Webscrapping assignment 3

December 21, 2023

[ ]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import os
import urllib
```

[ ]:
```
# Question 1

def search_amazon_product(product_name):
    # Defines the base URL for the Amazon search
    base_url = "https://www.amazon.in/s"

    # Sets up parameters for the search query
    params = {"k": product_name}

    # Sends a GET request to Amazon with the search query
    response = requests.get(base_url, params=params)

    # Checks if the request was successful (status code 200)
    if response.status_code == 200:
        # Return the HTML content of the response
        return response.text
    else:
        # Print an error message if the request was not successful
        print(f"Failed to retrieve data. Status code: {response.status_code}")

        # Return None to indicate that the search failed
        return None


# Prompts the user to input the product they want to search on Amazon
user_input = input("Enter the product to search on Amazon: ")

# Calls the search_amazon_product function with the user's input
search_results = search_amazon_product(user_input)
```

[ ]:
```
# Question 2

def scrape_asmazon_results(html_content):
    # Create a BeautifulSoup object to parse the HTML content
    soup = BeautifulSoup(html_content, 'html.parser')

    # Extracts details from the search results
    product_details = []
    for result in soup.select('[data-asin]'):
        # Extract relevant information for each product in the search results
        brand_name = result.find('span', class_='a-size-base-plus␣
 ↪a-color-base').text.strip()
        product_name = result.find('span', class_='a-text-normal').text.strip()
        price = result.find('span', class_='a-offscreen')
        price = price.text.strip() if price else '-'
        return_exchange = result.find('span', class_='a-declarative').text.
 ↪strip()
        expected_delivery = result.find('span', class_='a-text-bold').text.
 ↪strip()
        availability = result.find('div', class_='a-row a-size-base␣
 ↪a-color-secondary').text.strip()
        product_url = result.find('a', class_='a-link-normal')['href']

        # Store the extracted information in a dictionary
        product_details.append({
            'Brand Name': brand_name,
            'Name of the Product': product_name,
            'Price': price,
            'Return/Exchange': return_exchange,
            'Expected Delivery': expected_delivery,
            'Availability': availability,
            'Product URL': product_url
        })

    # Returns a list of dictionaries containing product details
    return product_details

# Checks if search results are available
if search_results:
    # Calls the scrape_amazon_results function to extract details from the␣
 ↪search results HTML
    product_details = scrape_amazon_results(search_results)

    # Converts the scraped data to a DataFrame
    df = pd.DataFrame(product_details)

    # Saves the DataFrame to a CSV file
```

```
    df.to_csv('amazon_search_results.csv', index=False)

    # Prints a success message
    print("Data saved to amazon_search_results.csv")
```

[ ]:
```
# Question 3
def scrape_google_images(keywords, num_images=10):
    # Defines the base URL and parameters for Google image search
    base_url = "https://www.google.com/search"
    params = {"q": "", "tbm": "isch"}

    # Iterates through each keyword in the list
    for keyword in keywords:
        # Sets the search query parameter to the current keyword
        params["q"] = keyword

        # Sends a GET request to Google with the search query
        response = requests.get(base_url, params=params)

        # Checks if the request was successful (status code 200)
        if response.status_code == 200:
            # Parse the HTML content of the response using BeautifulSoup
            soup = BeautifulSoup(response.text, 'html.parser')

            # Finds all image tags with the specified class
            img_tags = soup.find_all('img', class_='t0fcAb')

            # Downloads images and store them in a directory named after the␣
 ↪keyword
            os.makedirs(keyword, exist_ok=True)
            for i, img_tag in enumerate(img_tags[:num_images]):
                img_url = img_tag['src']
                img_path = os.path.join(keyword, f"{keyword}_{i + 1}.jpg")

                # Downloads the image using the URL and save it to the␣
 ↪specified path
                urllib.request.urlretrieve(img_url, img_path)
                print(f"Downloaded: {img_path}")
        else:
            # Prints an error message if the request was not successful
            print(f"Failed to retrieve data for {keyword}. Status code:␣
 ↪{response.status_code}")


# Defines a list of keywords to search for images
keywords_to_search = ['fruits', 'cars', 'Machine Learning', 'Guitar', 'Cakes']
```

```
# Calls the scrape_google_images function with the list of keywords
scrape_google_images(keywords_to_search)
```

[ ]:
```
# Question 4
def scrape_flipkart_smartphones(product_name):
    # Defines the base URL for Flipkart search with the specified product name
    base_url = f"https://www.flipkart.com/search?q={product_name}"

    # Sends a GET request to Flipkart with the search query
    response = requests.get(base_url)

    # Checks if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the response using BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Initializes an empty list to store product details
        product_details = []

        # Extracts details from each product card in the search results
        for card in soup.find_all('div', class_='_1AtVbE'):
            brand_name = card.find('div', class_='_4rR01T').text.strip()
            smartphone_name = card.find('a', class_='IRpwTa').text.strip()
            color = card.find('div', class_='tVe95H').text.strip()
            other_details = [detail.text.strip() for detail in card.
 ↪find_all('li', class_='rgWa7D')]
            price = card.find('div', class_='_30jeq3').text.strip()
            product_url = "https://www.flipkart.com" + card.find('a',␣
 ↪class_='IRpwTa')['href']

