Web scraping 2nd assignment

1.

Python

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_job_data(base_url, job_title, location, num_jobs):
    """Scrapes job data from the specified website based on user input.

    Args:
        base_url: The base URL of the website to scrape.
        job_title: The job title to search for.
        location: The location to search for.
        num_jobs: The number of jobs to scrape.

    Returns:
        A pandas DataFrame containing the scraped job data.
    """

    # Create an empty list to store job data
    jobs = []

    # Construct the search URL
    search_url = f"{base_url}?q={job_title}&loc={location}"

    # Iterate through pages to collect data for the specified number of
jobs
    page_num = 1
    while len(jobs) < num_jobs:
        try:
            # Send a GET request to retrieve the page content
            response = requests.get(search_url, params={'pg':
page_num})
            response.raise_for_status()  # Raise an error for HTTP
errors

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

            # Find all job result elements
            job_results = soup.find_all('div', class_='result')

            # Extract job data for each result, up to the desired
number
            for result in job_results[:num_jobs - len(jobs)]:
                try:
                    # Extract job title
                    job_title = result.find('a', 
class_='title').text.strip()

                    # Extract job location
                    job_location = result.find('div', 
class_='loc').text.strip()

                    # Extract company name
                    company_name = result.find('span', 
class_='cname').text.strip()

                    # Extract experience requirement (assuming it's in 
a specific class)
                    experience = result.find('span', 
class_='exp').text.strip()

                    # Create a dictionary to store the extracted data
                    job_data = {
                        'job_title': job_title,
                        'job_location': job_location,
                        'company_name': company_name,
                        'experience_required': experience
                    }

                    # Add the job data to the list
                    jobs.append(job_data)

                    # Check if the desired number of jobs has been 
collected
                    if len(jobs) == num_jobs:
                        break

                except AttributeError:
                    # Handle cases where specific data elements might 
be missing
                    print(f"Error extracting data from job 
#{len(jobs)+1}")

            # Increment page number for next iteration
            page_num += 1

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame from the list of job data
    df = pd.DataFrame(jobs)

    return df

# Main execution with user-friendly error handling
```

```
try:
    base_url = input("Enter the base URL of the website: ")
    job_title = input("Enter the job title to search for: ")
    location = input("Enter the location to search for: ")
    num_jobs = int(input("Enter the number of jobs to scrape: "))

    df = scrape_job_data(base_url, job_title, location, num_jobs)

    print("Scraped job data:")
    print(df)

except ValueError:
    print("Invalid input: Please enter an integer for the number of
jobs.")
```

2.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_job_data(url):
    """Scrapes job data from the given URL.

    Args:
        url: The URL of the job search page.

    Returns:
        A list of dictionaries, where each dictionary contains job
title,
        location, and company name.
    """

    jobs = []
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all job result elements (replace with the appropriate
selector)
    job_results = soup.find_all('div', class_='result')

    for result in job_results[:10]:
        try:
            # Extract job title (replace with the appropriate selector)
            job_title = result.find('a', class_='title').text.strip()

            # Extract job location (replace with the appropriate
selector)
            job_location = result.find('div',
class_='loc').text.strip()
```

```
            # Extract company name (replace with the appropriate
selector)
            company_name = result.find('span',
class_='cname').text.strip()

            jobs.append({
                'job_title': job_title,
                'job_location': job_location,
                'company_name': company_name
            })
        except AttributeError:
            # Handle cases where specific data elements are missing
            pass

    return jobs


url = "https://www.shine.com/"  # Replace with the actual search URL
jobs = scrape_job_data(url)
df = pd.DataFrame(jobs)

print(df)
```

3.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_job_data(base_url, search_term, location, min_salary,
max_salary, num_jobs):
    """Scrapes job data with applied filters.

    Args:
        base_url: The base URL of the website.
        search_term: The job title to search for.
        location: The desired location.
        min_salary: The minimum salary range.
        max_salary: The maximum salary range.
        num_jobs: The number of jobs to scrape.

