Web Scrapping Assignment4

December 31, 2023

[3]:
```
#question 1
# Import necessary libraries
import requests
from bs4 import BeautifulSoup

# Defines the URL of the Wikipedia page containing the list of most-viewed␣
 ↪YouTube videos
url = "https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos"

# Sends a GET request to the URL and store the response
response = requests.get(url)

# Creates a BeautifulSoup object to parse the HTML content of the page
soup = BeautifulSoup(response.text, 'html.parser')

# Initializes an empty list to store details of each video
video_details = []

# Iterates through each row of the table (skipping the header row) to extract␣
 ↪video details
for row in soup.select(".wikitable tbody tr")[1:]:
    # Extract individual columns for each video
    columns = row.find_all('td')
    rank = columns[0].text.strip()
    name = columns[1].text.strip()
    artist = columns[2].text.strip()
    upload_date = columns[3].text.strip()
    views = columns[4].text.strip()

    # Stores the extracted details in a dictionary and append it to the list
    video_details.append({
        'Rank': rank,
        'Name': name,
        'Artist': artist,
        'Upload Date': upload_date,
        'Views': views
    })
```

```
# Prints or store the extracted details
for video in video_details:
    print(video)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16192\1215668182.py in <module>
     10 for row in soup.select(".wikitable tbody tr")[1:]:  # Skip header row
     11     columns = row.find_all('td')
---> 12     rank = columns[0].text.strip()
     13     name = columns[1].text.strip()
     14     artist = columns[2].text.strip()

IndexError: list index out of range
```

[4]:
```
#question 2
# Imports the Selenium webdriver module
from selenium import webdriver

# Defines the URL of the BCCI website
url = "https://www.bcci.tv/"

# Creates a new Chrome webdriver instance
driver = webdriver.Chrome()

# Opens the specified URL in the Chrome browser
driver.get(url)

# Finds and click on the link to navigate to the international fixtures page
fixture_link = driver.find_element_by_xpath("//
 ↪div[@class='navigation__drop-down drop-down drop-down--reveal-on-hover']/
 ↪div[2]/div/ul/li/a")
fixture_link.click()

# Initializes an empty list to store details of each fixture
fixture_details = []

# Iterates through each fixture element on the page to extract relevant␣
 ↪information
for fixture in driver.find_elements_by_xpath("//div[@class='fixture__info']/
 ↪div[@class='fixture__description']"):
    # Extract individual details for each fixture
    series = fixture.find_element_by_xpath(".//
 ↪p[@class='fixture__additional-info']/span[1]").text
```

```
    place = fixture.find_element_by_xpath(".//
 ↪p[@class='fixture__additional-info']/span[3]").text
    date = fixture.find_element_by_xpath(".//div[@class='fixture__full-date']").
 ↪text
    time = fixture.find_element_by_xpath(".//div[@class='fixture__full-date']/
 ↪span").text

    # Store the extracted details in a dictionary and append it to the list
    fixture_details.append({
        'Series': series,
        'Place': place,
        'Date': date,
        'Time': time
    })

# Prints or store the extracted details
for fixture in fixture_details:
    print(fixture)

# Closes the Chrome browser window
driver.quit()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16192\2440940590.py in <module>
      6
      7 # Navigate to the international fixtures page
----> 8 fixture_link = driver.find_element_by_xpath("//
 ↪div[@class='navigation__drop-down drop-down drop-down--reveal-on-hover']/
 ↪div[2]/div/ul/li/a")
      9 fixture_link.click()
     10

AttributeError: 'WebDriver' object has no attribute 'find_element_by_xpath'
```