            # Appends the extracted details to the product_details list as a␣
 ↪dictionary
            product_details.append({
                'Brand Name': brand_name,
                'Smartphone Name': smartphone_name,
                'Colour': color,
                'RAM': other_details[0] if len(other_details) > 0 else '-',
                'Storage (ROM)': other_details[1] if len(other_details) > 1␣
 ↪else '-',
                'Primary Camera': other_details[2] if len(other_details) > 2␣
 ↪else '-',
                'Secondary Camera': other_details[3] if len(other_details) > 3␣
 ↪else '-',
                'Display Size': other_details[4] if len(other_details) > 4 else␣
 ↪'-',
```

```
                'Battery Capacity': other_details[5] if len(other_details) > 5␣
 ↪else '-',
                'Price': price,
                'Product URL': product_url
            })

        # Converts the scraped data to a DataFrame
        df = pd.DataFrame(product_details)

        # Saves the DataFrame to a CSV file named after the product name
        df.to_csv(f'{product_name}_search_results.csv', index=False)

        # Prints a success message
        print(f"Data saved to {product_name}_search_results.csv")
    else:
        # Prints an error message if the request was not successful
        print(f"Failed to retrieve data for {product_name}. Status code:␣
 ↪{response.status_code}")


# Prompts the user to input the smartphone they want to search on Flipkart
product_to_search = input("Enter the smartphone to search on Flipkart: ")

# Calls the scrape_flipkart_smartphones function with the user's input
scrape_flipkart_smartphones(product_to_search)
```

[ ]:
```
# Question 5

def scrape_google_maps_coordinates(city):
    # Defines the base URL for the Google Maps Geocoding API
    base_url = "https://maps.googleapis.com/maps/api/geocode/json"

    # Sets up parameters for the Geocoding API request, including the city name␣
 ↪and your API key
    params = {
        "address": city,
        "key": "YOUR_GOOGLE_MAPS_API_KEY"  # add the API key
    }

    # Sends a GET request to the Google Maps Geocoding API with the specified␣
 ↪parameters
    response = requests.get(base_url, params=params)

    # Checks if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the JSON content of the response
        data = response.json()
```

```
        # Extracts the geospatial coordinates (latitude and longitude) from the␣
↪response
        location = data.get("results", [])[0].get("geometry", {}).
↪get("location", {})
        latitude = location.get("lat", "-")
        longitude = location.get("lng", "-")

        # Prints the geospatial coordinates for the specified city
        print(f"Geospatial coordinates for {city}: Latitude - {latitude},␣
↪Longitude - {longitude}")
    else:
        # Prints an error message if the request was not successful
        print(f"Failed to retrieve data. Status code: {response.status_code}")

# Prompts the user to input the city they want to search on Google Maps
city_to_search = input("Enter the city to search on Google Maps: ")

# Calls the scrape_google_maps_coordinates function with the user's input
scrape_google_maps_coordinates(city_to_search)
```

[ ]:
```
# Question 6

def scrape_digit_in_gaming_laptops():
    # Defines the base URL for scraping gaming laptops from digit.in
    base_url = "https://www.digit.in/top-products/best-gaming-laptops-40.html"

    # Sends a GET request to the specified URL
    response = requests.get(base_url)

    # Checks if the request was successful (status code 200)
    if response.status_code == 200:
        # Parses the HTML content of the response using BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Initializes an empty list to store laptop details
        laptop_details = []

        # Extracts details from each gaming laptop section in the HTML
        for laptop in soup.find_all('div', class_='TopNumbeHeading active␣
↪sticky-footer'):
            name = laptop.find('div', class_='heading-wraper').text.strip()
            specs = [spec.text.strip() for spec in laptop.find_all('div',␣
↪class_='Top10ProductDetail')]
            rating = laptop.find('div', class_='tdp-number').text.strip()
```

```
            # Appends the extracted details to the laptop_details list as a␣
 ↪dictionary
            laptop_details.append({
                'Name': name,
                'Specifications': ', '.join(specs),
                'Rating': rating
            })

        # Converts the scraped data to a DataFrame
        df = pd.DataFrame(laptop_details)

        # Saves the DataFrame to a CSV file named 'gaming_laptops_details.csv'
        df.to_csv('gaming_laptops_details.csv', index=False)

        # Prints a success message
        print("Data saved to gaming_laptops_details.csv")
    else:
        # Prints an error message if the request was not successful
        print(f"Failed to retrieve data. Status code: {response.status_code}")

# Calls the scrape_digit_in_gaming_laptops function to scrape gaming laptop␣
 ↪details
scrape_digit_in_gaming_laptops()
```

[ ]:
```
# Question 7

def scrape_forbes_billionaires():
    # Defines the base URL for scraping billionaire data from Forbes
    base_url = "https://www.forbes.com/billionaires/"

    # Sends a GET request to the specified URL
    response = requests.get(base_url)

    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parses the HTML content of the response using BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Initializes an empty list to store billionaire details
        billionaires_details = []

