Web Scrapping Assignment-4

In [ ]:
```
# Importing Libraries
import selenium
import pandas as pd
import time
from bs4 import BeautifulSoup

# Importing selenium webdriver
from selenium import webdriver

# Importing required Exceptions which needs to handled
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException

#Importing requests
import requests

# importing regex
import re
```

1. Scrape the details of most viewed videos on YouTube from Wikipedia. Url = https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos You need to find following details: A) Rank B) Name C) Artist D) Upload date E) Views

In [ ]:
```
# Activating the chrome browser
driver = webdriver.Chrome()
```

In [ ]:
```
url_1="https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos/"
```

In [ ]:
```
driver.get(url_1)
```

In [ ]:
```
Rank=[]
Name=[]
Artist=[]
Upload_Date=[]
Views=[]
```

In [ ]:
```
search_1 = driver.find_element_by_xpath('//*[@id="noarticletext"]/tbody/tr/td/span/a')       # Locating page fo
search_1.click()
```

In [ ]:
```
search_2 = driver.find_element_by_xpath('//*[@id="mw-content-text"]/div[3]/ul/li[1]/div[1]/a')       # Locating
search_2.click()
```

In [ ]:
```
#scraping the Rank
rk=driver.find_elements_by_xpath("//*[@id='mw-content-text']/div[1]/table[1]/tbody/tr/td[1]")
for i in rk:
    if i.text is None :
        Rank.append("--")
    else:
        Rank.append(i.text)
print(len(Rank),Rank)
```

In [ ]:
```
#scraping the Video Name
nm=driver.find_elements_by_xpath("//*[@id='mw-content-text']/div[1]/table[1]/tbody/tr/td[2]")
for i in nm:
    if i.text is None :
        Name.append("--")
    else:
        Name.append(i.text)
print(len(Name),Name)
```

In [ ]:
```
#scraping the Artist
Ar=driver.find_elements_by_xpath("//*[@id='mw-content-text']/div[1]/table[1]/tbody/tr/td[3]")
for i in Ar:
    if i.text is None :
        Artist.append("--")
    else:
        Artist.append(i.text)
print(len(Artist),Artist)
```

In [ ]:
```
#scraping the Upload_Date
date=driver.find_elements_by_xpath("//*[@id='mw-content-text']/div[1]/table[1]/tbody/tr/td[5]")
for i in date:
    if i.text is None :
        Upload_Date.append("--")
    else:
        Upload_Date.append(i.text)
print(len(Upload_Date),Upload_Date)
```

In [ ]:
```
#scraping the Views
```

```
v=driver.find_elements_by_xpath("//*[@id='mw-content-text']/div[1]/table[1]/tbody/tr/td[4]")
for i in v:
    if i.text is None :
        Views.append("--")
    else:
        Views.append(i.text)
print(len(Views),Views)
```

In [ ]:
```
Youtube_Video=pd.DataFrame([])
Youtube_Video['Rank']=Rank
Youtube_Video['Video Title']=Name
Youtube_Video['Artist']=Artist
Youtube_Video['Upload_Date']=Upload_Date
Youtube_Video['Views In Bllion']=Views
Youtube_Video
```

1. Scrape the details team India's international fixtures from bcci.tv. Url = https://www.bcci.tv/. You need to find following details: A) Series B) Place C) Date D) Time

In [ ]:
```
url_2="https://www.bcci.tv/international/fixtures"
```

In [ ]:
```
driver.get(url_2)
```

In [ ]:
```
Team=[]
Date_Time =[]
Ground=[]
Test_ODI=[]
```

In [ ]:
```
#scraping the company_name
tm=driver.find_elements_by_xpath("//div[@class='fixture__teams']")
for i in tm:
    if i.text is None :
        Team.append("--")
    else:
        Team.append(i.text)
print(len(Team),Team)
```

In [ ]:
```
#scraping the Date_Time
DT=driver.find_elements_by_xpath("//span[@class='fixture__datetime tablet-only']")
for i in DT:
    if i.text is None :
        Date_Time.append("--")
    else:
        Date_Time.append(i.text)
print(len(Date_Time),Date_Time)
```

In [ ]:
```
#scraping the Ground
G=driver.find_elements_by_xpath("//p[@class='fixture__additional-info']/span")
for i in G:
    if i.text is None :
        Ground.append("--")
    else:
        Ground.append(i.text)
print(len(Ground),Ground)
```

In [ ]:
```
#scraping the Test_ODI
TO=driver.find_elements_by_xpath("//p[@class='fixture__additional-info']/strong")
for i in TO:
    if i.text is None :
        Test_ODI.append("--")
    else:
        Test_ODI.append(i.text)
print(len(Test_ODI),Test_ODI)
```

In [ ]:
```
International_Fixtures=pd.DataFrame([])
International_Fixtures['Team']=Team
International_Fixtures['Date_Time']=Date_Time
International_Fixtures['Series']=Test_ODI
International_Fixtures['Ground']=Ground
International_Fixtures
```

1. Scrape the details of State-wise GDP of India from statisticstime.com. Url = http://statisticstimes.com/ You have to find following details: A) Rank B) State C) GSDP(18-19)- at current prices D) GSDP(19-20)- at current prices E) Share(18-19) F) GDP($ billion)

In [ ]:
```
url_4="http://statisticstimes.com/"
```

In [ ]:
```
driver.get(url_4)
```

In [ ]:
```
Rank=[]
State =[]
```

```
GDP=[]
GSDP_Current=[]
GSDP_Previous=[]
Share=[]
```

In [ ]:
```
economy = driver.find_element_by_xpath('//*[@id="top"]/div[2]/div[2]/button')       # Locating page foe top vid
economy.click()
```

In [ ]:
```
ind = driver.find_element_by_xpath('//*[@id="top"]/div[2]/div[2]/div/a[3]')       # Locating page foe top video
ind.click()
```

In [ ]:
```
gdp = driver.find_element_by_xpath('/html/body/div[2]/div[2]/div[2]/ul/li[1]/a')       # Locating page foe top
gdp.click()
```

In [ ]:
```
#scraping the Rank
r=driver.find_elements_by_xpath("//td[@class='data1']")
for i in r:
    if i.text is None :
        Rank.append("--")
    else:
        Rank.append(i.text)
print(len(Rank),Rank)
```

In [ ]:
```
#scraping the State
St=driver.find_elements_by_xpath("//td[@class='name']")
for i in St:
    if i.text is None :
        State.append("--")
    else:
        State.append(i.text)
print(len(State),State)
```

In [ ]:
```
#scraping the GDP
gdp=driver.find_elements_by_xpath("//*[@id='table_id']/tbody/tr/td[6]")
for i in gdp:
    if i.text is None :
        GDP.append("--")
    else:
        GDP.append(i.text)
print(len(GDP),GDP)
```

In [ ]:
```
#scraping the Share
shr=driver.find_elements_by_xpath("//*[@id='table_id']/tbody/tr/td[5]")
for i in shr:
    if i.text is None :
        Share.append("--")
    else:
        Share.append(i.text)
print(len(Share),Share)
```

In [ ]:
```
#scraping the GSDP_Current
shr=driver.find_elements_by_xpath("//*[@id='table_id']/tbody/tr/td[4]")
for i in shr:
    if i.text is None :
        GSDP_Current.append("--")
    else:
        GSDP_Current.append(i.text)
print(len(GSDP_Current),GSDP_Current)
```

In [ ]:
```
#scraping the GSDP_Previous
shr=driver.find_elements_by_xpath("//*[@id='table_id']/tbody/tr/td[8]")
for i in shr:
    if i.text is None :
        GSDP_Previous.append("--")
    else:
        GSDP_Previous.append(i.text)
print(len(GSDP_Previous),GSDP_Previous)
```

In [ ]:
```
State_GDP=pd.DataFrame([])
State_GDP['Rank']=Rank[:33]
State_GDP['State']=State[:33]
State_GDP['Share In GDP']=Share[:33]
State_GDP['GDP of State']=GDP[:33]
State_GDP['GSDP_Current']=GSDP_Current[:33]
State_GDP['GSDP_Previous']=GSDP_Previous[:33]
State_GDP
```

1. Scrape the details of trending repositories on Github.com. Url = https://github.com/ You have to find the following details: A) Repository title B) Repository description C) Contributors count D) Language used