[5]:
```
#question 3
# Imports necessary libraries
import requests
from bs4 import BeautifulSoup

# Defines the URL of the Statistics Times website
url = "http://statisticstimes.com/"

# Sends a GET request to the URL and create a BeautifulSoup object to parse the␣
 ↪HTML content
response = requests.get(url)
```

```
soup = BeautifulSoup(response.text, 'html.parser')

# Finds the link to the "Economy" page and navigate to it
economy_link = soup.find("a", text="Economy")
economy_url = url + economy_link["href"]
response_economy = requests.get(economy_url)
soup_economy = BeautifulSoup(response_economy.text, 'html.parser')

# Finds the link to the "GDP of Indian states" page and navigate to it
gdp_link = soup_economy.find("a", text="GDP of Indian states")
gdp_url = url + gdp_link["href"]
response_gdp = requests.get(gdp_url)
soup_gdp = BeautifulSoup(response_gdp.text, 'html.parser')

# Initializes an empty list to store details of GDP for each Indian state
state_gdp_details = []

# Iterates through each row of the table (skipping the header row) to extract␣
 ↪state-wise GDP details
for row in soup_gdp.select(".display tbody tr")[1:]:
    # Extracts individual columns for each state
    columns = row.find_all('td')
    rank = columns[0].text.strip()
    state = columns[1].text.strip()
    gdp_18_19_current_prices = columns[2].text.strip()
    gdp_19_20_current_prices = columns[3].text.strip()
    share_18_19 = columns[4].text.strip()
    gdp_billion = columns[5].text.strip()

    # Stores the extracted details in a dictionary and append it to the list
    state_gdp_details.append({
        'Rank': rank,
        'State': state,
        'GSDP (18-19) - Current Prices': gdp_18_19_current_prices,
        'GSDP (19-20) - Current Prices': gdp_19_20_current_prices,
        'Share (18-19)': share_18_19,
        'GDP ($ billion)': gdp_billion
    })

# Prints or store the extracted details
for state_gdp in state_gdp_details:
    print(state_gdp)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16192\470147221.py in <module>
      8 # Find the link to the economy page and navigate to it
```

```
      9 economy_link = soup.find("a", text="Economy")
---> 10 economy_url = url + economy_link["href"]
     11 response_economy = requests.get(economy_url)
     12 soup_economy = BeautifulSoup(response_economy.text, 'html.parser')

TypeError: 'NoneType' object is not subscriptable
```

