My-name - vaishali Ramdas savve
Batch No - DS 2311
sir = Shwetank Mishra
Date = 30-1-2024

PAGE No.
DATE / /

Assignment - 2
Web-Scraping

Q1 write a Python program to scrape data for "data Analyst" job position in "Bangalore" location. You have to scrape the job-title, job-location, company name, experience-required. You have to scrape first 10 jobs data.
This task will be done in following steps:

1. First get the webpage https://www.sine.com/
2. Enter "data Analyst" in job title, skills filed and enter "Bangalore" in enter the location" field.
3. Then click the Searchbutton.
4. Then scrape the data for the first 10 jobs results you get.
5) Finally create a dataframe of the scraped data.

⇒

To achieve this task, you can use the requests library to send HTTP requests to the website and BeautifulSoup for parsing the HTML content. Additionally you may use the 'pandas' library to create a Data Frame for the scraped data. Here's a sample Python program to get you started.

PAGE No.
DATE / /

python

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
# step 1: get the webpage
url = "https://www.shine.com/"
response = requests.get(url)

# check if the request was successful
if response.status_code == 200:
    print("Successfully fetched the webpage.")
else:
    print(f"Failed to fetch the webpage status code:
        {response.status_code}")
    exit()
# step 2: Enter search criteria and click the search butt
search_data = {
    'q': 'Data Analyst',
    'l': 'Bangalore'
}

search_url = "https://www.shine.com/job-search/dat
    analyst-jobs-in-bangalore"
search_response = requests.get(search_url, params=search
                               data)
if search_response.status_code == 200:
    print("successfully performed the search:")
else:
    print(f"Failed to perform the search. Status code:
        {search_response.status_code}")
    exit()
# step 3: Scrape the data for the first 40 jobs
soup = BeautifulSoup(search_response.text, 'html.parser')
```

```
job_listings = soup.find_all('div', class_='search listing
data = []
for job in job_listings[:10]:
title = job.find('h3', class_='job-title').text.strip()
location = job.find('span', class_='job-location').text.strip()
company_name = job.find('span', class_='job-comp-name
                        ').text.strip()
experience_required = job.find('span', class_='job-exp
                               ').text.strip()
data.append({
'Job Title': title,
'Job Location': location,
'Company Name': company_name,
'Experience Required': experience_required
})
# step 4: create a Data Frame
df = pd.DataFrame(data)
# Display the Data Frame
print(df)
```

Note: The code above assumes the structure of the HTML elements on the shine.com website as of my last knowledge update in January 2022. website structure may change, so you may need to inspect the HTML of the website and adjust the code accordingly if there are any changes.

Q2 write a python program to scrape data for data scientist" job position in "Bangalore" location. you have to scrape the job-title, job-location, company name. you have to scrape first 10 jobs data.

This task will be done in following steps:

1. First get the web page https://www.shine.com/
2. Enter data "Scientist" in "job title, skills field and "Bangalore" in "enter the location" field.
3. Then click the search button.

Then scrape the data for the first 10 jobs results you get.

Finally create a dataframe of the.

⇒ To achieve achieve this task, you can follow a similar approach as in the previous example. I'll use the 'requests' library for sending HTTP requests, 'BeautifulSoup' for parsing the HTML content, and 'pandas' to create a DataFrame. Here's a sample python program:

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

# Step 1: Get the web page
url = "https://www.shine.com/"
response = requests.get(url)
# Check if the request was successful
if Response.status_code == 200:
print("Successfully fetched the webpage")
else:
```

print (f" Failed to fetch the webpages. status code
& response.status_code }")
exit()
# step 2 : Enter search criteria and click the
search Button
search_data = {
'q' : 'data scientist',
'l' : 'Bangalore',
}
search_url = "https://www.shine.com/job-search/
data-scientist-jobs-in-Bangalore"
search_response = requests.get(search_url, params=search
_data)
if search_response.status_code == 200:
print ("successfully performed the search.")
else
print(f"Failed to perform the search, status code: {
search_response.status_code}")
exit()
# step 3: Scrape the data for the first 10 jobs
soup = BeautifulSoup(search_response.text, 'html.parser')
job_listing = soup.find_all('div', class_='search-listing')
data = []
for job in job_listing[:10]:
title = job.find('h3', class_='job-title').text.strip()
location = job.find('span', class_='job-location').text.strip()
company_name = job.find('span', class_='jobname').text.strip()
data.append({
'Job Title': title,
'Job location': location,
'Company name': company_name
})

# Step 4: create a Data Frame
df = pd.DataFrame(data)
# Display the Data Frame
print(df)

As mentioned earlier the structure of the HTML elements may change over time, so make sure to inspect the HTML of the website and adjust the code accordingly if there are any changes.