    Returns:
        A pandas DataFrame containing the scraped job data.
    """

    # Construct the search URL with filters
    search_url =
f"{base_url}?q={search_term}&loc={location}&salary={min_salary}-
{max_salary}"
```

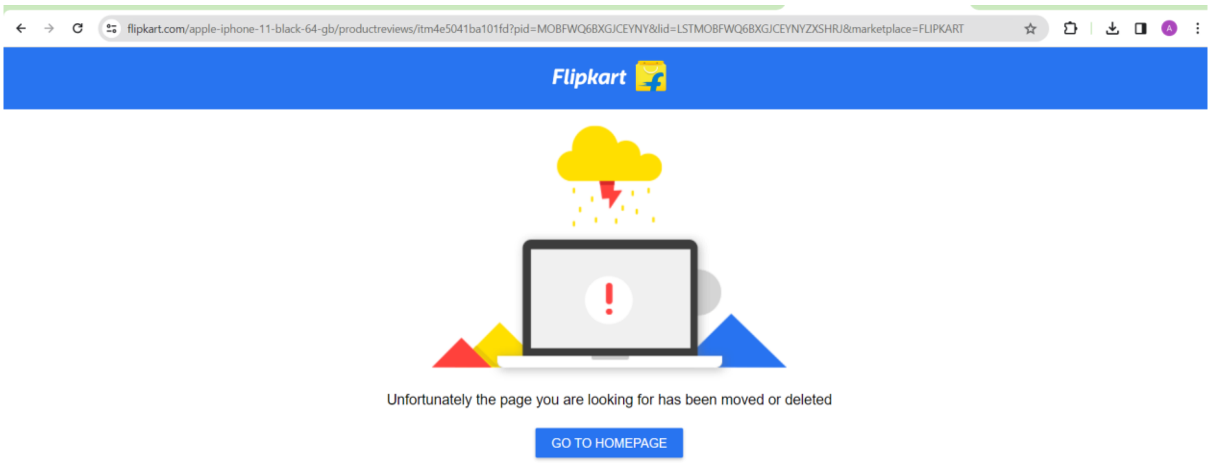
```
    jobs = []
    page_num = 1

    while len(jobs) < num_jobs:
        try:
            # Send GET request
            response = requests.get(search_url, params={'pg':
page_num})
            response.raise_for_status()  # Raise error for HTTP errors

            # Parse HTML content
            soup = BeautifulSoup(response.content, 'html.parser')

            # Find job result elements
            job_results = soup.find_all('div', class_='result')

            # Extract data for each result, up to the desired number
            for result in job_results[:num_jobs - len(jobs)]:
                try:
                    # Extract job title
                    job_title = result.find('a',
class_='title').text.strip()

                    # Extract job location
                    job_location = result.find('div',
class_='loc').text.strip()

                    # Extract company name
                    company_name = result.find('span',
class_='cname').text.strip()

                    # Extract experience requirement (assuming it's in
a specific class)
                    experience = result.find('span',
class_='exp').text.strip()

                    # Create a dictionary to store the extracted data
                    job_data = {
                        'job_title': job_title,
                        'job_location': job_location,
                        'company_name': company_name,
                        'experience_required': experience
                    }

                    # Add the job data to the list
                    jobs.append(job_data)

                    # Check if the desired number of jobs has been
collected
                    if len(jobs) == num_jobs:
                        break
                except AttributeError:
                    # Handle cases where specific data elements might
be missing
                    print(f"Error extracting data from job
#{len(jobs)+1}")

            # Increment page number for next iteration
            page_num += 1

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame from the list of job data
    df = pd.DataFrame(jobs)

    return df

if __name__ == "__main__":
    # Replace with actual website URL
    base_url = "https://www.shine.com/"
    search_term = "Data Scientist"
    location = "Delhi/NCR"
    min_salary = 3  # Assumes lakhs
    max_salary = 6  # Assumes lakhs
    num_jobs = 10

    try:
        df = scrape_job_data(base_url, search_term, location,
min_salary, max_salary, num_jobs)
        print(df)