[6]:
```
#question 4
from selenium import webdriver
from selenium.webdriver.common.by import By

# Opens GitHub and navigate to the trending repositories page
url = "https://github.com/"
driver = webdriver.Chrome()
driver.get(url)

# Clicks on the "Explore" menu and then select "Trending"
explore_menu = driver.find_element(By.XPATH, "//summary[contains(text(),␣
 ↪'Explore')]")
explore_menu.click()

trending_option = driver.find_element(By.XPATH, "//a[contains(text(),␣
 ↪'Trending')]")
trending_option.click()

# Extracts details of the trending repositories
repository_details = []

for repository in driver.find_elements(By.XPATH, "//article[@class='Box-row']"):
    title = repository.find_element(By.TAG_NAME, "h1").text.strip()
    description = repository.find_element(By.TAG_NAME, "p").text.strip()
    contributors_count = repository.find_element(By.XPATH, ".//
 ↪a[contains(@href, '/contributors')]").text.strip()
    language_used = repository.find_element(By.XPATH, ".//
 ↪span[@itemprop='programmingLanguage']").text.strip()

    repository_details.append({
        'Repository Title': title,
        'Repository Description': description,
        'Contributors Count': contributors_count,
        'Language Used': language_used
    })

# Prints or store the extracted details
for repo in repository_details:
    print(repo)
```

```
# Closes the browser window
driver.quit()
```

```
---------------------------------------------------------------------------
NoSuchElementException                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16192\3021161790.py in <module>
      8
      9 # Click on the "Explore" menu and then select "Trending"
---> 10 explore_menu = driver.find_element(By.XPATH, "//summary[contains(text()␣
 ↪'Explore')]")
     11 explore_menu.click()
     12

~\anaconda3\lib\site-packages\selenium\webdriver\remote\webdriver.py in␣
 ↪find_element(self, by, value)
    739             value = f'[name="{value}"]'
    740
--> 741         return self.execute(Command.FIND_ELEMENT, {"using": by, "value"␣
 ↪value})["value"]
    742
    743     def find_elements(self, by=By.ID, value: Optional[str] = None) ->␣
 ↪List[WebElement]:

~\anaconda3\lib\site-packages\selenium\webdriver\remote\webdriver.py in␣
 ↪execute(self, driver_command, params)
    345         response = self.command_executor.execute(driver_command, params
    346         if response:
--> 347             self.error_handler.check_response(response)
    348             response["value"] = self._unwrap_value(response.get("value"␣
 ↪None))
    349             return response

~\anaconda3\lib\site-packages\selenium\webdriver\remote\errorhandler.py in␣
 ↪check_response(self, response)
    227                 alert_text = value["alert"].get("text")
    228             raise exception_class(message, screen, stacktrace,␣
 ↪alert_text)  # type: ignore[call-arg]  # mypy is not smart enough here
--> 229         raise exception_class(message, screen, stacktrace)

NoSuchElementException: Message: no such element: Unable to locate element:␣
 ↪{"method":"xpath","selector":"//summary[contains(text(), 'Explore')]"}
  (Session info: chrome=120.0.6099.71); For documentation on this error, please␣
 ↪visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/
 ↪errors#no-such-element-exception
Stacktrace:
        GetHandleVerifier [0x00007FF6FC012142+3514994]
```

```
        (No symbol) [0x00007FF6FBC30CE2]
        (No symbol) [0x00007FF6FBAD76AA]
        (No symbol) [0x00007FF6FBB21860]
        (No symbol) [0x00007FF6FBB2197C]
        (No symbol) [0x00007FF6FBB64EE7]
        (No symbol) [0x00007FF6FBB4602F]
        (No symbol) [0x00007FF6FBB628F6]
        (No symbol) [0x00007FF6FBB45D93]
        (No symbol) [0x00007FF6FBB14BDC]
        (No symbol) [0x00007FF6FBB15C64]
        GetHandleVerifier [0x00007FF6FC03E16B+3695259]
        GetHandleVerifier [0x00007FF6FC096737+4057191]
        GetHandleVerifier [0x00007FF6FC08E4E3+4023827]
        GetHandleVerifier [0x00007FF6FBD604F9+689705]
        (No symbol) [0x00007FF6FBC3C048]
        (No symbol) [0x00007FF6FBC38044]
        (No symbol) [0x00007FF6FBC381C9]
        (No symbol) [0x00007FF6FBC288C4]
        BaseThreadInitThunk [0x00007FFE1F8A7344+20]
        RtlUserThreadStart [0x00007FFE20BE26B1+33]
```