In [ ]:
```
url_5="https://github.com/"
```

In [ ]:
```
driver.get(url_5)
```

In [ ]:
```
driver.get(url_5)
```

In [ ]:
```
Repository_Name =[]
Language=[]
Muted_Link=[]
```

In [ ]:
```
explore = driver.find_element_by_xpath('/html/body/div[1]/header/div/div[2]/nav/ul/li[4]/details')       # Loca
explore.click()
```

In [ ]:
```
trending = driver.find_element_by_xpath('/html/body/div[1]/header/div/div[2]/nav/ul/li[4]/details/div/ul[2]/li[
trending.click()
```

In [ ]:
```
#scraping the Repositor_Name
RN=driver.find_elements_by_xpath("//span[@class='text-normal']")
for i in RN:
    if i.text is None :
        Repository_Name.append("--")
    else:
        Repository_Name.append(i.text)
print(len(Repository_Name),Repository_Name)
```

In [ ]:
```
Description=[]
#scraping the Description
des=driver.find_elements_by_xpath("//p[@class='col-9 text-gray my-1 pr-4']")
for i in des:
    if i.text is None :
        Description.append("--")
    else:
        Description.append(i.text)
print(len(Description),Description)
```

In [ ]:
```
#scraping the Language
L=driver.find_elements_by_xpath("//span[@itemprop='programmingLanguage']")
for i in L:
    if i.text is None :
        Language.append("NAN")
    else:
        Language.append(i.text)
print(len(Language),Language)
```

In [ ]:
```
#scraping the Muted_Link And Star
ml=driver.find_elements_by_xpath("//a[@class='muted-link d-inline-block mr-3']")
for i in ml:
    if i.text is None :
        Muted_Link.append("NAN")
    else:
        Muted_Link.append(i.text)
print(len(Muted_Link),Muted_Link)
```

In [ ]:
```
Muted=[]
for i in range(1,len(Muted_Link),2):
    Muted.append(Muted_Link[i])

print(len(Muted),Muted)
```

In [ ]:
```
Muted_Star=[]
for i in range(0,len(Muted_Link),2):
    Muted_Star.append(Muted_Link[i])

print(len(Muted_Star),Muted_Star)
```

In [ ]:
```
Trending_Repository=pd.DataFrame([])
Trending_Repository['Name']=Repository_Name[:22]
Trending_Repository['Description']=Description[:22]
Trending_Repository['Language']=Language[:22]
Trending_Repository['Conutrybuted']=Muted[:22]
Trending_Repository['Muted_Star']=Muted_Star[:22]
Trending_Repository
```

1. Scrape the details of top 100 songs on billiboard.com. Url = https:/www.billboard.com/ You have to find the following details: A) Song name B) Artist name C) Last week rank D) Peak rank E) Weeks on board

In [ ]:
```
url_6="https://www.billboard.com/"
```

In [ ]:
```
driver.get(url_6)
```

In [ ]:
```
Song_Name =[]
Singer=[]
rank=[]
Last_Week=[]
Weeks_on_board=[]
```

In [ ]:
```
top100 = driver.find_element_by_xpath('//*[@id="root"]/div[2]/div[2]/nav/ul/li[2]')       # Locating page for t
```

```
top100 = driver.find_element_by_xpath('//*[@id="root"]/div[2]/div[2]/nav/ul/li[3]')        # Locating page Toe t
top100.click()
```

```
#scraping the Rank
rb=driver.find_elements_by_xpath("//div[@class='chart-element__meta text--center color--secondary text--peak']"
for i in rb:
    if i.text is None :
        rank.append("--")
    else:
        rank.append(i.text)
print(len(rank),rank)
```

```
#scraping the Song_Name
son=driver.find_elements_by_xpath("//span[@class='chart-element__information__song text--truncate color--primar
for i in son:
    if i.text is None :
        Song_Name.append("--")
    else:
        Song_Name.append(i.text)
print(len(Song_Name),Song_Name)
```

```
#scraping the Singer
sin=driver.find_elements_by_xpath("//span[@class='chart-element__information__artist text--truncate color--seco
for i in sin:
    if i.text is None :
        Singer.append("--")
    else:
        Singer.append(i.text)
print(len(Singer),Singer)
```

```
#scraping the Last_Week Rank
lwr=driver.find_elements_by_xpath("//div[@class='chart-element__meta text--center color--secondary text--last']
for i in lwr:
    if i.text is None :
        Last_Week.append("--")
    else:
        Last_Week.append(i.text)
print(len(Last_Week),Last_Week)
```

```
#scraping the Weeks_on_board
wob=driver.find_elements_by_xpath("//div[@class='chart-element__meta text--center color--secondary text--week']
for i in wob:
    if i.text is None :
        Weeks_on_board.append("--")
    else:
        Weeks_on_board.append(i.text)
print(len(Weeks_on_board),Weeks_on_board)
```

```
Top_Song=pd.DataFrame([])
Top_Song['Peak_rank']=rank
Top_Song['Song_Name']=Song_Name
Top_Song['Singer / Crew']=Singer
Top_Song['Last_Week_Rank']=Last_Week
Top_Song['Weeks_on_board']=Weeks_on_board
Top_Song
```