Q3 In this question you have to scrape data using the filters available on the webpage you have to use the location and salary filter.
You have to scrape data for "Data Scientist" designation for first 10 job results.
You have to scrape the job-title, job-location, company name, experience required.
The location filter to be used is "Delhi/NCR". The salary filter to be used is "3-6" lakhs
The task will be done in following steps:

1. First get the web page https://www.shine.com/
2. Enter "Data Scientist" in Skill Designation and Company filed.
3. Then click the search button.
4. Then apply the location filter and salary filter by checking the respective boxes.
5. Then scrape the data for the first 10 jobs results you get.
6. Finally create a dataframe of the scraped data.

PAGE No.
DATE / /

→ To achieve this task you can the request library to send HTTP request to the website and BeautifulSoup for parsing the HTML content. Additionally, you may use the pandas library to create a DataFrame for the scraped data. Here's a sample Python program

Copy code

```
python
import requests
from bs4 import BeautifulSoup
import pandas as pd
# Step 1: Get the webpage.
url = "https://www.shine.com/"
response = requests.get(url)
# check if the request we was successful
if response.status_code == 200:
    print("Successfully fetched the webpage.")
else:
    print(f"Failed to fetch the webpage. Status code
        : {response.status_code}")
# Step 2: Enter search criteria and click the search button
search_data = {
    'q': 'Data Scientist'
}
search_url = "https://www.shine.com/job-search/
                data-scientist-jobs"
search_response = requests.get(search_url, params=search_
                                data)
if search_response.status_code == 200:
    print("Successfully performed the search.")
else:
    print(f"Failed to perform the search. Status code: {search
        _response.status_code}")
exit()
```

Step 2: Enter search criteria and click the search button

```
search_data = {
    'q': 'Data Scientist'
}
search_url = "https://www.shine.com/job-search/data-scientist-jobs"
search_response = requests.get(search_url, params=search_data)
if search_response.status_code == 200:
    print("Successfully performed the search")
else:
    print("Failed to perform the search. Status code:", search_response.status_code)
    exit()
```

Step 3: Apply location and salary filters

```
filters_data = {
    'location': 'Delhi / NCR',
    'salary': '3-6 Lakhs'
}
filter_url = https://www.shine.com/job-search/data-scientist-jobs
filter_response = requests.post(filter_url, params=search_data, data=filters_data)
if filter_response.status_code == 200:
    print("Successfully applied filters")
else:
    print("Failed to apply filters", status_code:", filter_response.status_code)
    exit()
```

Step 4: Scrape the data for the first 10 jobs

```
soup = BeautifulSoup(filter_response.text, 'html.parser')
job_listing = soup.find_all('div', class_="search-listing")
data = []
```

```
for job in job_listing [:10]:
    title = job.find('h3', class_='job-title').text.strip()
    location = job.find('span', class_='job-location').text.strip()
    company_name = job.find('span', class_='job-comp-name').text.strip()
    experience_required = job.find('span', class_='job-exp').text.strip()
    data.append({
        'Job Title': title
        'Job Location': location
        'Company Name': company_name
        'Experience Required': experience_required
    })
# Step 5: Create a Data Frame
df = pd.DataFrame(data)
# Display the Data Frame
print(df)
```

As always keep in mind that website structures may change. So you may need to inspect the HTML of the website and adjust the code accordingly if there are any changes.

PAGE No.
DATE / /

Q 4 Scrape data of first 100 Sunglasses listing on Flipk
you have to scrape four attributes.
Brand (7) product description (8) price.
The attributes which you have to scrape is ticked
marked in the below image.
To scrape the data you have to go through following
1 Go to Flipkart webpage. by url: https://www.flipkart.co
2 Enter "Sunglasses" in the search field where "Searc
for products brand and more" is written and
click the search icon.
3 After that you will reach to the page having a lot of
Sunglasses. form this page you can scrape three requ
data as usual. 4 After scraping data from the firs
page, go to the 'next' Button at the bottom of the pag
then. Click on it.
5 Now scrape data from this page as usual.
Repeat this until you get data for 100 sunglasses

⇒ To achieve this task, you can the requests' library to send HTTP requests to the website, BeautifulSoup for parsing the HTML content, and pandas to create a Data Frame for the scraped data. Additionally you might need to handle pagination By navigating through the "next Button. Here's a simple Python program.