[7]:
```python
#question 5
from selenium import webdriver
from selenium.webdriver.common.by import By

# Opens Billboard and navigate to the Hot 100 page
url = "https://www.billboard.com/"
driver = webdriver.Chrome()
driver.get(url)

# Clicks on the "Charts" option and then select "Hot 100"
charts_menu = driver.find_element(By.XPATH, "//a[contains(@href, '/charts')]")
charts_menu.click()

hot_100_link = driver.find_element(By.XPATH, "//a[contains(@href, '/hot-100')]")
hot_100_link.click()

# Extracts details of the top 100 songs
songs_details = []

for song in driver.find_elements(By.XPATH, "//
 ↪li[@class='chart-list__element']"):
    name = song.find_element(By.CLASS_NAME, "chart-element__information__song").
 ↪text.strip()
```

```
    artist = song.find_element(By.CLASS_NAME,␣
 ↪"chart-element__information__artist").text.strip()
    last_week_rank = song.find_element(By.CLASS_NAME, "chart-element__meta.
 ↪text--center.color--secondary.text--last").text.strip()
    peak_rank = song.find_element(By.CLASS_NAME, "chart-element__meta.
 ↪text--center.color--secondary.text--peak").text.strip()
    weeks_on_board = song.find_element(By.CLASS_NAME, "chart-element__meta.
 ↪text--center.color--secondary.text--week").text.strip()

    songs_details.append({
        'Song Name': name,
        'Artist Name': artist,
        'Last Week Rank': last_week_rank,
        'Peak Rank': peak_rank,
        'Weeks on Board': weeks_on_board
    })

# Prints or store the extracted details
for song in songs_details:
    print(song)

# Closes the browser window
driver.quit()
```

```
---------------------------------------------------------------------------
ElementNotInteractableException                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16192\3948064181.py in <module>
      9 # Click on the "Charts" option and then select "Hot 100"
     10 charts_menu = driver.find_element(By.XPATH, "//a[contains(@href, '/
 ↪charts')]")
---> 11 charts_menu.click()
     12
     13 hot_100_link = driver.find_element(By.XPATH, "//a[contains(@href, '/
 ↪hot-100')]")

~\anaconda3\lib\site-packages\selenium\webdriver\remote\webelement.py in␣
 ↪click(self)
     91     def click(self) -> None:
     92         """Clicks the element."""
---> 93         self._execute(Command.CLICK_ELEMENT)
     94
     95     def submit(self):

~\anaconda3\lib\site-packages\selenium\webdriver\remote\webelement.py in␣
 ↪_execute(self, command, params)
    392             params = {}
    393         params["id"] = self._id
```

```
--> 394         return self._parent.execute(command, params)
    395
    396     def find_element(self, by=By.ID, value=None) -> WebElement:

~\anaconda3\lib\site-packages\selenium\webdriver\remote\webdriver.py in␣
 ↪execute(self, driver_command, params)
    345         response = self.command_executor.execute(driver_command, params
    346         if response:
--> 347             self.error_handler.check_response(response)
    348             response["value"] = self._unwrap_value(response.get("value"␣
 ↪None))
    349             return response

~\anaconda3\lib\site-packages\selenium\webdriver\remote\errorhandler.py in␣
 ↪check_response(self, response)
    227                 alert_text = value["alert"].get("text")
    228             raise exception_class(message, screen, stacktrace,␣
 ↪alert_text)  # type: ignore[call-arg]  # mypy is not smart enough here
--> 229         raise exception_class(message, screen, stacktrace)

ElementNotInteractableException: Message: element not interactable
  (Session info: chrome=120.0.6099.71)
Stacktrace:
        GetHandleVerifier [0x00007FF6FC012142+3514994]
        (No symbol) [0x00007FF6FBC30CE2]
        (No symbol) [0x00007FF6FBAD74C3]
        (No symbol) [0x00007FF6FBB22D29]
        (No symbol) [0x00007FF6FBB16A0F]
        (No symbol) [0x00007FF6FBB45FEA]
        (No symbol) [0x00007FF6FBB163B6]
        (No symbol) [0x00007FF6FBB46490]
        (No symbol) [0x00007FF6FBB628F6]
        (No symbol) [0x00007FF6FBB45D93]
        (No symbol) [0x00007FF6FBB14BDC]
        (No symbol) [0x00007FF6FBB15C64]
        GetHandleVerifier [0x00007FF6FC03E16B+3695259]
        GetHandleVerifier [0x00007FF6FC096737+4057191]
        GetHandleVerifier [0x00007FF6FC08E4E3+4023827]
        GetHandleVerifier [0x00007FF6FBD604F9+689705]
        (No symbol) [0x00007FF6FBC3C048]
        (No symbol) [0x00007FF6FBC38044]
        (No symbol) [0x00007FF6FBC381C9]
        (No symbol) [0x00007FF6FBC288C4]
        BaseThreadInitThunk [0x00007FFE1F8A7344+20]
        RtlUserThreadStart [0x00007FFE20BE26B1+33]
```

