Web Scrapping Assignment 2

December 10, 2023

[1]:
```
#Question 1
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
import pandas as pd  # For handling data in a tabular format

# Step 1: Get the webpage
url = "https://www.shine.com/"
response = requests.get(url)  # Send a GET request to the specified URL
soup = BeautifulSoup(response.text, 'html.parser')  # Parse the HTML content of␣
 ↪the page

# Step 2-3: Enter job title and location, then click the search button
job_title = "Data Analyst"
location = "Bangalore"

# Create a dictionary payload with job title and location
payload = {
    'q': job_title,
    'l': location
}

# Construct the search URL with the job title and location
search_url = "https://www.shine.com/job-search/{job_title}-jobs-in-{location}".
 ↪format(job_title=job_title, location=location)

# Send a GET request to the search URL with the payload
search_response = requests.get(search_url, params=payload)
search_soup = BeautifulSoup(search_response.text, 'html.parser')  # Parse the␣
 ↪HTML content of the search results page

# Step 4: Scrape data for the first 10 jobs
jobs_data = []  # Initialize an empty list to store job data
job_results = search_soup.find_all('div', class_='search_listing')  # Find all␣
 ↪job listings on the page

# Loop through the first 10 job listings and extract relevant information
```

```
for job_result in job_results[:10]:
    job_title = job_result.find('li', class_='srl_head').text.strip()  #␣
 ↪Extract job title
    job_location = job_result.find('li', class_='srl_exp').text.strip()  #␣
 ↪Extract job location
    company_name = job_result.find('li', class_='srl_cmp').text.strip()  #␣
 ↪Extract company name
    experience_required = job_result.find('li', class_='srl_role').text.strip()␣
 ↪ # Extract experience required

    # Create a dictionary with the extracted information for each job
    job_data = {
        'Job Title': job_title,
        'Job Location': job_location,
        'Company Name': company_name,
        'Experience Required': experience_required
    }

    # Append the job data dictionary to the list of jobs_data
    jobs_data.append(job_data)

# Step 5: Create a dataframe using the collected job data
df = pd.DataFrame(jobs_data)

# Display the dataframe
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

[14]:
```
#Question 2
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
import pandas as pd  # For handling data in a tabular format

# Define a function to scrape job data from Shine website
def scrape_shine_data(job_title, location, num_jobs=10):
    # Step 1: Get the webpage
    url = "https://www.shine.com/"
    response = requests.get(url)  # Send a GET request to the specified URL
    soup = BeautifulSoup(response.text, 'html.parser')  # Parse the HTML␣
 ↪content of the page

    # Step 2-3: Enter job title and location, then click the search button
    payload = {'q': job_title, 'l': location}
```

```
    # Construct the search URL with the formatted job title and location
    search_url = f"https://www.shine.com/job-search/{job_title.lower().
 ↪replace(' ', '-')}-jobs-in-{location.lower().replace(' ', '-')}"

    # Send a GET request to the search URL with the payload
    search_response = requests.get(search_url, params=payload)
    search_soup = BeautifulSoup(search_response.text, 'html.parser')  # Parse␣
 ↪the HTML content of the search results page

    # Step 4: Scrape data for the first 'num_jobs' jobs
    jobs_data = []  # Initialize an empty list to store job data
    job_results = search_soup.find_all('div', class_='search_listing')  # Find␣
 ↪all job listings on the page

    # Loop through the specified number of job listings and extract relevant␣
 ↪information
    for job_result in job_results[:num_jobs]:
        job_title = job_result.find('li', class_='srl_head').text.strip()  #␣
 ↪Extract job title
        job_location = job_result.find('li', class_='srl_exp').text.strip()  #␣
 ↪Extract job location
        company_name = job_result.find('li', class_='srl_cmp').text.strip()  #␣
 ↪Extract company name

        # Create a dictionary with the extracted information for each job
        job_data = {
            'Job Title': job_title,
            'Job Location': job_location,
            'Company Name': company_name
        }

        # Append the job data dictionary to the list of jobs_data
        jobs_data.append(job_data)

    # Step 5: Create a dataframe using the collected job data
    df = pd.DataFrame(jobs_data)

    return df

# Example usage for scraping Data Scientist jobs in Bangalore
data_scientist_df = scrape_shine_data("Data Scientist", "Bangalore",␣
 ↪num_jobs=10)

# Display the dataframe
print(data_scientist_df)
```

```
Empty DataFrame
Columns: []
Index: []
```

[15]:
```python
#Question 3
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
import pandas as pd  # For handling data in a tabular format

# Define a function to scrape job data from Shine website with additional␣
 ↪filters
def scrape_shine_data_with_filters(job_title, location, salary_range,␣
 ↪num_jobs=10):
    # Step 1: Get the webpage
    url = "https://www.shine.com/"
    response = requests.get(url)  # Send a GET request to the specified URL
    soup = BeautifulSoup(response.text, 'html.parser')  # Parse the HTML␣
 ↪content of the page

    # Step 2-3: Enter job title and click the search button
    payload = {'q': job_title}

    # Construct the search URL with the formatted job title
    search_url = f"https://www.shine.com/job-search/{job_title.lower().
 ↪replace(' ', '-')}-jobs"

    # Send a GET request to the search URL with the payload
    search_response = requests.get(search_url, params=payload)
    search_soup = BeautifulSoup(search_response.text, 'html.parser')  # Parse␣
 ↪the HTML content of the search results page

    # Step 4: Apply location and salary filters
    location_filter_url = f"https://www.shine.com/job-search/{job_title.lower().
 ↪replace(' ', '-')}-jobs-in-{location.lower().replace('/', '-')}"
    location_response = requests.get(location_filter_url)
    location_soup = BeautifulSoup(location_response.text, 'html.parser')  #␣
 ↪Parse the HTML content after applying location filter

    salary_filter_url = f"https://www.shine.com/job-search/{job_title.lower().
 ↪replace(' ', '-')}-jobs-{salary_range.lower().replace('-', '')}"
    salary_response = requests.get(salary_filter_url)
    salary_soup = BeautifulSoup(salary_response.text, 'html.parser')  # Parse␣
 ↪the HTML content after applying salary filter

    # Step 5: Scrape data for the first 'num_jobs' jobs
    jobs_data = []  # Initialize an empty list to store job data
```

```
    job_results = search_soup.find_all('div', class_='search_listing')  # Find␣
 ↪all job listings on the page

    # Loop through the specified number of job listings and extract relevant␣
 ↪information
    for job_result in job_results[:num_jobs]:
        job_title = job_result.find('li', class_='srl_head').text.strip()  #␣
 ↪Extract job title
        job_location = job_result.find('li', class_='srl_exp').text.strip()  #␣
 ↪Extract job location
        company_name = job_result.find('li', class_='srl_cmp').text.strip()  #␣
 ↪Extract company name
        experience_required = job_result.find('li', class_='srl_role').text.
 ↪strip()  # Extract experience required

        # Create a dictionary with the extracted information for each job
        job_data = {
            'Job Title': job_title,
            'Job Location': job_location,
            'Company Name': company_name,
            'Experience Required': experience_required
        }

        # Append the job data dictionary to the list of jobs_data
        jobs_data.append(job_data)

    # Step 6: Create a dataframe using the collected job data
    df = pd.DataFrame(jobs_data)

    return df

# Example usage for scraping Data Scientist jobs in Delhi/NCR with a salary␣
 ↪range of 3-6 lakhs
data_scientist_delhi_df = scrape_shine_data_with_filters("Data Scientist",␣
 ↪"Delhi/NCR", "3-6", num_jobs=10)

# Display the dataframe
print(data_scientist_delhi_df)
```

```
Empty DataFrame
Columns: []
Index: []
```

[16]:
```
#Question 4
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
```

```
import pandas as pd  # For handling data in a tabular format

# Define a function to scrape sunglasses data from Flipkart
def scrape_flipkart_sunglasses(url):
    # Step 1: Send a GET request to the URL
    response = requests.get(url)

    # Step 2: Parse the HTML content
    soup = BeautifulSoup(response.text, 'html.parser')

    # Step 3: Find the sunglasses listings
    sunglasses_listings = soup.find_all('div', class_='_1AtVbE')

    # Step 4: Scrape data for the first 100 sunglasses
    data = []  # Initialize an empty list to store sunglasses data
    for sunglasses in sunglasses_listings[:100]:
        brand = sunglasses.find('div', class_='_2WkVRV').text.strip()  #␣
 ↪Extract brand
        product_description = sunglasses.find('a', class_='IRpwTa').text.
 ↪strip()  # Extract product description
        price = sunglasses.find('div', class_='_30jeq3').text.strip()  #␣
 ↪Extract price

        # Create a dictionary with the extracted information for each pair of␣
 ↪sunglasses
        sunglasses_data = {
            'Brand': brand,
            'Product Description': product_description,
            'Price': price
        }
        # Append the sunglasses data dictionary to the list of data
        data.append(sunglasses_data)

    return data

# URL for Flipkart sunglasses listings
flipkart_url = "https://www.flipkart.com/search?
 ↪q=sunglasses&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off"

# Scrape data using the defined function
sunglasses_data = scrape_flipkart_sunglasses(flipkart_url)

# Create a dataframe using the collected sunglasses data
df_flipkart = pd.DataFrame(sunglasses_data)

# Display the dataframe
print(df_flipkart)
```

```
Empty DataFrame
Columns: []
Index: []
```

[17]:
```python
#Question 5
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
import pandas as pd  # For handling data in a tabular format

# Define a function to scrape iPhone 11 reviews data from Flipkart
def scrape_flipkart_reviews(url):
    # Step 1: Send a GET request to the URL
    response = requests.get(url)

    # Step 2: Parse the HTML content
    soup = BeautifulSoup(response.text, 'html.parser')

    # Step 3: Find the review listings
    review_listings = soup.find_all('div', class_='_27M-vq')

    # Step 4: Scrape data for the first 100 reviews
    data = []  # Initialize an empty list to store reviews data
    for review in review_listings[:100]:
        rating = review.find('div', class_='E_uFuv').text.strip()  # Extract␣
 ↪rating
        review_summary = review.find('p', class_='_2-N8zT').text.strip()  #␣
 ↪Extract review summary
        full_review = review.find('div', class_='t-ZTKy').text.strip()  #␣
 ↪Extract full review

        # Create a dictionary with the extracted information for each review
        review_data = {
            'Rating': rating,
            'Review Summary': review_summary,
            'Full Review': full_review
        }
        # Append the review data dictionary to the list of data
        data.append(review_data)

    return data

# URL for iPhone 11 reviews on Flipkart
iphone11_reviews_url = "https://www.flipkart.com/apple-iphone-11-black-64-gb/
 ↪product-reviews/itm4e5041ba101fd?
 ↪pid=MOBFWQ6BXGJCEYNY&lid=LSTMOBFWQ6BXGJCEYNYZXSHRJ&marketplace=FLIPKART"
```

```
# Scrape data using the defined function
reviews_data = scrape_flipkart_reviews(iphone11_reviews_url)

# Create a dataframe using the collected reviews data
df_reviews = pd.DataFrame(reviews_data)

# Display the dataframe
print(df_reviews)
```

```
Empty DataFrame
Columns: []
Index: []
```

[18]:
```
#Question 6
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping
import pandas as pd  # For handling data in a tabular format

# Define a function to scrape sneakers data from Flipkart
def scrape_flipkart_sneakers(url):
    # Step 1: Send a GET request to the URL
    response = requests.get(url)

    # Step 2: Parse the HTML content
    soup = BeautifulSoup(response.text, 'html.parser')

    # Step 3: Find the sneaker listings
    sneaker_listings = soup.find_all('div', class_='_1AtVbE')

    # Step 4: Scrape data for the first 100 sneakers
    data = []  # Initialize an empty list to store sneakers data
    for sneaker in sneaker_listings[:100]:
        brand = sneaker.find('div', class_='_2WkVRV').text.strip()  # Extract␣
↪brand
        product_description = sneaker.find('a', class_='IRpwTa').text.strip() ␣
↪# Extract product description
        price = sneaker.find('div', class_='_30jeq3').text.strip()  # Extract␣
↪price

        # Create a dictionary with the extracted information for each pair of␣
↪sneakers
        sneaker_data = {
            'Brand': brand,
            'Product Description': product_description,
            'Price': price
        }
```

```
        # Append the sneaker data dictionary to the list of data
        data.append(sneaker_data)

    return data

# URL for sneakers listings on Flipkart
sneakers_url = "https://www.flipkart.com/search?
 ↪q=sneakers&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off"

# Scrape data using the defined function
sneakers_data = scrape_flipkart_sneakers(sneakers_url)

# Create a dataframe using the collected sneakers data
df_sneakers = pd.DataFrame(sneakers_data)

# Display the dataframe
print(df_sneakers)
```

```
Empty DataFrame
Columns: []
Index: []
```

[19]:
```
#Question 7
# Import necessary libraries
import requests  # For making HTTP requests
from bs4 import BeautifulSoup  # For web scraping

# Define the URL for Amazon laptops listing
url = "https://www.amazon.in/s?
 ↪k=laptop&rh=n%3A1375424031%2Cp_n_feature_thirteen_browse-bin%3A12598163031&dc&ds=v1%3Aq0%2F

# Send a GET request to the URL
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Find the laptops listings
laptops = soup.find_all('div', {'data-asin': True})[:10]  # Extract the first␣
 ↪10 laptops

# Loop through the laptops and extract information
for laptop in laptops:
    title = laptop.find('span', {'class': 'a-text-normal'}).text.strip()  #␣
 ↪Extract laptop title

    # Ratings may not be available for all laptops, handle it accordingly
    ratings_tag = laptop.find('span', {'class': 'a-icon-alt'})
```

```
    ratings = ratings_tag.text.strip() if ratings_tag else 'Not available'  #␣
 ↪Extract ratings or set as 'Not available'

    price_tag = laptop.find('span', {'class': 'a-offscreen'})
    price = price_tag.text.strip() if price_tag else 'Not available'  # Extract␣
 ↪price or set as 'Not available'

    # Print the information for each laptop
    print(f"Title: {title}\nRatings: {ratings}\nPrice: {price}\n---")
```

[20]:
```
#Question 8
# Import necessary libraries
from selenium import webdriver  # For browser automation
from selenium.webdriver.common.by import By  # For locating elements
from bs4 import BeautifulSoup  # For web scraping
import time  # For introducing delays

# Step 1: Open the webpage
url = "https://www.azquotes.com/top_quotes.html"
driver = webdriver.Chrome()  # You need to have ChromeDriver installed
driver.get(url)

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 2: Click on "Top Quotes"
top_quotes_button = driver.find_element(By.XPATH, '//a[@href="/top-quotes/"]')
top_quotes_button.click()

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 3: Scrape data - Quote, Author, Type of Quotes
quotes_data = []

for page in range(1, 11):  # Assuming there are 100 quotes per page and you␣
 ↪want the top 1000
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    quotes = soup.find_all('div', {'class': 'wrap-block'})

    for quote in quotes:
        quote_text = quote.find('a', {'class': 'title'}).text.strip()  #␣
 ↪Extract quote text
        author = quote.find('a', {'class': 'author'}).text.strip()  # Extract␣
 ↪author
        quote_type = quote.find('div', {'class': 'qti'}).text.strip()  #␣
 ↪Extract type of quote
```

```
        # Append the extracted information to the quotes_data list
        quotes_data.append({
            'Quote': quote_text,
            'Author': author,
            'Type of Quote': quote_type
        })

    # Navigate to the next page (if available)
    next_page_button = driver.find_element(By.XPATH, '//
 ↪li[@class="pagination-next"]/a')
    if 'disabled' in next_page_button.get_attribute("class"):
        break  # No more pages to navigate
    else:
        next_page_button.click()
        time.sleep(2)  # Wait for some time to ensure the next page is fully␣
 ↪loaded

# Close the browser window
driver.quit()

# Display the scraped data
for i, quote_data in enumerate(quotes_data, start=1):
    print(f"{i}. Quote: {quote_data['Quote']}")
    print(f"   Author: {quote_data['Author']}")
    print(f"   Type of Quote: {quote_data['Type of Quote']}")
    print("---")
```

```
---------------------------------------------------------------------------
NoSuchElementException                    Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9896\1803318473.py in <module>
     13
     14 # Step 2: Click on "Top Quotes"
---> 15 top_quotes_button = driver.find_element(By.XPATH, '//a[@href="/
 ↪top-quotes/"]')
     16 top_quotes_button.click()
     17

~\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":"//a[@href="/top-quotes/"]"}
  (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 [0x00007FF6F93E4D02+56194]
        (No symbol) [0x00007FF6F93504B2]
        (No symbol) [0x00007FF6F91F76AA]
        (No symbol) [0x00007FF6F92416D0]
        (No symbol) [0x00007FF6F92417EC]
        (No symbol) [0x00007FF6F9284D77]
        (No symbol) [0x00007FF6F9265EBF]
        (No symbol) [0x00007FF6F9282786]
        (No symbol) [0x00007FF6F9265C23]
        (No symbol) [0x00007FF6F9234A45]
        (No symbol) [0x00007FF6F9235AD4]
        GetHandleVerifier [0x00007FF6F975D5BB+3695675]
        GetHandleVerifier [0x00007FF6F97B6197+4059159]
        GetHandleVerifier [0x00007FF6F97ADF63+4025827]
        GetHandleVerifier [0x00007FF6F947F029+687785]
        (No symbol) [0x00007FF6F935B508]
        (No symbol) [0x00007FF6F9357564]
        (No symbol) [0x00007FF6F93576E9]
        (No symbol) [0x00007FF6F9348094]
        BaseThreadInitThunk [0x00007FFB014D7344+20]
        RtlUserThreadStart [0x00007FFB020E26B1+33]
```

[ ]:
```
#Question 9
# Import necessary libraries
import pandas as pd  # For handling data in a tabular format
from selenium import webdriver  # For browser automation
from selenium.webdriver.common.by import By  # For locating elements
from bs4 import BeautifulSoup  # For web scraping
import time  # For introducing delays

# Step 1: Open the webpage
url = "https://www.jagranjosh.com/"
driver = webdriver.Chrome()  # You need to have ChromeDriver installed
driver.get(url)

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 2: Click on "GK" option
gk_option = driver.find_element(By.XPATH, '//a[contains(text(), "GK")]')
gk_option.click()

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 3: Click on "List of all Prime Ministers of India"
pm_list_option = driver.find_element(By.XPATH, '//a[contains(text(), "List of␣
 ↪all Prime Ministers of India")]')
pm_list_option.click()

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 4: Scrap the data - Name, Born-Dead, Term of office, Remarks
pm_data = []

soup = BeautifulSoup(driver.page_source, 'html.parser')
table = soup.find('table', {'class': 'table4'})

rows = table.find_all('tr')[1:]  # Skipping the header row

for row in rows:
    columns = row.find_all('td')
    name = columns[0].text.strip()  # Extract Prime Minister's name
    born_dead = columns[1].text.strip()  # Extract Born-Dead information
    term_of_office = columns[2].text.strip()  # Extract Term of Office␣
 ↪information
    remarks = columns[3].text.strip()  # Extract Remarks
```

```
    # Append the extracted information to the pm_data list
    pm_data.append({
        'Name': name,
        'Born-Dead': born_dead,
        'Term of Office': term_of_office,
        'Remarks': remarks
    })

# Create a DataFrame using the collected data
df = pd.DataFrame(pm_data)

# Display the DataFrame
print(df)

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

[ ]:
```
#Question 10
# Import necessary libraries
import pandas as pd  # For handling data in a tabular format
from selenium import webdriver  # For browser automation
from selenium.webdriver.common.by import By  # For locating elements
from selenium.webdriver.common.keys import Keys  # For simulating keyboard keys
from bs4 import BeautifulSoup  # For web scraping
import time  # For introducing delays

# Step 1: Open the webpage
url = "https://www.motor1.com/"
driver = webdriver.Chrome()  # You need to have ChromeDriver installed
driver.get(url)

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 2: Type in the search bar '50 most expensive cars'
search_bar = driver.find_element(By.XPATH, '//input[@name="q"]')
search_bar.send_keys("50 most expensive cars")
search_bar.send_keys(Keys.RETURN)

# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 3: Click on '50 most expensive cars in the world..'
expensive_cars_link = driver.find_element(By.XPATH, '//a[contains(@href,␣
 ↪"most-expensive-cars")]')
expensive_cars_link.click()
```

```
# Wait for some time to ensure the page is fully loaded
time.sleep(5)

# Step 4: Scrap the data - Car name and Price
car_data = []

soup = BeautifulSoup(driver.page_source, 'html.parser')
cars = soup.find_all('div', {'class': 'article-entry'})

for car in cars:
    car_name = car.find('h2').text.strip()  # Extract car name
    car_price = car.find('span', {'class': 'price'}).text.strip()  # Extract␣
 ↪car price

    # Append the extracted information to the car_data list
    car_data.append({
        'Car Name': car_name,
        'Price': car_price
    })

# Create a DataFrame using the collected data
df = pd.DataFrame(car_data)

# Display the DataFrame
print(df)

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