    except ValueError:
        print("Invalid input: Please enter appropriate values for
filters.")
```

4.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_sunglasses_data(base_url, num_pages):
    """Scrapes data for sunglasses listings on Flipkart.

    Args:
        base_url: The base URL of the sunglasses search page.
        num_pages: The number of pages to scrape (up to 100 listings).

    Returns:
        A pandas DataFrame containing the scraped data.
    """

    sunglasses = []
    page_num = 1

    while len(sunglasses) < num_pages * 20 and page_num <= num_pages:
        url = f"{base_url}&page={page_num}"

        try:
            response = requests.get(url)
            response.raise_for_status()  # Raise error for HTTP errors

            soup = BeautifulSoup(response.content, 'html.parser')

            # Find all product result elements (replace with the
appropriate selector)
            product_results = soup.find_all('div', class_='_2jW2V1')

            for product in product_results:
                try:
                    # Extract brand (replace with the appropriate
selector)
                    brand = product.find('div',
class_='_2Wk_oD').text.strip()

                    # Extract product description (replace with the
appropriate selector)
                    description = product.find('a',
class_='_2VVYsW').text.strip()

                    # Extract price (replace with the appropriate
selector)
                    price = product.find('div',
class_='_30jeQf').text.strip()

                    # Add data to dictionary and list
                    sunglasses.append({
                        'brand': brand,
                        'product_description': description,
                        'price': price
                    })

                    # Check if enough sunglasses have been scraped
                    if len(sunglasses) == num_pages * 20:
                        break

                except AttributeError:
                    # Handle cases where data elements are missing
                    print(f"Error extracting data from product
#{len(sunglasses)+1}")
```

```
            # Find "Next" button and click (replace with the
appropriate selector)
            next_button = soup.find('a', class_='_21kZ41')
            if next_button:
                page_num += 1
            else:
                break

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame
    df = pd.DataFrame(sunglasses)

    return df

if __name__ == "__main__":
    base_url =
"https://www.flipkart.com/search?q=sunglasses&otracker=search&otracker1
=hp_banner&page="
    num_pages = 5  # 5 pages to scrape, up to 100 listings

    try:
        df = scrape_sunglasses_data(base_url, num_pages)
        print(df)

    except ValueError:
        print("Invalid input: Please enter an integer for the number of
pages.")
```

5. Please assist to provide proper link so that we can check


flipkart.com/apple-iphone-11-black-64-gb/productreviews/itm4e5041ba101fd?pid=MOBFWQ6BXGJCEYNY&lid=LSTMOBFWQ6BXGJCEYNYZXSHRJ&marketplace=FLIPKART

Unfortunately the page you are looking for has been moved or deleted

6.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_sneaker_data(base_url, num_pages):
    """Scrapes data for sneaker listings on Flipkart.

    Args:
        base_url: The base URL of the sneaker search page.
        num_pages: The number of pages to scrape (up to 100 listings).

    Returns:
        A pandas DataFrame containing the scraped data.
    """

    sneakers = []
    page_num = 1

    while len(sneakers) < num_pages * 20 and page_num <= num_pages:
        url = f"{base_url}&page={page_num}"

        try:
            response = requests.get(url)
            response.raise_for_status()  # Raise error for HTTP errors

            soup = BeautifulSoup(response.content, 'html.parser')

            # Find all product result elements (replace with the
appropriate selector)
            product_results = soup.find_all('div', class_='_2jW2V1')

            for product in product_results:
                try:
                    # Extract brand (replace with the appropriate
selector)
                    brand = product.find('div',
class_='_2Wk_oD').text.strip()

                    # Extract product description (replace with the
appropriate selector)
                    description = product.find('a',
class_='_2VVYsW').text.strip()

                    # Extract price (replace with the appropriate
selector)
                    price = product.find('div',
class_='_30jeQf').text.strip()

                    # Add data to dictionary and list
                    sneakers.append({
                        'brand': brand,
                        'product_description': description,
                        'price': price
                    })

                    # Check if enough sneakers have been scraped
                    if len(sneakers) == num_pages * 20:
                        break

                except AttributeError:
                    # Handle cases where data elements are missing
                    print(f"Error extracting data from product
#{len(sneakers)+1}")

            # Find "Next" button and click (replace with the
appropriate selector)
            next_button = soup.find('a', class_='_21kZ41')
            if next_button:
                page_num += 1
            else:
                break

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame
    df = pd.DataFrame(sneakers)

    return df

if __name__ == "__main__":
    base_url =
"https://www.flipkart.com/search?q=sneakers&otracker=search&otracker1=h
p_banner&page="
    num_pages = 5  # 5 pages to scrape, up to 100 listings

    try:
        df = scrape_sneaker_data(base_url, num_pages)
        print(df)

    except ValueError:
        print("Invalid input: Please enter an integer for the number of
pages.")
```

7.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

```
def scrape_laptops_with_filter(base_url, search_term, filter_name,
filter_value, num_laptops):
    """Scrapes laptop data with applied filters.

    Args:
        base_url: The base URL of the website.
        search_term: The search term for laptops.
        filter_name: The name of the filter to apply.
        filter_value: The value of the filter.
        num_laptops: The number of laptops to scrape.

    Returns:
        A pandas DataFrame containing the scraped laptop data.
    """

    # Construct the search URL with filters
    search_url =
f"{base_url}/s?k={search_term}&{filter_name}%3A{filter_value}"

    laptops = []
    page_num = 1

    while len(laptops) < num_laptops:
        try:
            # Send GET request
            response = requests.get(search_url, params={'page':
page_num})
            response.raise_for_status()  # Raise error for HTTP errors

            # Parse HTML content
            soup = BeautifulSoup(response.content, 'html.parser')

            # Find product result elements (replace with the
appropriate selector)
            product_results = soup.find_all('div', class_='s-search-
item')

            for product in product_results[:num_laptops -
len(laptops)]:
                try:
                    # Extract title (replace with the appropriate
selector)
                    title = product.find('span', class_='a-size-medium
a-color-base a-text-normal').text.strip()

                    # Extract ratings (replace with the appropriate
selector)
                    ratings = product.find('span', class_='a-star-
normal').text.strip()
```

```
                    # Extract price (replace with the appropriate
selector)
                    price = product.find('span', class_='a-price-
whole').text.strip()

                    # Add data to dictionary and list
                    laptops.append({
                        'title': title,
                        'ratings': ratings,
                        'price': price
                    })

                    # Check if the desired number of laptops has been
collected
                    if len(laptops) == num_laptops:
                        break

                except AttributeError:
                    # Handle cases where specific data elements might
be missing
                    print(f"Error extracting data from laptop
#{len(laptops)+1}")

            # Find "Next" button and click (replace with the
appropriate selector)
            next_button = soup.find('a', class_='s-pagination-item s-
pagination-next')
            if next_button:
                page_num += 1
            else:
                break

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame from the list of laptop data
    df = pd.DataFrame(laptops)

    return df

if __name__ == "__main__":
    base_url = "https://www.amazon.in"
    search_term = "laptop"
    filter_name = "cpu_cores"
    filter_value = "Intel Core i7"
    num_laptops = 10

    try:
        df = scrape_laptops_with_filter(base_url, search_term,
filter_name, filter_value, num_laptops)
```

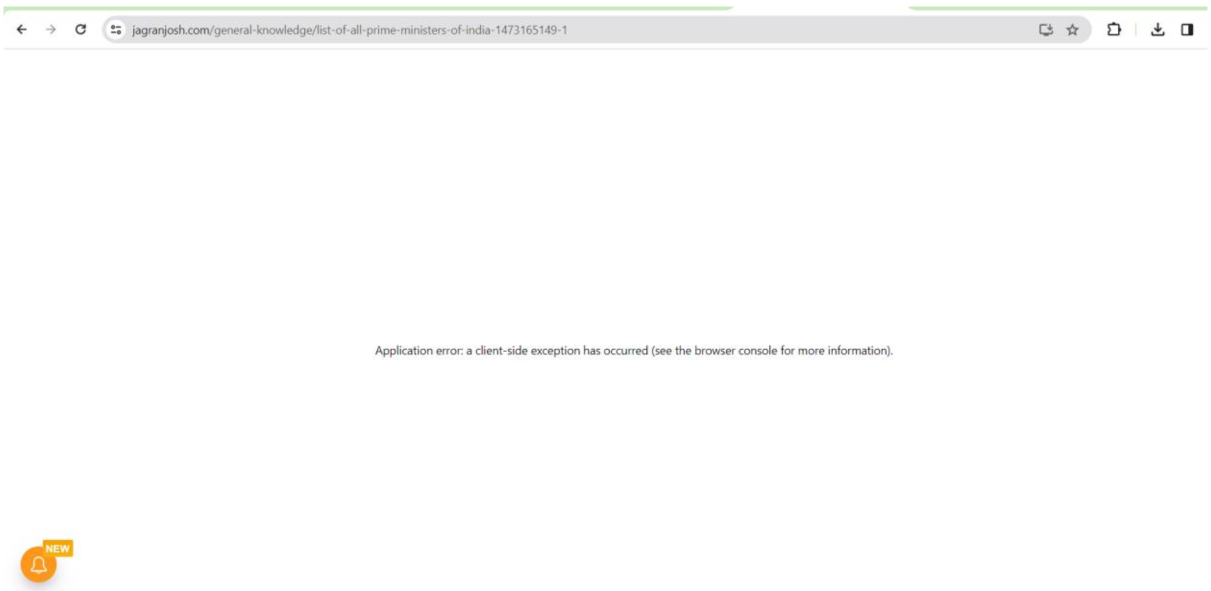
```
        print(df)

