```
In [ ]: # WEB SCRAPING - ASSIGNMENT 4
```

```
In [1]: # let's first install the selenium library
        ! pip install selenium
```

```
Requirement already satisfied: selenium in c:\users\gaura\anaconda3\lib\site-packages
(4.17.2)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in c:\users\gaura\anaconda3\li
b\site-packages (from selenium) (1.26.16)
Requirement already satisfied: trio~=0.17 in c:\users\gaura\anaconda3\lib\site-packag
es (from selenium) (0.24.0)
Requirement already satisfied: trio-websocket~=0.9 in c:\users\gaura\anaconda3\lib\si
te-packages (from selenium) (0.11.1)
Requirement already satisfied: certifi>=2021.10.8 in c:\users\gaura\anaconda3\lib\sit
e-packages (from selenium) (2023.7.22)
Requirement already satisfied: typing_extensions>=4.9.0 in c:\users\gaura\anaconda3\l
ib\site-packages (from selenium) (4.9.0)
Requirement already satisfied: attrs>=20.1.0 in c:\users\gaura\anaconda3\lib\site-pac
kages (from trio~=0.17->selenium) (22.1.0)
Requirement already satisfied: sortedcontainers in c:\users\gaura\anaconda3\lib\site-
packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in c:\users\gaura\anaconda3\lib\site-packages (fr
om trio~=0.17->selenium) (3.4)
Requirement already satisfied: outcome in c:\users\gaura\anaconda3\lib\site-packages
(from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied: sniffio>=1.3.0 in c:\users\gaura\anaconda3\lib\site-pa
ckages (from trio~=0.17->selenium) (1.3.0)
Requirement already satisfied: cffi>=1.14 in c:\users\gaura\anaconda3\lib\site-packag
es (from trio~=0.17->selenium) (1.15.1)
Requirement already satisfied: wsproto>=0.14 in c:\users\gaura\anaconda3\lib\site-pac
kages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in c:\users\gaura\anaconda
3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied: pycparser in c:\users\gaura\anaconda3\lib\site-package
s (from cffi>=1.14->trio~=0.17->selenium) (2.21)
Requirement already satisfied: h11<1,>=0.9.0 in c:\users\gaura\anaconda3\lib\site-pac
kages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.14.0)
```

```
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, NoSuchElementEx

        #Importing requests
        import requests

        # importing regex
        import re
```

```
In [ ]: 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 fo
```

```
Rank
B) Name
C) Artist
D) Upload date
E) Views
```

In [ ]:
```
# let's first connect to the web driver
driver = webdriver.Chrome(r"C:\Users\gaurav\Downloads\chromedriver_win32\chromedriver.
```

In [ ]:
```
url="https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos/"
driver.get(url)
```

In [2]:
```
search_bar=driver.find_element_by_xpath('//input[@type="search"]')
search_bar
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 search_bar=driver.find_element_by_xpath('//input[@type="search"]')
      2 search_bar

NameError: name 'driver' is not defined
```

In [ ]:
```
search_bar.send_keys('List_of_most-viewed_YouTube_videos')
```

In [ ]:
```
search_btn= driver .find_element_by_xpath("//input[@type='submit']")
search_btn.click()
```

In [ ]:
```
#click on the first option
L1=driver.find_element_by_xpath("//span[@class='searchmatch'] ")
L1.click()
```

In [ ]:
```
#make empty list
Rank=[]
Name=[]
Artist=[]
Upload_date=[]
Views=[]
```

In [ ]:
```
# extracting rank from xpath

rank=driver.find_elements_by_xpath('//td[@align="center"]')
for i in rank:
    Rank.append(i.text)
Rank
```

In [ ]:
```
int=[]
for j in range(0,len(Rank),3):
    int.append(Rank[i])
int
```

In [ ]:
```
# Extracting name from xpath
try:
    vname=driver.find_elements_by_xpath("//a[@href='/wiki/Baby_Shark_Dance']")
    Name.append(vname.text)
```

```
except NoSuchElementException:
    Name.append("-")
```

In [ ]:
```
driver.close()
```

In [ ]:
```
2. 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
Note: - From bcci.tv home page you have reach to the international fixture page throug
```

In [ ]:
```
# let's first connect to the web driver
driver = webdriver.Chrome(r"C:\Users\Gaurav\Downloads\chromedriver_win32\chromedriver.
```

In [ ]:
```
url = "https://www.bcci.tv/"
driver.get(url)
```

In [ ]:
```
in_btn=driver.find_element_by_xpath('//div[@class="navigation__drop-down drop-down drc
in_btn.click()
```

In [ ]:
```
fix_btn=driver.find_element_by_xpath('//a[@class="navigation__link navigation__link--i
fix_btn.click()
```