        # Extracts details from each billionaire entry in the HTML
        for entry in soup.find_all('div', class_='personInfo'):
            rank = entry.find('div', class_='rank').text.strip()
            name = entry.find('div', class_='personName').text.strip()
            net_worth = entry.find('div', class_='netWorth').text.strip()
            age = entry.find('div', class_='age').text.strip()
```

```
            citizenship = entry.find('div', class_='countryOfCitizenship').text.
 ↪strip()
            source = entry.find('div', class_='source').text.strip()
            industry = entry.find('div', class_='category').text.strip()

            # Appends the extracted details to the billionaires_details list as␣
 ↪a dictionary
            billionaires_details.append({
                'Rank': rank,
                'Name': name,
                'Net worth': net_worth,
                'Age': age,
                'Citizenship': citizenship,
                'Source': source,
                'Industry': industry
            })

        # Converts the scraped data to a DataFrame
        df = pd.DataFrame(billionaires_details)

        # Saves the DataFrame to a CSV file named 'forbes_billionaires_details.
 ↪csv'
        df.to_csv('forbes_billionaires_details.csv', index=False)

        # Prints a success message
        print("Data saved to forbes_billionaires_details.csv")
    else:
        # Print an error message if the request was not successful
        print(f"Failed to retrieve data. Status code: {response.status_code}")


# Calls the scrape_forbes_billionaires function to scrape billionaire details␣
 ↪from Forbes
scrape_forbes_billionaires()
```

[ ]:
```
# Question 8


import googleapiclient.discovery
def get_youtube_comments(api_key, video_id, max_results=500):
    # Build the YouTube API client using the provided API key
    youtube = googleapiclient.discovery.build('youtube', 'v3',␣
 ↪developerKey=api_key)

    # Get video comments
    request = youtube.commentThreads().list(
        part='snippet',
```

```
        videoId=video_id,
        textFormat='plainText',
        maxResults=max_results
    )
    response = request.execute()

    # Initialize an empty list to store comment details
    comments_data = []

    # Extract comment details from the API response
    for item in response.get('items', []):
        comment = item['snippet']['topLevelComment']['snippet']
        comments_data.append({
            'Comment': comment['textDisplay'],
            'Upvotes': comment['likeCount'],
            'Time': comment['publishedAt']
        })

    # Return the list of comment details
    return comments_data

# Example usage:
# Replace 'YOUR_API_KEY' with your actual YouTube API key
api_key = 'YOUR_API_KEY'
video_id = '0X0Jm8QValY'  # Extracted from the YouTube video URL

# Call the get_youtube_comments function to retrieve comments data
comments_data = get_youtube_comments(api_key, video_id)

# Convert the retrieved data to a DataFrame and save it to CSV
df = pd.DataFrame(comments_data)
df.to_csv('youtube_comments_data_video.csv', index=False)

# Print a success message
print("Comments data saved to youtube_comments_data_video.csv")
```

[ ]:
```
# Question 9

def scrape_hostels_in_london():
    # Defines the base URL for scraping hostels in London from hostelworld.com
    base_url = "https://www.hostelworld.com/s?
 ↪q=London&country=England&city=London&type=city&id=3"

    # Sends a GET request to the specified URL
    response = requests.get(base_url)

    # Checks if the request was successful (status code 200)
```

```
    if response.status_code == 200:
        # Parse the HTML content of the response using BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Initializes an empty list to store hostel details
        hostels_details = []

        # Extracts details from each hostel entry in the HTML
        for hostel in soup.find_all('div', class_='property-card'):
            name = hostel.find('h2', class_='title').text.strip()
            distance = hostel.find('span', class_='description').text.strip()
            ratings = hostel.find('div', class_='score orange big').text.strip()
            reviews = hostel.find('span', class_='reviews').text.strip()
            overall_reviews = hostel.find('div', class_='keyword').text.strip()
            price_privates = hostel.find('div', class_='price-col').text.strip()
            price_dorms = hostel.find('div', class_='price-col').text.strip()
            facilities = ', '.join([f.text.strip() for f in hostel.
↪find_all('div', class_='facilities-item')])
            description = hostel.find('div', class_='description-container').
↪text.strip()

            # Appends the extracted details to the hostels_details list as a␣
↪dictionary
            hostels_details.append({
                'Hostel Name': name,
                'Distance from City Centre': distance,
                'Ratings': ratings,
                'Total Reviews': reviews,
                'Overall Reviews': overall_reviews,
                'Privates from Price': price_privates,
                'Dorms from Price': price_dorms,
                'Facilities': facilities,
                'Property Description': description
            })

        # Converts the scraped data to a DataFrame
        df = pd.DataFrame(hostels_details)

        # Saves the DataFrame to a CSV file named 'london_hostels_details.csv'
        df.to_csv('london_hostels_details.csv', index=False)

        # Prints a success message
        print("Data saved to london_hostels_details.csv")
    else:
        # Print an error message if the request was not successful
        print(f"Failed to retrieve data. Status code: {response.status_code}")
```

```
# Calls the scrape_hostels_in_london function to scrape hostel details in London
scrape_hostels_in_london()
```

[ ]:

[ ]:

[ ]: