WEB SCRAPING-ASSIGNMENT3

Instructions:

- All questions are compulsory.
- In each of the questions you have to automate the process. You do not have to click on any button, click any clickable element, enter keywords in search boxes manually. Each process has to be performed via coding.
- Q1 and Q2 are connected questions i.e. after attempting Q1 proceed to Q2. Do not write whole code from beginning for Q2.
- You may use any web scraping library and tools.
- The question can be attempted in various ways; the correctness of question depends on the output.
- If you encounter any Null values during scraping, you may replace it by hyphen.

Exercise:

1. Write a python program which searches all the product under a particular product from www.amazon.in. The product to be searched will be taken as input from user. For e.g. If user input is 'guitar'. Then search for guitars.

Answer:
```
import requests
from bs4 import BeautifulSoup


def search_amazon_products(product_name):
    # Format the product name for the search URL
    formatted_product = product_name.replace(' ', '+')


    # Send a GET request to Amazon.in search page
    url = f"https://www.amazon.in/s?k={formatted_product}"
    response = requests.get(url)
    response.raise_for_status()


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


    # Find all the product listings
    product_listings = soup.find_all('div', {'data-component-type': 's-search-result'})
    # Iterate over the product listings and extract relevant information
    for listing in product_listings:
        # Extract the product title
        title_element = listing.find('span', {'class': 'a-size-medium'})
        if title_element:
            title = title_element.text.strip()
        else:
            continue

        # Extract the product price
        price_element = listing.find('span', {'class': 'a-offscreen'})
        if price_element:
            price = price_element.text.strip()
        else:
            continue

        # Extract the product rating
        rating_element = listing.find('span', {'class': 'a-icon-alt'})
        if rating_element:
            rating = rating_element.text.strip()
        else:
            rating = "N/A"

        # Print the product details
        print(f"Title: {title}")
        print(f"Price: {price}")
        print(f"Rating: {rating}")
        print()

# Take user input for the product to search
```

```
product = input("Enter the product to search on Amazon.in: ")

# Call the search_amazon_products function
search_amazon_products(product)
```

2. In the above question, now scrape the following details of each product listed in first 3 pages of your search results and save it in a data frame and csv. In case if any product has less than 3 pages in search results then scrape all the products available under that product name. Details to be scraped are: "Brand Name", "Name of the Product", "Price", "Return/Exchange", "Expected Delivery", "Availability" and "Product URL". In case, if any of the details are missing for any of the product then replace it by "-".

Answer:

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

def scrape_product_details(product_name):
    formatted_product = product_name.replace(' ', '+')
    page_number = 1
    products_data = []

    while True:
        # Send a GET request to the search page
        url = f"https://www.amazon.in/s?k={formatted_product}&page={page_number}"
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all the product listings on the current page
        product_listings = soup.find_all('div', {'data-component-type': 's-search-result'})

        if not product_listings:
            break  # Break the loop if no more products found on the page
        for listing in product_listings:
            # Extract the product details
            title_element = listing.find('span', {'class': 'a-size-medium'})
            title = title_element.text.strip() if title_element else '-'

            price_element = listing.find('span', {'class': 'a-offscreen'})
            price = price_element.text.strip() if price_element else '-'

            rating_element = listing.find('span', {'class': 'a-icon-alt'})
            rating = rating_element.text.strip() if rating_element else '-'

            brand_element = listing.find('span', {'class': 'a-size-base-plus'})
            brand = brand_element.text.strip() if brand_element else '-'

            return_exchange_element = listing.find('div', {'class': 'a-row a-size-small'})
            return_exchange = return_exchange_element.text.strip() if return_exchange_element else '-
'

            expected_delivery_element = listing.find('span', {'class': 'a-text-bold'})
            expected_delivery = expected_delivery_element.text.strip() if expected_delivery_element
else '-'

            availability_element = listing.find('span', {'class': 'a-size-base-plus a-color-success a-text-
bold'})
            availability = availability_element.text.strip() if availability_element else '-'

            product_url = listing.find('a', {'class': 'a-link-normal s-no-outline'})
            product_url = 'https://www.amazon.in' + product_url['href'] if product_url else '-'

            # Create a dictionary of product details
            product_data = {
              'Brand Name': brand,
              'Name of the Product': title,
              'Price': price,
              'Return/Exchange': return_exchange,
              'Expected Delivery': expected_delivery,
              'Availability': availability,
              'Product URL': product_url
            }

            # Append the product data to the list
            products_data.append(product_data)

        page_number += 1

      # Create a dataframe from the list of product data
      df = pd.DataFrame(products_data)

      return df


# Take user input for the product to search
product = input("Enter the product to search on Amazon.in: ")

# Scrape product details and create dataframe
data_frame = scrape_product_details(product)

# Save the dataframe to a CSV file
data_frame.to_csv('amazon_products.csv', index=False)
```

3. Write a python program to access the search bar and search button on images.google.com and scrape 10 images each for keywords ‘fruits’, ‘cars’ and ‘Machine Learning’, ‘Guitar’, ‘Cakes’.

Answer:
```
import requests
from bs4 import BeautifulSoup
```

```
def scrape_google_images(keyword, num_images):
    # Format the keyword for the search URL
    formatted_keyword = keyword.replace(' ', '+')

    # Send a GET request to images.google.com search page
    url = f"https://www.google.com/search?q={formatted_keyword}&tbm=isch"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
    response = requests.get(url, headers=headers)
    response.raise_for_status()

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

    # Find all the image elements
    image_elements = soup.find_all('img', {'class': 'rg_i'})

    # Scrape the image URLs
    image_urls = []
    for i, element in enumerate(image_elements):
        if i >= num_images:
            break
        image_url = element.get('src')
        if image_url and image_url.startswith("http"):
            image_urls.append(image_url)

    return image_urls


# Keywords to search for
```

```
keywords = ['fruits', 'cars', 'Machine Learning', 'Guitar', 'Cakes']

# Number of images to scrape for each keyword
num_images = 10

# Scrape images for each keyword
for keyword in keywords:
    image_urls = scrape_google_images(keyword, num_images)
    print(f"Images for keyword '{keyword}':")
    for url in image_urls:
        print(url)
    print()
```

4. Write a python program to search for a smartphone(e.g.: Oneplus Nord, pixel 4A, etc.) on www.flipkart.com and scrape following details for all the search results displayed on 1st page. Details to be scraped: “Brand Name”, “Smartphone name”, “Colour”, “RAM”, “Storage(ROM)”, “Primary Camera”, “Secondary Camera”, “Display Size”, “Battery Capacity”, “Price”, “Product URL”. Incase if any of the details is missing then replace it by “- “. Save your results in a dataframe and CSV.

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

def scrape_flipkart_smartphones(product):
    # Format the product name for the search URL
    formatted_product = product.replace(' ', '+')

    # Send a GET request to Flipkart search page
    url = f"https://www.flipkart.com/search?q={formatted_product}"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
    response = requests.get(url, headers=headers)
    response.raise_for_status()
```

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

# Find all the product listings on the first page
product_listings = soup.find_all('div', {'class': '_1AtVbE'})

# Create lists to store the scraped data
brand_names = []
smartphone_names = []
colours = []
rams = []
storages = []
primary_cameras = []
secondary_cameras = []
display_sizes = []
battery_capacities = []
prices = []
product_urls = []

# Iterate over the product listings and extract the details
for listing in product_listings:
    # Extract the brand name
    brand_element = listing.find('div', {'class': '_4rR01T'})
    brand_name = brand_element.text.strip() if brand_element else '-'
    brand_names.append(brand_name)

    # Extract the smartphone name
    smartphone_element = listing.find('a', {'class': 'IRpwTa'})
    smartphone_name = smartphone_element.text.strip() if smartphone_element else '-'
    smartphone_names.append(smartphone_name)
```

```
        # Extract the color
        colour_element = listing.find('div', {'class': '_4rR01T'})
        colour = colour_element.text.strip() if colour_element else '-'
        colours.append(colour)

        # Extract the RAM
        ram_element = listing.find('li', text=lambda text: text and 'RAM' in text)
        ram = ram_element.text.split(' ')[0] if ram_element else '-'
        rams.append(ram)

        # Extract the storage (ROM)
        storage_element = listing.find('li', text=lambda text: text and 'ROM' in text)
        storage = storage_element.text.split(' ')[0] if storage_element else '-'
        storages.append(storage)

        # Extract the primary camera
        primary_camera_element = listing.find('li', text=lambda text: text and 'Primary Camera' in
text)
        primary_camera = primary_camera_element.text.split(' ')[0] if primary_camera_element else
'-'
        primary_cameras.append(primary_camera)

        # Extract the secondary camera
        secondary_camera_element = listing.find('li', text=lambda text: text and 'Secondary Camera'
in text)
        secondary_camera = secondary_camera_element.text.split(' ')[0] if
secondary_camera_element else '-'
        secondary_cameras.append(secondary_camera)

        # Extract the display size
        display_size_element = listing.find('li', text=lambda text: text and 'Display Size' in text)
        display_size = display_size_element.text.split(' ')[0] if display_size_element else '-'
        display_sizes.append(display_size)

        # Extract the battery capacity
        battery_capacity_element = listing.find('li', text=lambda text: text and 'Battery Capacity' in text)
        battery_capacity = battery_capacity_element.text.split(' ')[0] if battery_capacity_element else '-'
        battery_capacities.append(battery_capacity)

        # Extract the price
        price_element = listing.find('div', {'class': '_30jeq3 _1_WHN1'})
        price = price_element.text.replace('₹', '').strip() if price_element else '-'
        prices.append(price)

        # Extract the product URL
        product_url = listing.find('a', {'class': 'IRpwTa'})
        product_url = 'https://www.flipkart.com' + product_url['href'] if product_url else '-'
        product_urls.append(product_url)

    # Create a dataframe from the scraped data
    df = pd.DataFrame({
        'Brand Name': brand_names,
        'Smartphone Name': smartphone_names,
        'Colour': colours,
        'RAM': rams,
        'Storage(ROM)': storages,
        'Primary Camera': primary_cameras,
        'Secondary Camera': secondary_cameras,
        'Display Size': display_sizes,
        'Battery Capacity': battery_capacities,
        'Price': prices,
        'Product URL': product_urls
    })

    return df


# Take user input for the smartphone to search
product = input("Enter the smartphone to search on Flipkart: ")

# Scrape smartphone details and create dataframe
data_frame = scrape_flipkart_smartphones(product)

# Save the dataframe to a CSV file
data_frame.to_csv('flipkart_smartphones.csv', index=False)
```

5. Write a program to scrap geospatial coordinates (latitude, longitude) of a city searched on google maps.

Answer:

```
import requests
from bs4 import BeautifulSoup

def scrape_google_maps_coordinates(city):
    # Format the city name for the search URL
    formatted_city = city.replace(' ', '+')

    # Send a GET request to Google Maps search page
    url = f"https://www.google.com/maps/search/{formatted_city}"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
    response = requests.get(url, headers=headers)
    response.raise_for_status()

    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(response.text, 'html.parser')
    # Find the map element
    map_element = soup.find('div', {'class': 'tactile-searchbox-input'})

    # Extract the coordinates from the map element
    coordinates = map_element.get('data-value') if map_element else None

    return coordinates

# Take user input for the city to search
city = input("Enter the city name to search on Google Maps: ")

# Scrape the coordinates
coordinates = scrape_google_maps_coordinates(city)

if coordinates:
    print(f"Coordinates of {city}: {coordinates}")
else:
    print(f"No coordinates found for {city}.")
```

6. Write a program to scrap all the available details of best gaming laptops from digit.in.

Answer:

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

def scrape_digit_in_gaming_laptops():
    # Send a GET request to digit.in gaming laptops page
    url = "https://www.digit.in/top-products/best-gaming-laptops-40.html"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
    response = requests.get(url, headers=headers)
    response.raise_for_status()


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


    # Find the list of gaming laptop listings
    laptop_listings = soup.find_all('div', {'class': 'TopNumbeHeading'})


    # Create lists to store the scraped data
    laptop_names = []
    operating_systems = []
    display_sizes = []
    processors = []
    memories = []
    graphics_cards = []
    prices = []


    # Iterate over the laptop listings and extract the details
    for listing in laptop_listings:
        # Extract the laptop name
        laptop_name = listing.find('div', {'class': 'TopNumbeHeading'}).text.strip()
        laptop_names.append(laptop_name)


        # Extract the operating system
        operating_system = listing.find('div', text=lambda text: text and 'OS' in
text).text.split(':')[1].strip()
        operating_systems.append(operating_system)


        # Extract the display size
        display_size = listing.find('div', text=lambda text: text and 'Display' in
text).text.split(':')[1].strip()
        display_sizes.append(display_size)
        # Extract the processor
        processor = listing.find('div', text=lambda text: text and 'Processor' in
text).text.split(':')[1].strip()
        processors.append(processor)

        # Extract the memory
        memory = listing.find('div', text=lambda text: text and 'Memory' in text).text.split(':')[1].strip()
        memories.append(memory)

        # Extract the graphics card
        graphics_card = listing.find('div', text=lambda text: text and 'Graphics' in
text).text.split(':')[1].strip()
        graphics_cards.append(graphics_card)

        # Extract the price
        price = listing.find('div', {'class': 'smprice'}).text.strip()
        prices.append(price)

    # Create a dataframe from the scraped data
    df = pd.DataFrame({
        'Laptop Name': laptop_names,
        'Operating System': operating_systems,
        'Display Size': display_sizes,
        'Processor': processors,
        'Memory': memories,
        'Graphics Card': graphics_cards,
        'Price': prices
    })

    return df
```

# Scrape gaming laptop details and create dataframe

data_frame = scrape_digit_in_gaming_laptops()

# Save the dataframe to a CSV file

data_frame.to_csv('digit_in_gaming_laptops.csv', index=False)

7. Write a python program to scrape the details for all billionaires from www.forbes.com. Details to be scrapped: “Rank”, “Name”, “Net worth”, “Age”, “Citizenship”, “Source”, “Industry”.

Answer: import requests

```
import pandas as pd

from bs4 import BeautifulSoup


def scrape_forbes_billionaires():

   # Send a GET request to Forbes billionaires page

   url = "https://www.forbes.com/billionaires/"

   headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}

   response = requests.get(url, headers=headers)

   response.raise_for_status()


   # Parse the HTML content using BeautifulSoup

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


   # Find the table that contains the billionaire details

   table = soup.find('table', {'class': 'table'})


   # Find the table headers

   headers = table.find('thead').find_all('th')


   # Get the column names
```

```
column_names = [header.text.strip() for header in headers]

# Create lists to store the scraped data
data = []

# Find all the rows in the table body
rows = table.find('tbody').find_all('tr')

# Iterate over the rows and extract the details
for row in rows:
    # Find all the cells in the row
    cells = row.find_all('td')

    # Extract the details from each cell
    rank = cells[0].text.strip()
    name = cells[1].text.strip()
    net_worth = cells[2].text.strip()
    age = cells[3].text.strip()
    citizenship = cells[4].text.strip()
    source = cells[5].text.strip()
    industry = cells[6].text.strip()

    # Append the details to the data list
    data.append([rank, name, net_worth, age, citizenship, source, industry])

# Create a dataframe from the scraped data
df = pd.DataFrame(data, columns=column_names)

return df
```

```
# Scrape billionaire details and create dataframe
data_frame = scrape_forbes_billionaires()


# Save the dataframe to a CSV file
data_frame.to_csv('forbes_billionaires.csv', index=False)
```

8. Write a program to extract at least 500 Comments, Comment upvote and time when comment was posted from any YouTube Video.