In [ ]:
```
Match_title=[]
Series=[]
Place=[]
Date=[]
D_month=[]
Time=[]
```

In [3]:
```
series=driver.find_elements_by_xpath("//span[@class='u-unskewed-text fixture__format']
for i in series:
    if i.text is None:
        Series.append('Not')
    else:
        Series.append(i.text)
Series
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 series=driver.find_elements_by_xpath("//span[@class='u-unskewed-text fixture_
_format']")
      2 for i in series:
      3     if i.text is None:

NameError: name 'driver' is not defined
```

In [ ]:
```
title=driver.find_elements_by_xpath("//span[@class='u-unskewed-text fixture__tournamer
for j in title:
    Match_title.append(j.text)
Match_title
```

In [ ]:
```
place=driver.find_elements_by_xpath("//p[@class='fixture__additional-info']")

for k in place:
    Place.append(k.text)
Place
```

In [ ]:
```
date=driver.find_elements_by_xpath("//span[@class='fixture__date']")

for l in date:
    Date.append(l.text)
Date
```

In [ ]:
```
month=driver.find_elements_by_xpath("//span[@class='fixture__month']")
for m in month:
    D_month.append(m.text)
D_month
```

In [ ]:
```
bcc_df=pd.DataFrame({'M_Series':Series,'Title of match':Match_title,'M Place':Place,'M
```

In [ ]:
```
bcc_df
```

In [ ]:
```
driver.close()
```

In [ ]:
```
3. 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)
Note: - From statisticstimes home page you have to reach to economy page through code.
```

In [ ]:
```
# let's first connect to the web driver
driver = webdriver.Chrome(r"C:\Users\Gaurav\Downloads\chromedriver_win32\chromedriver.
```

In [ ]:
```
url ="http://statisticstimes.com"
driver.get(url)
```

In [ ]:
```
btn_e=driver.find_element_by_xpath("//button[@class='dropbtn']")
btn_e.click()
```

In [ ]:
```
Urls=[]
e_india=driver.find_elements_by_xpath("//a[@href='economy/india-statistics.php']")
```

In [ ]:
```
[]
```

In [ ]:
```
driver.close()
```

In [ ]:
```
4. Scrape the details of top 100 songs on billiboard.com. Url = https:/www.billboard.c
following details:
A) Song name
B) Artist name
C) Last week rank
```

```
D) Peak rank
E) Weeks on board
 Note: - From the home page you have to click on the charts option then hot 100-page l
```

In [ ]:
```
# import the necessary libraries:
import requests
from bs4 import BeautifulSoup
```

In [ ]:
```
# Send a GET request to the Billboard Hot 100 page:
url = "https://www.billboard.com/charts/hot-100"
response = requests.get(url)
```

In [ ]:
```
# Create a BeautifulSoup object to parse the HTML content:
soup = BeautifulSoup(response.content, "html.parser")
```

In [ ]:
```
# Find the container that holds the song details:
container = soup.find("ol", class_="chart-list__elements")
```

In [ ]:
```
# Iterate over each song in the container and extract the required details:
for song in container.find_all("li"):
```

In [ ]:
```
# Extract song name
  song_name = song.find("span", class_="chart-element__information__song").text
```

In [ ]:
```
# Extract artist name
  artist_name = song.find("span", class_="chart-element__information__artist").text
```

In [ ]:
```
 # Extract last week rank
  last_week_rank = song.find("span", class_="chart-element__meta text--last").text
```

In [ ]:
```
# Extract peak rank
  peak_rank = song.find("span", class_="chart-element__meta text--peak").text
```

In [ ]:
```
# Extract weeks on board
  weeks_on_board = song.find("span", class_="chart-element__meta text--week").text
```

In [ ]:
```
# Print or store the extracted details
  print("Song:", song_name)
  print("Artist:", artist_name)
  print("Last Week Rank:", last_week_rank)
  print("Peak Rank:", peak_rank)
  print("Weeks on Board:", weeks_on_board)
  print()
```

In [ ]:
```
5. Scrape the details of Highest selling novels.
A) Book name
B) Author name
C) Volumes sold
D) Publisher
E) Genre
 Url - https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-ti
```

In [ ]:
```
import requests
from bs4 import BeautifulSoup
```

In [ ]:
```
# Send a GET request to the URL
url = "https://www.theguardian.com/news/datablog/2012/aug/09/best-selling-books-all-ti
response = requests.get(url)
```

In [ ]:
```
# Parse the HTML content
soup = BeautifulSoup(response.content, 'html.parser')
```

In [ ]:
```
# Find the relevant HTML elements and extract the data
novels = []
table = soup.find('table')
rows = table.find_all('tr')[1:]  # Exclude the header row
```

In [ ]:
```
for row in rows:
  columns = row.find_all('td')
  book_name = columns[1].text.strip()
  author_name = columns[2].text.strip()
  volumes_sold = columns[3].text.strip()
  publisher = columns[4].text.strip()
  genre = columns[5].text.strip()

  novel = {
  'Book Name': book_name,
  'Author Name': author_name,
  'Volumes Sold': volumes_sold,
  'Publisher': publisher,
  'Genre': genre
  }
  novels.append(novel)
```

In [ ]:
```
# Print the scraped data
for novel in novels:
  print(novel)
```