
ot;publisher&quot;:&quot;Elsevier&quot;,&quot;acquireLicensePage&quot;:&qu
ot;https://www.elsevier.com/journals/artificial-intelligence/0004-3702/sub
scribe?subscriptiontype=institutional&quot;}</script><link as="image" imag
esizes="100vw" imagesrcset="/_next/image?url=%2Fimages%2Flogos%2Felsevier-
graphic.svg&amp;w=640&amp;q=75 640w, /_next/image?url=%2Fimages%2Flogos%2F
elsevier-graphic.svg&amp;w=750&amp;q=75 750w, /_next/image?url=%2Fimages%2
Flogos%2Felsevier-graphic.svg&amp;w=828&amp;q=75 828w, /_next/image?url=%2
Fimages%2Flogos%2Felsevier-graphic.svg&amp;w=1080&amp;q=75 1080w, /_next/i
```

In [224]:

```
paper_title=soup.find('h2', class_="sc-1qrq3sd-1 MKjKb sc-1nmom32-0 sc-1nmom32-1 hqhUYH ebT
paper_title
```

Out[224]:

```
<h2 class="sc-1qrq3sd-1 MKjKb sc-1nmom32-0 sc-1nmom32-1 hqhUYH ebTA-dR">Rewa
rd is enough</h2>
```

In [225]:

```
paper_title.text
```

Out[225]:

```
'Reward is enough'
```

In [226]:

```
authors=soup.find('span',class_="sc-1w3fpd7-0 pgLAT")
authors.text
```

Out[226]:

```
'Silver, David, Singh, Satinder, Precup, Doina, Sutton, Richard S. '
```

In [227]:

```
published_date=soup.find('span',class_="sc-1thf9ly-2 bKddwo")
published_date.text
```

Out[227]:

```
'October 2021'
```

In [228]:

```
paper_url=soup.find('li',class_="sc-9zxyh7-1 sc-9zxyh7-2 exAXfr jQmQZp")
paper_url.text
```

Out[228]:

```
'Reward is enoughSilver, David, Singh, Satinder, Precup, Doina, Sutton, Rich
ard S. Open AccessOctober 2021'
```

In [ ]:

```
#paper_url="sc-9zxyh7-1 sc-9zxyh7-2 exAXfr jQmQZp"
```

In [229]:

```
authors  =[] # empty list for store the

for i in soup.find_all('span',class_="sc-1w3fpd7-0 pgLAT"):
    authors.append(i.text)

authors
```

Out[229]:

```
['Silver, David, Singh, Satinder, Precup, Doina, Sutton, Richard S. ',
 'Evans, Richard, Bošnjak, Matko and 5 more',
 'Prakken, Henry, Sartor, Giovanni ',
 'Boden, Margaret A. ',
 'Lemaignan, Séverin, Warnier, Mathieu and 3 more',
 'Miller, Tim ',
 'Evans, Richard, Hernández-Orallo, José and 3 more',
 'Sharon, Guni, Stern, Roni, Felner, Ariel, Sturtevant, Nathan R. ',
 'Sutton, Richard S., Precup, Doina, Singh, Satinder ',
 'Bard, Nolan, Foerster, Jakob N. and 13 more',
 'van der Waa, Jasper, Nieuwburg, Elisabeth, Cremers, Anita, Neerincx, Mark
',
 'Bench-Capon, T.J.M., Dunne, Paul E. ',
 'Bošanský, Branislav, Lisý, Viliam and 3 more',
 'Luo, Wenhan, Xing, Junliang and 4 more',
 'Blum, Avrim L., Langley, Pat ',
 'Arora, Saurabh, Doshi, Prashant ',
 'Aas, Kjersti, Jullum, Martin, Løland, Anders ',
 'Kliegr, Tomáš, Bahník, Štěpán, Fürnkranz, Johannes ',
 'Pereira, Gonçalo, Prada, Rui, Santos, Pedro A. ',
 'Riveiro, Maria, Thill, Serge ',
 'Kenny, Eoin M., Ford, Courtney, Quinn, Molly, Keane, Mark T. ',
 'Hutter, Frank, Xu, Lin, Hoos, Holger H., Leyton-Brown, Kevin ',
 'Kohavi, Ron, John, George H. ',
 'Suchan, Jakob, Bhatt, Mehul, Varadarajan, Srikrishna ',
 'Ying, Mingsheng ']
```

In [230]:

```
published_date  =[] # empty list for store the

for i in soup.find_all('span',class_="sc-1thf9ly-2 bKddwo"):
    published_date.append(i.text)

published_date
```

Out[230]:

```
['October 2021',
 'October 2021',
 'October 2015',
 'August 1998',
 'June 2017',
 'February 2019',
 'April 2021',
 'February 2015',
 'August 1999',
 'March 2020',
 'February 2021',
 'October 2007',
 'August 2016',
 'April 2021',
 'December 1997',
 'August 2021',
 'September 2021',
 'June 2021',
 'December 2016',
 'September 2021',
 'May 2021',
 'January 2014',
 'December 1997',
 'October 2021',
 'February 2010']
```

In [231]:

```
paper_title =[] # empty list for store the

for i in soup.find_all('h2', class_="sc-1qrq3sd-1 MKjKb sc-1nmom32-0 sc-1nmom32-1 hqhUYH eb
    paper_title.append(i.text)

paper_title
```

Out[231]:

```
['Reward is enough',
 'Making sense of raw input',
 'Law and logic: A review from an argumentation perspective',
 'Creativity and artificial intelligence',
 'Artificial cognition for social human–robot interaction: An implementatio
n',
 'Explanation in artificial intelligence: Insights from the social science
s',
 'Making sense of sensory input',
 'Conflict-based search for optimal multi-agent pathfinding',
 'Between MDPs and semi-MDPs: A framework for temporal abstraction in reinfo
rcement learning',
 'The Hanabi challenge: A new frontier for AI research',
 'Evaluating XAI: A comparison of rule-based and example-based explanation
s',
 'Argumentation in artificial intelligence',
 'Algorithms for computing strategies in two-player simultaneous move game
s',
 'Multiple object tracking: A literature review',
 'Selection of relevant features and examples in machine learning',
 'A survey of inverse reinforcement learning: Challenges, methods and progre
ss',
 'Explaining individual predictions when features are dependent: More accura
te approximations to Shapley values',
 'A review of possible effects of cognitive biases on interpretation of rule
-based machine learning models',
 'Integrating social power into the decision-making of cognitive agents',
 "“That's (not) the output I expected!” On the role of end user expectations
in creating explanations of AI systems",
 'Explaining black-box classifiers using post-hoc explanations-by-example: T
he effect of explanations and error-rates in XAI user studies',
 'Algorithm runtime prediction: Methods & evaluation',
 'Wrappers for feature subset selection',
 'Commonsense visual sensemaking for autonomous driving – On generalised neu
rosymbolic online abduction integrating vision and semantics',
 'Quantum computation, quantum theory and AI']
```

In [232]:

```
import pandas as pd
df=pd.DataFrame({'paper_title':paper_title,'authors':authors,'published_dat
df
```

Out[232]:

	paper_title	authors	published_date
0	Reward is enough	Silver, David, Singh, Satinder, Precup, Doina,...	October 2021
1	Making sense of raw input	Evans, Richard, Bošnjak, Matko and 5 more	October 2021
2	Law and logic: A review from an argumentation ...	Prakken, Henry, Sartor, Giovanni	October 2015
3	Creativity and artificial intelligence	Boden, Margaret A.	August 1998
4	Artificial cognition for social human–robot in...	Lemaignan, Séverin, Warnier, Mathieu and 3 more	June 2017
5	Explanation in artificial intelligence: Insigh...	Miller, Tim	February 2019
6	Making sense of sensory input	Evans, Richard, Hernández-Orallo, José and 3 more	April 2021
7	Conflict-based search for optimal multi-agent ...	Sharon, Guni, Stern, Roni, Felner, Ariel, Stur...	February 2015
8	Between MDPs and semi-MDPs: A framework for te...	Sutton, Richard S., Precup, Doina, Singh, Sati...	August 1999
9	The Hanabi challenge: A new frontier for AI re...	Bard, Nolan, Foerster, Jakob N. and 13 more	March 2020
10	Evaluating XAI: A comparison of rule-based and...	van der Waa, Jasper, Nieuwburg, Elisabeth, Cre...	February 2021
11	Argumentation in artificial intelligence	Bench-Capon, T.J.M., Dunne, Paul E.	October 2007
12	Algorithms for computing strategies in two-pla...	Bošanský, Branislav, Lisý, Viliam and 3 more	August 2016
13	Multiple object tracking: A literature review	Luo, Wenhan, Xing, Junliang and 4 more	April 2021
14	Selection of relevant features and examples in...	Blum, Avrim L., Langley, Pat	December 1997
15	A survey of inverse reinforcement learning: Ch...	Arora, Saurabh, Doshi, Prashant	August 2021
16	Explaining individual predictions when feature...	Aas, Kjersti, Jullum, Martin, Løland, Anders	September 2021
17	A review of possible effects of cognitive bias...	Kliegr, Tomáš, Bahník, Štěpán, Fürnkranz, Joha...	June 2021
18	Integrating social power into the decision-mak...	Pereira, Gonçalo, Prada, Rui, Santos, Pedro A.	December 2016
19	“That's (not) the output I expected!” On the r...	Riveiro, Maria, Thill, Serge	September 2021
20	Explaining black-box classifiers using post-ho...	Kenny, Eoin M., Ford, Courtney, Quinn, Molly, ...	May 2021
21	Algorithm runtime prediction: Methods & evalua...	Hutter, Frank, Xu, Lin, Hoos, Holger H., Leyto...	January 2014

	paper_title	authors	published_date
22	Wrappers for feature subset selection	Kohavi, Ron, John, George H.	December 1997
23	Commonsense visual sensemaking for autonomous ...	Suchan, Jakob, Bhatt, Mehul, Varadarajan, Srik...	October 2021
24	Quantum computation, quantum theory and AI	Ying, Mingsheng	February 2010

```
In [ ]:
```

```
In [ ]:
```

9) Write a python program to scrape mentioned details from dineout.co.in :

i) Restaurant name

ii) Cuisine

iii) Location

iv) Ratings

v) Image URL

```
In [165]:
page=requests.get('https://www.dineout.co.in/delhi-restaurants/family-dining')
page
```

```
Out[165]:
<Response [200]>
```

In [166]:

```
soup=BeautifulSoup(page.content)
soup
```

```
            iframe[name="google_conversion_frame"]{
                display:none !important;
            }
        </style><link href="https://dn1.dineout-cdn.co.in" rel="preconnec
t"/><link href="https://im1.dineout.co.in" rel="preconnect"/><link href="h
ttps://wp.dineout.co.in" rel="preconnect"/><link href="https://wzrkt.com"
rel="preconnect"/><link href="https://d2r1yp2w7bby2u.cloudfront.net" rel
="preconnect"/><link href="https://s3-eu-west-1.amazonaws.com" rel="precon
nect"/><link href="https://connect.facebook.net" rel="preconnect"/><link h
ref="https://www.google-analytics.com" rel="preconnect"/><link href="http
s://st1.dineout-cdn.co.in" rel="dns-prefetch"/><link href="https://dn1.din
eout-cdn.co.in" rel="dns-prefetch"/><link href="https://im1.dineout.co.in"
rel="dns-prefetch"/><link href="https://wp.dineout.co.in" rel="dns-prefetc
h"/><link href="https://wzrkt.com" rel="dns-prefetch"/><link href="http
s://d2r1yp2w7bby2u.cloudfront.net" rel="dns-prefetch"/><link href="http
s://s3-eu-west-1.amazonaws.com" rel="dns-prefetch"/><link href="https://co
nnect.facebook.net" rel="dns-prefetch"/><link href="https://www.google-ana
lytics.com" rel="dns-prefetch"/><link href="https://st2.dineout-cdn.co.in"
rel="dns-prefetch"/><link href="https://st3.dineout-cdn.co.in" rel="dns-pr
efetch"/><link href="https://st4.dineout-cdn.co.in" rel="dns-prefetch"/><l
```

In [168]:

```
Restaurant_name=soup.find('a', class_="restnt-name ellipsis")
Restaurant_name.text
```

Out[168]:

```
'The Imperial Spice'
```

In [169]:

```
Cuisine=soup.find('div', class_="detail-info")
Cuisine.text
```

Out[169]:

```
'₹ 3,000 for 2 (approx) | North Indian, Chinese, Continental, Mughlai'
```

In [170]:

```
location=soup.find('div', class_="restnt-loc ellipsis")
location.text
```

Out[170]:

```
'M-Block,Connaught Place, Central Delhi'
```

In [172]:

```
ratings=soup.find('div', class_="restnt-rating rating-4")
ratings.text
```

Out[172]:

```
'4.4'
```

In [174]:

```
image_url=soup.find('img', class_="no-img")
image_url.text
```

Out[174]:

```
''
```

In [175]:

```
Restaurant_name =[] # empty list for store the

for i in soup.find_all('a', class_="restnt-name ellipsis"):
    Restaurant_name.append(i.text)

Restaurant_name
```

Out[175]:

```
['The Imperial Spice',
 'Out Of The Box Courtyard',
 'The Great Kabab Factory',
 'Baluchi',
 'The Connaught Bar',
 'Fifty9',
 '24/7',
 'OKO',
 'The Grill Room',
 'The Lalit Boulangerie',
 'Out Of The Box',
 'Veg Gulati',
 'Yellow Brick Road',
 'G2 Cafe',
 "Larry's China",
 'Jungle Jamboree',
 'Antidot Waterbar and Cafe',
 'Veg Gulati',
 "TK's Oriental Grill",
 'The Construction Co. Cafe',
 'The China Kitchen']
```

In [176]:

```
Cuisine =[] # empty list for store the

for i in soup.find_all('div', class_="detail-info"):
    Cuisine.append(i.text)

Cuisine
```

Out[176]:

```
['₹ 3,000 for 2 (approx) | North Indian, Chinese, Continental, Mughlai',
 '₹ 2,200 for 2 (approx) | North Indian, Mediterranean, Chinese, Italian',
 '₹ 3,500 for 2 (approx) | North Indian, Finger Food',
 '₹ 4,500 for 2 (approx) | North Indian, Mughlai',
 '₹ 3,400 for 2 (approx) | Finger Food',
 '₹ 2,900 for 2 (approx) | North Indian, Continental',
 '₹ 4,800 for 2 (approx) | Continental, North Indian, Asian, Italian',
 '₹ 5,200 for 2 (approx) | Asian, Chinese, Thai, Japanese',
 '₹ 5,000 for 2 (approx) | Continental, European, Seafood',
 '₹ 1,000 for 2 (approx) | Desserts',
 '₹ 2,200 for 2 (approx) | North Indian, Italian, Continental, Asian, Tex Me
x',
 '₹ 1,500 for 2 (approx) | North Indian',
 '₹ 3,500 for 2 (approx) | Continental, North Indian, Italian',
 '₹ 1,600 for 2 (approx) | Continental, North Indian, Asian, Italian, Europe
an, Beverages',
 '₹ 3,500 for 2 (approx) | Chinese, Seafood',
 '₹ 1,400 for 2 (approx) | North Indian, Asian, Italian',
 '₹ 1,800 for 2 (approx) | Asian, North Indian, Continental1 deal availabl
e',
 '₹ 1,200 for 2 (approx) | North Indian',
 '₹ 3,400 for 2 (approx) | Japanese, Indonesian, Sushi, Thai',
 '₹ 1,200 for 2 (approx) | Chinese, Italian, Continental',
 '₹ 4,500 for 2 (approx) | Chinese, Desserts']
```

In [177]:

```
location =[] # empty list for store the

for i in soup.find_all('div', class_="restnt-loc ellipsis"):
    location.append(i.text)

location
```

Out[177]:

```
['M-Block,Connaught Place, Central Delhi',
 'Connaught Place, Central Delhi',
 'Radisson Blu Marina,Connaught Place, Central Delhi',
 'The Lalit New Delhi,Connaught Place, Central Delhi',
 'Radisson Blu Marina,Connaught Place, Central Delhi',
 'Radisson Blu Marina,Connaught Place, Central Delhi',
 'The Lalit New Delhi,Connaught Place, Central Delhi',
 'The Lalit New Delhi,Connaught Place, Central Delhi',
 'The Lalit New Delhi,Connaught Place, Central Delhi',
 'The Lalit New Delhi,Connaught Place, Central Delhi',
 'Khan Market, Central Delhi',
 'Pandara Road, Central Delhi',
 'Ambassador,Khan Market, Central Delhi',
 'Anand Vihar, East Delhi',
 'Ambassador,Khan Market, Central Delhi',
 '3CS Mall,Lajpat Nagar - 3, South Delhi',
 'Safdarjung Enclave Market,Safdarjung, South Delhi',
 'Green Park, South Delhi',
 'Hyatt Regency Delhi,Bhikaji Cama Place, South Delhi',
 'Satya Niketan, South Delhi',
 'Hyatt Regency Delhi,Bhikaji Cama Place, South Delhi']
```

In [178]:

```
ratings =[] # empty list for store the

for i in soup.find_all('div', class_="restnt-rating rating-4"):
    ratings.append(i.text)

ratings
```

Out[178]:

```
['4.4',
 '4',
 '4.1',
 '4.4',
 '4.2',
 '3.8',
 '3.9',
 '4.2',
 '4',
 '4.1',
 '4.3',
 '4.4',
 '4.2',
 '4.4',
 '3.9',
 '4',
 '4.4',
 '4.4',
 '3.9',
 '4.4']
```

In [180]:

```
image_url =[] # empty list for store the

for i in soup.find_all('img', class_="no-img"):
    image_url.append(i['data-src'])

image_url
```

Out[180]:

```
['https://im1.dineout.co.in/images/uploads/restaurant/sharpen/4/h/o/p42484-1
5683760515d7b84f35d214.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/8/x/t/p83921-1
6017104805f782990b9b0c.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/n/f/p2675-14
996644935963106d60179.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/n/e/p2620-15
610295235d0b6b9397635.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/a/n/p281-163
108768261386c42106da.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/k/r/p282-156
94783585d8c56d6dce0f.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/w/e/p2900-15
610290405d0b69b0c589b.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/4/x/k/p48129-1
5444251565c0e0ec4af17e.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/h/o/p2872-14
86443948589955acc29bb.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/5/a/q/p50537-1
5610295535d0b6bb19f759.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/t/x/p2899-16
017267115f7868f7ee080.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/3/r/i/p3142-14
6279424557307805d39e1.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/1/q/u/p10239-1
44652724156384109a7ad7.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/5/e/e/p52262-1
5780350315e0ee757c2bf8.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/9/w/n/p923-164
085436561cd735d2ab07.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/3/h/c/p3643-14
4497865356209fdd65746.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/4/m/f/p49957-1
5478931165c42f97cd6440.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/6/r/g/p63049-1
5804707965e34120cdf620.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/3/i/e/p3070-16
046564945fa51d6e8a014.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/3/x/i/p34490-1
502548927598f13bf9a466.jpg?tr=tr:n-medium',
 'https://im1.dineout.co.in/images/uploads/restaurant/sharpen/2/d/n/p2847-16
046575895fa521b5bd5ee.jpg?tr=tr:n-medium']
```

In [182]:

```
import pandas as pd
df=pd.DataFrame({'Restaurant_name':Restaurant_name,'Cuisine':Cuisine,'location':location,'i
df
```

Out[182]:

	Restaurant_name	Cuisine	location	image_
0	The Imperial Spice	₹ 3,000 for 2 (approx) | North Indian, Chinese...	M-Block,Connaught Place, Central Delhi	https://im1.dineout.co.in/images/uploads/rest
1	Out Of The Box Courtyard	₹ 2,200 for 2 (approx) | North Indian, Mediter...	Connaught Place, Central Delhi	https://im1.dineout.co.in/images/uploads/rest
2	The Great Kabab Factory	₹ 3,500 for 2 (approx) | North Indian, Finger ...	Radisson Blu Marina,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
3	Baluchi	₹ 4,500 for 2 (approx) | North Indian, Mughlai	The Lalit New Delhi,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
4	The Connaught Bar	₹ 3,400 for 2 (approx) | Finger Food	Radisson Blu Marina,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
5	Fifty9	₹ 2,900 for 2 (approx) | North Indian, Contine...	Radisson Blu Marina,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
6	24/7	₹ 4,800 for 2 (approx) | Continental, North In...	The Lalit New Delhi,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
7	OKO	₹ 5,200 for 2 (approx) | Asian, Chinese, Thai,...	The Lalit New Delhi,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
8	The Grill Room	₹ 5,000 for 2 (approx) | Continental, European...	The Lalit New Delhi,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
9	The Lalit Boulangerie	₹ 1,000 for 2 (approx) | Desserts	The Lalit New Delhi,Connaught Place, Central D...	https://im1.dineout.co.in/images/uploads/rest
10	Out Of The Box	₹ 2,200 for 2 (approx) | North Indian, Italian...	Khan Market, Central Delhi	https://im1.dineout.co.in/images/uploads/rest
11	Veg Gulati	₹ 1,500 for 2 (approx) | North Indian	Pandara Road, Central Delhi	https://im1.dineout.co.in/images/uploads/rest
12	Yellow Brick Road	₹ 3,500 for 2 (approx) | Continental, North In...	Ambassador,Khan Market, Central Delhi	https://im1.dineout.co.in/images/uploads/rest

	Restaurant_name	Cuisine	location	image_
13	G2 Cafe	₹ 1,600 for 2 (approx) | Continental, North In...	Anand Vihar, East Delhi	https://im1.dineout.co.in/images/uploads/rest
14	Larry's China	₹ 3,500 for 2 (approx) | Chinese, Seafood	Ambassador,Khan Market, Central Delhi	https://im1.dineout.co.in/images/uploads/rest
15	Jungle Jamboree	₹ 1,400 for 2 (approx) | North Indian, Asian, ...	3CS Mall,Lajpat Nagar - 3, South Delhi	https://im1.dineout.co.in/images/uploads/rest
16	Antidot Waterbar and Cafe	₹ 1,800 for 2 (approx) | Asian, North Indian, ...	Safdarjung Enclave Market,Safdarjung, South Delhi	https://im1.dineout.co.in/images/uploads/rest
17	Veg Gulati	₹ 1,200 for 2 (approx) | North Indian	Green Park, South Delhi	https://im1.dineout.co.in/images/uploads/rest
18	TK's Oriental Grill	₹ 3,400 for 2 (approx) | Japanese, Indonesian,...	Hyatt Regency Delhi,Bhikaji Cama Place, South ...	https://im1.dineout.co.in/images/uploads/rest
19	The Construction Co. Cafe	₹ 1,200 for 2 (approx) | Chinese, Italian, Con...	Satya Niketan, South Delhi	https://im1.dineout.co.in/images/uploads/rest
20	The China Kitchen	₹ 4,500 for 2 (approx) | Chinese, Desserts	Hyatt Regency Delhi,Bhikaji Cama Place, South ...	https://im1.dineout.co.in/images/uploads/rest

In [ ]:

```

```

In [ ]:

```

```

10) Write a python program to scrape the details of top publications from Google Scholar from

https://scholar.google.com/citations?view_op=top_venues&hl=en (https://scholar.google.com/citations?view_op=top_venues&hl=en)

i) Rank

ii) Publication

iii) h5-index

iv) h5-median

In [183]:

```
page=requests.get('https://scholar.google.com/citations?view_op=top_venues&hl=en')
page
```

Out[183]:

```
<Response [200]>
```

In [184]:

```
soup=BeautifulSoup(page.content)
soup
```

```
t:100%;min-width:964px;-webkit-tap-highlight-color:rgba(0,0,0,0);}#gs_top>
*:not(#x){-webkit-tap-highlight-color:rgba(204,204,204,.5);}.gs_el_ph #gs_
top,.gs_el_ta #gs_top{min-width:320px;}#gs_top.gs_nscl{position:fixed;widt
h:100%;}body,td,input,button{font-size:13px;font-family:Arial,sans-serif;l
ine-height:1.24;}body{background:#fff;color:#222;-webkit-text-size-adjust:
100%;-moz-text-size-adjust:none;}body{background-color:#f9f9f9;}.gs_gray{c
olor:#777777}.gs_red{color:#dd4b39}.gs_grn{color:#006621}.gs_lil{font-siz
e:11px}.gs_med{font-size:16px}.gs_hlt{font-weight:bold;}a:link{color:#1a0d
ab;text-decoration:none}a:visited{color:#660099;text-decoration:none}a:hov
er,a:hover .gs_lbl{text-decoration:underline}a:active,a:active .gs_lbl,a .
gs_lbl:active{color:#d14836}.gs_el_tc a:hover,.gs_el_tc a:hover .gs_lbl{te
xt-decoration:none}.gs_pfcs a:focus,.gs_pfcs button:focus,.gs_pfcs input:f
ocus,.gs_pfcs label:focus{outline:none}.gs_a,.gs_a a:link,.gs_a a:visited
{color:#006621}.gs_a a:active{color:#d14836}a.gs_fl:link,.gs_fl a:link{col
or:#1a0dab}a.gs_fl:visited,.gs_fl a:visited{color:#660099}a.gs_fl:active,.
gs_fl a:active{color:#d14836}.gs_fl{color:#777777}.gs_ctc,.gs_ctu{vertical
-align:middle;font-size:11px;font-weight:bold}.gs_ctc{color:#1a0dab}.gs_ct
g,.gs_ctg2{font-size:13px;font-weight:bold}.gs_ctg{color:#1a0dab}a.gs_pd
a,.gs_pda a{padding:7px 0 5px 0}.gs_alrt{background:#f9edbe;border:1px sol
```

In [ ]:

```
```

In [185]:

```
rank=soup.find('th', class_="gsc_mvt_p")
rank.text
```

Out[185]:

```
''
```

In [186]:

```
publication=soup.find('td', class_="gsc_mvt_t")
publication.text
```

Out[186]:

```
'Nature'
```

In [187]:

```
h_5_index=soup.find('td', class_="gsc_mvt_n")
h_5_index.text
```

Out[187]:

```
'414'
```

In [189]:

```
h_5_median=soup.find('span', class_="gs_ibl gsc_mp_anchor")
h_5_median.text
```

Out[189]:

```
'607'
```

In [ ]:

```

```

In [190]:

```
rank =[] # empty list for store the

for i in soup.find_all('th', class_="gsc_mvt_p"):
    rank.append(i.text)

rank
```

Out[190]:

```
['']
```

In [191]:

```python
publication =[] # empty list for store the

for i in soup.find_all('td', class_="gsc_mvt_t"):
    publication.append(i.text)

publication
```

Out[191]:

```
['Nature',
 'The New England Journal of Medicine',
 'Science',
 'IEEE/CVF Conference on Computer Vision and Pattern Recognition',
 'The Lancet',
 'Advanced Materials',
 'Cell',
 'Nature Communications',
 'Chemical Reviews',
 'International Conference on Learning Representations',
 'JAMA',
 'Neural Information Processing Systems',
 'Proceedings of the National Academy of Sciences',
 'Journal of the American Chemical Society',
 'Angewandte Chemie',
 'Chemical Society Reviews',
 'Nucleic Acids Research',
 'Renewable and Sustainable Energy Reviews',
```

In [192]:

```python
h_5_index =[] # empty list for store the

for i in soup.find_all('td', class_="gsc_mvt_n"):
    h_5_index.append(i.text)

h_5_index
```

```
 '257',
 '161',
 '244',
 '161',
 '239',
 '160',
 '244',
 '160',
 '203',
 '159',
 '252',
 '159',
 '212',
 '158',
 '246',
 '158',
 '237',
 '157',
 '265',
 '157'
```

In [210]:

```
h_5_index1=h_5_index[0:100]
h_5_index1
```

Out[210]:

```
['414',
 '607',
 '410',
 '704',
 '391',
 '564',
 '356',
 '583',
 '345',
 '600',
 '294',
 '406',
 '288',
 '459',
 '287',
 '389',
 '270',
 '434',
```

In [211]:

```
len(h_5_index1)
```

Out[211]:

```
100
```

In [200]:

```
h_5_median =[] # empty list for store the

for i in soup.find_all('span', class_="gs_ibl gsc_mp_anchor"):
    h_5_median.append(i.text)

h_5_median
```

...

In [205]:

```
h_5_median1=h_5_median[0:102]
h_5_median1
```

```
 '434',
 '470',
 '446',
 '422',
 '337',
 '330',
 '314',
 '339',
 '512',
 '294',
 '297',
 '297',
 '267',
 '356',
 '370',
 '306',
 '265',
 '262',
 '342',
```

In [208]:

```
len(h_5_median1)
```

Out[208]:

```
100
```

In [212]:

```
df=pd.DataFrame({'publication':publication,'h_5_index':h_5_index1,'h_5_median':h_5_median1,
df
```

Out[212]:

	publication	h_5_index	h_5_median
0	Nature	414	607
1	The New England Journal of Medicine	607	704
2	Science	410	564
3	IEEE/CVF Conference on Computer Vision and Pat...	704	583
4	The Lancet	391	600
...	...	...	...
95	Frontiers in Immunology	265	177
96	Small	161	173
97	Nature Immunology	257	210
98	JAMA Oncology	161	202
99	The Lancet Neurology	244	200

100 rows × 3 columns

In [ ]:

```
```