Answer:

```
from googleapiclient.discovery import build
import pandas as pd


# YouTube API key (replace with your own API key)
API_KEY = 'YOUR_API_KEY'


# YouTube video ID (replace with the desired video ID)
VIDEO_ID = 'VIDEO_ID'


# Number of comments to fetch
MAX_RESULTS = 500


# Create a YouTube Data API client
youtube = build('youtube', 'v3', developerKey=API_KEY)


# Fetch the comments for the specified video
def get_video_comments(video_id, max_results):
  comments = []
  comment_upvotes = []
  comment_times = []


  # Get the video comment thread list
  response = youtube.commentThreads().list(
    part='snippet',
    videoId=video_id,
    order='relevance',
    textFormat='plainText',
    maxResults=max_results
  ).execute()

  # Iterate over the comment threads
  for item in response['items']:
    comment = item['snippet']['topLevelComment']['snippet']['textDisplay']
    comments.append(comment)

    upvotes = item['snippet']['topLevelComment']['snippet']['likeCount']
    comment_upvotes.append(upvotes)

    time = item['snippet']['topLevelComment']['snippet']['publishedAt']
    comment_times.append(time)

  return comments, comment_upvotes, comment_times

# Fetch comments for the specified video
comments, upvotes, times = get_video_comments(VIDEO_ID, MAX_RESULTS)

# Create a dataframe from the scraped data
df = pd.DataFrame({
  'Comment': comments,
  'Upvotes': upvotes,
  'Time': times
})

# Save the dataframe to a CSV file
df.to_csv('youtube_comments.csv', index=False)
```

9. Write a python program to scrape a data for all available Hostels from https://www.hostelworld.com/ in “London” location. You have to scrape hostel name, distance from city centre, ratings, total reviews, overall reviews, privates from price, dorms from price, facilities and property description.

Answer:

```
import requests

import pandas as pd

from bs4 import BeautifulSoup


def scrape_hostel_data():

    # Send a GET request to the Hostelworld page for London hostels

    url = "https://www.hostelworld.com/search?city=London&country=England"

    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}

    response = requests.get(url, headers=headers)

    response.raise_for_status()


    # Parse the HTML content using BeautifulSoup

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


    # Find all the hostel listings

    listings = soup.find_all('div', {'class': 'property-card'})


    # Create lists to store the scraped data

    hostel_names = []

    distances = []

    ratings = []

    total_reviews = []

    overall_reviews = []

    privates_prices = []

    dorms_prices = []

    facilities = []

    descriptions = []
```

```
# Iterate over the hostel listings and extract the details
for listing in listings:
   # Extract the hostel name
   hostel_name = listing.find('h2', {'class': 'title'}).text.strip()
   hostel_names.append(hostel_name)

   # Extract the distance from city centre
   distance = listing.find('span', {'class': 'distance-title'}).text.strip()
   distances.append(distance)

   # Extract the rating
   rating = listing.find('div', {'class': 'score orange'}).text.strip()
   ratings.append(rating)

   # Extract the total reviews count
   total_review = listing.find('div', {'class': 'reviews'}).text.strip().split()[0]
   total_reviews.append(total_review)

   # Extract the overall reviews rating
   overall_review = listing.find('div', {'class': 'score'}).text.strip()
   overall_reviews.append(overall_review)

   # Extract the private room prices
   private_price = listing.find('div', {'class': 'price-col'})
   privates_prices.append(private_price.text.strip() if private_price else '-')

   # Extract the dorm room prices
   dorm_price = listing.find('div', {'class': 'price-col alt-price'})
   dorms_prices.append(dorm_price.text.strip() if dorm_price else '-')
```

```
        # Extract the facilities
        facility_tags = listing.find_all('div', {'class': 'facilities'})
        facility_list = [tag.text.strip() for tag in facility_tags]
        facilities.append(', '.join(facility_list))

        # Extract the property description
        description = listing.find('div', {'class': 'property-description'}).text.strip()
        descriptions.append(description)

    # Create a dataframe from the scraped data
    df = pd.DataFrame({
        'Hostel Name': hostel_names,
        'Distance from City Centre': distances,
        'Rating': ratings,
        'Total Reviews': total_reviews,
        'Overall Reviews': overall_reviews,
        'Privates from Price': privates_prices,
        'Dorms from Price': dorms_prices,
        'Facilities': facilities,
        'Property Description': descriptions
    })

    return df


# Scrape hostel data for London and create dataframe
data_frame = scrape_hostel_data()

# Save the dataframe to a CSV file
data_frame.to_csv('hostel_data.csv', index=False)
```