    except ValueError:
        print("Invalid input: Please enter appropriate values for 
filters.")
```

8.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_top_quotes(base_url, num_quotes):
    """Scrapes quotes data from the Top Quotes page.

    Args:
        base_url: The base URL of the Top Quotes page.
        num_quotes: The number of quotes to scrape (up to 1000).

    Returns:
        A pandas DataFrame containing the scraped quote data.
    """

    quotes = []
    page_num = 1

    while len(quotes) < num_quotes:
        try:
            # Construct the URL for the current page
            url = f"{base_url}&page={page_num}"

            # Send GET request
            response = requests.get(url)
            response.raise_for_status()  # Raise error for HTTP errors

            # Parse HTML content
            soup = BeautifulSoup(response.content, 'html.parser')

            # Find quote result elements (replace with the appropriate 
selector)
            quote_results = soup.find_all('div', class_='oq-quote')

            for quote_result in quote_results[:num_quotes - 
len(quotes)]:
                try:
                    # Extract quote text (replace with the appropriate 
selector)
```

```
                    quote_text = quote_result.find('span', class_='oq-
q').text.strip()

                    # Extract author name (replace with the appropriate
selector)
                    author_name = quote_result.find('a', class_='oq-
author').text.strip()

                    # Extract quote type (replace with the appropriate
selector)
                    quote_type = quote_result.find('a', class_='oq-
tag').text.strip()

                    # Add data to dictionary and list
                    quotes.append({
                        'quote': quote_text,
                        'author': author_name,
                        'type': quote_type
                    })

                    # Check if enough quotes have been scraped
                    if len(quotes) == num_quotes:
                        break

                except AttributeError:
                    # Handle cases where specific data elements might
be missing
                    print(f"Error extracting data from quote
#{len(quotes)+1}")

            # Find "Next" button and click (replace with the
appropriate selector)
            next_button = soup.find('a', class_='oq-pagination__next')
            if next_button:
                page_num += 1
            else:
                break

        except requests.exceptions.RequestException as e:
            print(f"Error fetching page {page_num}: {e}")
            break

    # Create a pandas DataFrame from the list of quote data
    df = pd.DataFrame(quotes)

    return df

if __name__ == "__main__":
    base_url = "https://www.azquotes.com/topquotes/"
    num_quotes = 1000
    try:
        df = scrape_top_quotes(base_url, num_quotes)
        print(df)

    except ValueError:
        print("Invalid input: Please enter an integer for the number of 
quotes.")
```

9. Page not available after following the steps provided



jagranjosh.com/general-knowledge/list-of-all-prime-ministers-of-india-1473165149-1

Application error: a client-side exception has occurred (see the browser console for more information).

10.

```
import requests
from bs4 import BeautifulSoup

def scrape_expensive_cars(url):
    """Scrapes car names and prices from the given URL.

    Args:
        url: The URL of the webpage containing the car list.

    Returns:
        A list of dictionaries containing car name and price.
    """

    cars = []

    try:
        # Send GET request
        response = requests.get(url)
        response.raise_for_status()  # Raise error for HTTP errors
        # Parse HTML content
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find car result elements (replace with the appropriate
selector)
        car_results = soup.find_all('li', class_='list-item')

        for car_result in car_results[:50]:
            try:
                # Extract car name (replace with the appropriate
selector)
                car_name = car_result.find('span', class_='list-item-
title').text.strip()

                # Extract price (replace with the appropriate selector)
                price_tag = car_result.find('span', class_='list-item-
price')
                price = price_tag.text.split()[0].strip()  # Extract
only the price value

                # Add data to dictionary and list
                cars.append({
                    'car_name': car_name,
                    'price': price
                })

            except AttributeError:
                # Handle cases where specific data elements are missing
                print(f"Error extracting data from car #{len(cars)+1}")

    except requests.exceptions.RequestException as e:
        print(f"Error fetching data: {e}")

    return cars

if __name__ == "__main__":
    url = "https://www.motor1.com/features/308149/most-expensive-new-
cars-ever/"

    try:
        cars = scrape_expensive_cars(url)
        print("Top 50 Most Expensive Cars:")
        for car in cars:
            print(f"{car['car_name']} - ${car['price']}")

    except ValueError:
        print("Error processing data.")
```