[ ]:
```python
#question 6
# Imports necessary libraries
import requests
from bs4 import BeautifulSoup

# Defines the URL of The Guardian's data blog page about best-selling books
url = "https://www.theguardian.com/news/datablog/2012/aug/09/
 ↪best-selling-books-all-time-fifty-shades-grey-compare"

# Sends a GET request to the URL and create a BeautifulSoup object to parse the␣
 ↪HTML content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Initializes an empty list to store details of best-selling novels
novel_details = []

# Iterates through each row of the table (skipping the header row) to extract␣
 ↪novel details
for row in soup.select(".interactive-body tbody tr")[1:]:
    # Extracts individual columns for each novel
    columns = row.find_all('td')
    book_name = columns[0].text.strip()
    author_name = columns[1].text.strip()
    volumes_sold = columns[2].text.strip()
    publisher = columns[3].text.strip()
    genre = columns[4].text.strip()

    # Stores the extracted details in a dictionary and append it to the list
    novel_details.append({
        'Book Name': book_name,
        'Author Name': author_name,
        'Volumes Sold': volumes_sold,
        'Publisher': publisher,
        'Genre': genre
    })

# Prints or store the extracted details
for novel in novel_details:
    print(novel)
```

[ ]:
```python
#question 7
# Imports necessary libraries
from bs4 import BeautifulSoup
import requests

# Defines the URL of the IMDb list of TV series
```

```
url = "https://www.imdb.com/list/ls095964455/"

# Sends a GET request to the URL and create a BeautifulSoup object to parse the␣
 ↪HTML content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Initializes an empty list to store details of TV series
series_details = []

# Iterates through each series element on the page to extract relevant␣
 ↪information
for series in soup.select(".lister-item-content"):
    # Extracts individual details for each TV series
    name = series.find("a").text
    year_span = series.find("span", class_="lister-item-year").text
    genre = series.find("span", class_="genre").text.strip()
    run_time = series.find("span", class_="runtime").text.strip()
    ratings = series.find("strong").text
    votes = series.find("span", attrs={"name": "nv"})["data-value"]

    # Stores the extracted details in a dictionary and append it to the list
    series_details.append({
        'Name': name,
        'Year Span': year_span,
        'Genre': genre,
        'Run Time': run_time,
        'Ratings': ratings,
        'Votes': votes
    })

# Prints or store the extracted details
for series in series_details:
    print(series)
```

[ ]:
```
#question 8
# Import necessary libraries
import requests
from bs4 import BeautifulSoup

# Defines the URL of the UCI Machine Learning Repository
url = "https://archive.ics.uci.edu/"

# Sends a GET request to the URL and create a BeautifulSoup object to parse the␣
 ↪HTML content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
```

```
# Finds the link to the "Show All Dataset" page and navigate to it
all_datasets_link = soup.find("a", text="View ALL Data Sets")
all_datasets_url = url + all_datasets_link["href"]
response_all_datasets = requests.get(all_datasets_url)
soup_all_datasets = BeautifulSoup(response_all_datasets.text, 'html.parser')

# Initializes an empty list to store details of datasets
dataset_details = []

# Iterates through each row of the table (skipping the header row) to extract␣
 ↪dataset details
for row in soup_all_datasets.select("table tr")[1:]:
    # Extract individual columns for each dataset
    columns = row.find_all('td')
    dataset_name = columns[0].text.strip()
    data_type = columns[1].text.strip()
    task = columns[2].text.strip()
    attribute_type = columns[3].text.strip()
    no_of_instances = columns[4].text.strip()
    no_of_attributes = columns[5].text.strip()
    year = columns[6].text.strip()

    # Stores the extracted details in a dictionary and append it to the list
    dataset_details.append({
        'Dataset Name': dataset_name,
        'Data Type': data_type,
        'Task': task,
        'Attribute Type': attribute_type,
        'No of Instances': no_of_instances,
        'No of Attributes': no_of_attributes,
        'Year': year
    })

# Prints or store the extracted details
for dataset in dataset_details:
    print(dataset)
```