PAGE No.
DATE / /

Python

copy code

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
# Function to scrape sunglasses data from a given Flipkart page.
def scrape_flipkart_sunglasses(url, count):
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Scraping data from page {count}...")
    else:
        print(f"Failed to fetch the webpage. Status code: {response.status_code}")
        return None
    soup = BeautifulSoup(response.text, 'html.parser')
    sunglasses_listings = soup.find_all('div', class_='_1AtVbE')
    data = []
    for sunglasses in sunglasses_listings:
        brand = sunglasses.find('div', class_='_2WkVRV').text.strip()
        description = sunglasses.find('a', class_='IRpwTa').text.strip()
        price = sunglasses.find('div', {'data-id': True}).text.strip()
        data.append({
            'Brand': brand,
            'Product Description': description,
            'Price': price
        })
    return data
# Main scraping logic
base_url = "https://www.flipkart.com"
search_url = "/search?q=sunglasses"
sunglasses_data = []
```

loop through multiple pages until we have data for 100 sunglasses.

```
page_count = 1
while (sunglasses_data) < 1008:
    current_url = base_url + search_url + &page= + str(page_count)
    current_data = scrape_flipkart_sunglasses(current_url)
    if not current_data:
        break
    sunglasses_data.extend(current_data)
    page_count += 1
# create a DataFrame
df = pd.DataFrame(sunglasses_data[:1008])
# Display the DataFrame
print(df)
```

This code navigates through the pages by incrementing the 'page_count' and concatenates the data until we have information for 100 sunglasses. As always keep in mind that the website structure may change, so you may need to inspect the HTML of the website and adjust the code accordingly if there are any changes.

PAGE No.
DATE / /

Q5. Scrape 100 review data from flipkart.com for iphone 11 phone. you have to go the link: https://www.flipkart.com/apple-iphone-11-black-64-gb/product-reviews/itm4e5041ba101fd?pid=MOBFWQ6BXGJCEYNY&lid=LSTMOBFWQ6BXGJCEYNY2X5HRJ&market.

Place As shown in the above page you have to scrape the tick market attributes. these are

(1) Rating (2) Review Summary (3) Full Review.

(4) you have to scrape this data for first 100 review.

⇒ To scrape 100 reviews data from Flipkart for the iphone 11, you can use the 'request' library to send HTTP request to the specified URL: 'BeautifulSoup' for parsing the HTML content and 'pandas' to create a Data frame for the scraped data. Here's sample Python program:

Python — Copy code

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
# function to scrape reviews data from a given
  Flipkart product review.
def scrape_flipkart_reviews(url, count):
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Scraping review data from page {count}...")
    else: print(f"Failed to fetch the web page. Status code=
```

```
f response.status_code=3")
return none
soup = BeautifulSoup(response.text,'html.parser')
review_listings = soup.find_all('div', class_='z=mVg')
data = []
for review in reviews_listings
rating = review.find('div',class_='a1w21k').text.strip
summary = review.find('p',class_='-2-nvB2T').text.strip
full review = review.find('div',class_='t-ZTKy').
                                    text.strip():
data.append({
'Rating': rating
'Review Summary': summary,
'Full_Review': full_review
})
return data.
# main scraping logic
base_url = "https://www.flipkart.com"
product_review_url = "/apple-iphone-11-black-64-gb
    itme5041ba101pdf&pid=MOBFWQ6BxGJCEY
    &lid=LSTMOBFWQ6BxGJCEYxxx
    &sortOrder=MOST_HELPFUL&marketplace"
reviews_data = []
# loop through multiple pages until we have data for
    100 reviews.
page_count = 1
while len(reviews_data) < 100:
current_url = base_url + product_review_url +
    "&page=" + str(page_count)
current_data = scrape_flipkart_reviews(current_url,
    page_count)
```

```
if not (current == data:
    break
reviews = data.extend(current_data)
page_count += 1
# Create a DataFrame
df = pd.DataFrame(reviews, data[:100])
# Display the DataFrame
print(df)
```

This code navigates through the pages by incrementing the page_count and concatenates the data until we have information for 100 reviews. Please note that the website structure may change. So you may need to inspect the HTML of the website and adjust the code accordingly if there are any changes.

PAGE No.
DATE / /

Q6 Scrape data for first 100 sneakers you find when you visit flipkart.com and search for sneakers in the search field
You have to scrape 3 attributes of each sneakers:
Brand (2) Product Description (3) price.

⇒ To scrape data for the first 100 sneakers from flipkart you can the request library to send HTTP request to the website Beautiful soup for parsing the HTML content and pandas to create a data frame for the scraped data. Here's sample python program:

```
python                                          Copy code
import requests
from bs4 import BeautifulSoup
import pandas as pd

# function to scrape sneakers data from a given
flipkart search page
def scrape_flipkart_sneakers(url, count):
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Scraping sneakers data from page {url}")
    else:
        print(f"Failed to fetch the webpage, (code
            status code: {response.status_code})
        return None
    soup = BeautifulSoup(response.text, 'html.parser')
    sneakers_listings = soup.find_all('div', class_='_1xHxb7')
```

data = []
for sneakers in sneakers - listings:
Brand = sneakers.find('div', class_='_2WkVRV').text.strip()
description = sneakers.find('a', class_='IRpwTa').text.strip()
price = sneakers.find('div', class_='30jeq3').text.strip()
data.append({
'Brand': brand
'Product Description': description,
'Price': price.
})
Return data.
# main scraping logic
base_url = "https://www.flipkart.com"
search_url = "search?q=sneakers"
sneakers_data = [].
# loop through multiple page until we have data for
100 sneakers.
page_count = 1
while len(sneakers_data) <= 100:
current_url = base_url + search_url + "&page=" + str
(page_count)
current_data = scrape_flipkart_sneakers_url(page_count)
if not current_data:
break.
sneakers_data.extend(current_data)
page_count += 1
# create a Data frame.
df = pd.DataFrame(sneakers_data[:100])
# display the Data frame
print(df)

PAGE No.
DATE / /

Q7 Go to webpage https://www.amazon.in/ Enter "Laptop" in the search filed and the click the search icon. Then.
Set CPU type filter to intel core i7 as shown in the below image. After setting the filters scrape first 10 laptops data. You have to scrape 3 attributes for each laptop.
(1) Title (2) Ratings (3) Price.

⇒ To scrape data for the first 10 laptops with the specified filters from Amazon.in you can use the request library to send HTTP request to the website. Beautiful for parsing the HTML content and pandas to created data frame for the scraped data. Here's sample python program

python

copy code

```
import Requests
from bs4 import BeautifulSoup
import pandas as pd
# function to scrape laptop data from a given
Amazon search page with filters.
def Scrape-amazon-laptops (url):
    response = request.get(url)
    if response.status_code == 200:
        print ("Successfully fetched the webpage.")
    else
        print ( f"Failed to fetch the webpage status
        code. {response.status_code}")
    return None
```

```
soup = BeautifulSoup(response.text, 'html.parser')
laptop_listings = soup.find_all('div', class_='s-inner
                    (content-margin')
data = []
for laptop in laptop_listings[:10]:
title = laptop.find('span', class_='text-normal')
                    text.strip()
ratings = laptop.find('span', class_='a-icon-alt')
ratings = ratings.text if ratings else "Not available
price = laptop.find('span', class_='a offscreen')
                    text.strip()
data.append({
    'Title': title,
    'Ratings': ratings,
    'price': price,
})
return data
# main scraping logic
base_url = "https://www.amazon.in"
search_url = "/s?k=laptop"
clp_filter = "&rh=n%3A1375424031%2Cp_n_feature
-thirteen-browse-bin%3A12598163031"
amazon_url = base_url + search_url + clp_filter
laptops_data = scrape_amazon_laptops(amazon_
                    url)
# create a Data frame
df = pd.DataFrame(laptops_data)
# Display the data frame
print(df)
```

This code performs a search for laptops with the specified filters and extracts data for the first 10 laptops, including the title, rating and price. Keep in mind that the website structure may change, so you may need to inspect the HTML of the website and adjust the code accordingly if there are any changes.

Q8: write a python program to scrape data for Top 1000 quotes of all time.

The above task will be done in following steps

1 First get the webpage https://www.azquotes.com
2 Click on Top Quotes
3 The scrap (a) quote (b) Author (c) Type of quotes.

⇒ To scrape data for the top 1000 quotes from the AZ Quotes website, you can use the 'requests' library to send HTTP requests to the website, Beautiful Soup for parsing the HTML content, and 'pandas' to create a data frame for the scraped data. Please note that web scraping should be done in compliance with the website's terms of service, and it's always a good practice to check if the website allows automated access.