1. Scrape the details of Highest selling novels. A) Book name B) Author name C) Volumes sold D) Publisher E) Genre

```
url_8="https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-time-fifty-shades-grey-comp
```

```
driver.get(url_8)
```

```
Book_name=[]
Author_name=[]
Volumes_sold=[]
Publisher=[]
Genre=[]
```

```
#scraping the Book_name
bname=driver.find_elements_by_xpath("/html/body/div/div[2]/div[2]/div/div[2]/div/table/tbody/tr/td[2]")
for i in bname:
    if i.text is None :
        Book_name.append("--")
    else:
        Book_name.append(i.text)
print(len(Book_name),Book_name)
```

```
#scraping the Author_name
Auth=driver.find_elements_by_xpath("/html/body/div/div[2]/div[2]/div/div[2]/div/table/tbody/tr/td[3]")
for i in Auth:
    if i.text is None :
        Author_name.append("--")
    else:
```

```
        Author_name.append(i.text)
print(len(Author_name),Author_name)
```

In [ ]:
```
#scraping the Genre
gen=driver.find_elements_by_xpath("/html/body/div/div[2]/div[2]/div/div[2]/div/table/tbody/tr/td[6]")
for i in gen:
    if i.text is None :
        Genre.append("--")
    else:
        Genre.append(i.text)
print(len(Genre),Genre)
```

In [ ]:
```
#scraping the Publisher
pub=driver.find_elements_by_xpath("/html/body/div/div[2]/div[2]/div/div[2]/div/table/tbody/tr/td[5]")
for i in pub:
    if i.text is None :
        Publisher.append("--")
    else:
        Publisher.append(i.text)
print(len(Publisher),Publisher)
```

In [ ]:
```
#scraping the Volumes_sold
vs=driver.find_elements_by_xpath("/html/body/div/div[2]/div[2]/div/div[2]/div/table/tbody/tr/td[4]")
for i in vs:
    if i.text is None :
        Volumes_sold.append("--")
    else:
        Volumes_sold.append(i.text)
print(len(Volumes_sold),Volumes_sold)
```

In [ ]:
```
Book=pd.DataFrame([])
Book['Book_name']=Book_name
Book['Author_name']=Author_name
Book['Genre']=Genre
Book['Publisher']=Publisher
Book['Volumes_sold']=Volumes_sold
Book
```

1. Scrape the details most watched tv series of all time from imdb.com. Url = https://www.imdb.com/list/ls095964455/ You have to find the following details: A) Name B) Year span C) Genre D) Run time E) Ratings F) Votes

In [ ]:
```
url_9="https://www.imdb.com/list/ls095964455/"
```

In [ ]:
```
driver.get(url_9)
```

In [ ]:
```
Name=[]
Year_span=[]
Genres=[]
Run_time=[]
Ratings=[]
Votes=[]
```

In [ ]:
```
#scraping the Name
mname=driver.find_elements_by_xpath("//div[@class='lister-item-content']/h3/a")
for i in mname:
    if i.text is None :
        Name.append("--")
    else:
        Name.append(i.text)
print(len(Name),Name)
```

In [ ]:
```
#scraping the Year_span
ys=driver.find_elements_by_xpath("//span[@class='lister-item-year text-muted unbold']")
for i in ys:
    if i.text is None :
        Year_span.append("--")
    else:
        Year_span.append(i.text)
print(len(Year_span),Year_span)
```

In [ ]:
```
#scraping the Genres
gnr=driver.find_elements_by_xpath("//p[@class='text-muted text-small']/span[5]")
for i in gnr:
    if i.text is None :
        Genres.append("--")
    else:
        Genres.append(i.text)
print(len(Genres),Genres)
```

In [ ]:
```
#scraping the Run_time
rt=driver.find_elements_by_xpath("//p[@class='text-muted text-small']/span[3]")
for i in rt:
```

```
    if i.text is None :
        Run_time.append("--")
    else:
        Run_time.append(i.text)
print(len(Run_time),Run_time)
```

In [ ]:
```
#scraping the Votes
v=driver.find_elements_by_xpath("//div[@class='lister-item-content']/p[4]/span[2]")
for i in v:
    if i.text is None :
        Votes.append("--")
    else:
        Votes.append(i.text)
print(len(Votes),Votes)
```

In [ ]:
```
TV_Series=pd.DataFrame([])
TV_Series['Name']=Name
TV_Series['Year_span']=Year_span
TV_Series['Run_time']=Run_time
TV_Series['Genres']=Genres
TV_Series['Ratings']=Ratings
TV_Series['Votes']=Votes
TV_Series
```

1. Details of Datasets from UCI machine learning repositories. Url = https://archive.ics.uci.edu/ You have to find the following details: A) Dataset name B) Data type C) Task D) Attribute type E) No of instances F) No of attribute G) Year

In [ ]:
```
url_10="https://archive.ics.uci.edu/ml/index.php"
```

In [ ]:
```
driver.get(url_10)
```

In [ ]:
```
Dataset_Name=[]
Data_Type=[]
Task=[]
Attribute_Type=[]
No_of_Instances=[]
No_of_Attribute=[]
Year=[]
```

In [ ]:
```
search_1 = driver.find_element_by_xpath('/html/body/table[1]/tbody/tr/td[2]/span[2]/a')       # Locating page f
search_1.click()
```

In [ ]:
```
#scraping the Dataset_Name
dname=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[1]/table/tbody/tr/
for i in dname:
    if i.text is None :
        Dataset_Name.append("--")
    else:
        Dataset_Name.append(i.text)
print(len(Dataset_Name),Dataset_Name)
```

In [ ]:
```
#scraping the Data_Type
dtype=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[2]")
for i in dtype:
    if i.text is None :
        Data_Type.append("--")
    else:
        Data_Type.append(i.text)
print(len(Data_Type),Data_Type)
```

In [ ]:
```
#scraping the Task
t=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[3]")
for i in t:
    if i.text is None :
        Task.append("--")
    else:
        Task.append(i.text)
print(len(Task),Task)
```

In [ ]:
```
#scraping the Attribute_Type
att=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[4]")
for i in att:
    if i.text is None :
        Attribute_Type.append("--")
    else:
        Attribute_Type.append(i.text)
print(len(Attribute_Type),Attribute_Type)
```

In [ ]:
```
#scraping the No_of_Instances
noi=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[5]")
for i in noi:
    if i.text is None :
```

```
        No_of_Instances.append("--")
    else:
        No_of_Instances.append(i.text)
print(len(No_of_Instances),No_of_Instances)
```

In [ ]:
```
#scraping the No_of_Attribute
noa=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[6]")
for i in noa:
    if i.text is None :
        No_of_Attribute.append("--")
    else:
        No_of_Attribute.append(i.text)
print(len(No_of_Attribute),No_of_Attribute)
```

In [ ]:
```
#scraping the Year
y=driver.find_elements_by_xpath("/html/body/table[2]/tbody/tr/td[2]/table[2]/tbody/tr/td[7]/p")
for i in y:
    if i.text is None :
        Year.append("--")
    else:
        Year.append(i.text)
print(len(Year),Year)
```

In [ ]:
```
UCI=pd.DataFrame([])
UCI['Dataset_Name']=Dataset_Name[:100]
UCI['Data_Type']=Data_Type[:100]
UCI['Task']=Task[:100]
UCI['Attribute_Type']=Attribute_Type[:100]
UCI['No_of_Instances']=No_of_Instances[:100]
UCI['No_of_Attribute']=No_of_Attribute[:100]
UCI['Year']=Year[:100]
UCI
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js