Here's a sample Python program to get you started:

PAGE No.
DATE / /

python

Copy code

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
# Function to scrape data for the top 1000 quotes
def scrape_azquotes_top_quotes(url):
    response = requests.get(url)
    if response.status_code == 200:
        print("Successfully fetched the webpage.")
    else:
        print(f"Failed to fetch the webpage. Status code: {response.status_code}")
        return None
    soup = BeautifulSoup(response.text, 'html.parser')
    quotes_listings = soup.find_all('div', class_='wrap-block')
    data = []
    for quote in quotes_listings:
        quote_text = quote.find('a', class_='title').text.strip()
        author = quote.find('div', class_='author').text.strip()
        quote_type = quote.find('div', class_='kw-box').text.strip()
        data.append({
            'Quote': quote_text,
            'Author': author,
            'Type of Quote': quote_type
        })
    return data
# Main scraping logic
base_url = "https://www.azquotes.com/"
top_quotes_url = "https://www.azquotes.com/top_quotes"
```

az_quotes_url = base_url + top_quotes_url
quotes_data = scrape_az_quotes_top_quotes(az_quotes_url)

# Create a Data Frame
df = pd.DataFrame(quotes_data)
# Display the Data Frame
print(df)

This code performs the necessary step to scrape data for the top 1000 quotes from the AZQuotes website, including the quote text, author, and type of quote. Remember to check the website's terms of service and adjust the code if there are any changes in the website structure.

PAGE No.
DATE / /

Q9 write a python program to display list of respected former prime ministers of India. (i.e. Name, Born-Dod) Term of office, Remarks) from https://www.jagranjosh.com/.

This task will be done in following steps.

1. First get the webpage https://www.jagranjosh.com/
2. Then you have to click on the GK option
3. Then click on the list of all prime Ministers of India
4. Then scrap the mentioned data and make the dataframe.

⇒ unfortunately, I cannot interact with websites or perform web scraping in real-time. However, I can provide you with a python code snippet that you can use as a starting point for web scraping the data you need. please note that web scraping should be done in compliance with the website's terms of service.

Copy code.

```
python
import requests
from bs4 import BeautifulSoup
import pandas as pd
# function to scrape data for former prime
Ministers of India
def scrape_prime_ministers_data(url):
    response = requests.get(url)
if response.status_code == 200:
    print("Successfully fetched the webpage.")
else:
```

print (f"Failed to fetch the webpage. Status code
& response.status_code}")
return none
soup = BeautifulSoup(response.text, 'html.parser')
# Assuming the structure of the HTML on the
page, you may need to inspect the actual HTML
prime_ministers_table = soup.find('table', class_=
'tableizer-table')
data = []
for row in prime_ministers_table.find_all('tr')
[1:]: # skipping the header row
columns = row.find_all('td')
name = columns[0].text.strip()
born_dead = columns[1].text.strip()
term_of_office = columns[2].text.strip()
remarks = columns[3].text.strip()
data.append({
'Name' : name,
'Born-Dead' : born_dead,
'Term of Office' : term_of_office,
'Remarks' : remarks
})
return data
# main scraping logic
base_url = "https://www.jagranjosh.com/"
gk_option_url = "gk/list-of-prime-ministers-of
-india-1440236026-1"
Jagranjosh_url = base_url + gk_option_url

```
prime_ministers_data = scrape_prime_ministers_
                       data('jagdanjosh.xlx')
# create a Data Frme
df = pd.DataFrme(prime_ministers_data)
# Display the Data Frme
print(df)
```

This code assumes a certain structure of the HTML on the page. You may need to inspect the actual HTML of the website to ensure that the class names and structure match. Additionally, be aware that web scraping can be sensitive to changes in the website structure. Adjust the code accordingly if there are any changes to the website.

PAGE No.
DATE / /

Q 10 write a python program to display of 50 most expensive cars in the world (i.e car name and price) from https://www.motor1.com/

This task will be done in following steps:

1. First get the web page https://www.motor1.com/
2. Then you have to type in the search bar 50 most expensive cars'
3. Then click on 50 expesive cars in the world
4. Then Scrap the mentioned data and make the dataframe

=> To achieve this task, you can use the 'request' libary to send HTTP requests to the website, Beautiful Soup' for parsing the HTML content, and pandas to create a data frame for the scraped data. Here' a sample python program.

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
# Function to scrape data for the 50 most expensive cars from motor1
def scrape_expensive_cars_data(url):
    response = requests.get(url)
    if response.status_code == 200:
        print("Successfully fetched the webpage.")
    else:
        print("Failed to fetch the webpage. Status code:", response.status_code)
```

```
return none
soup = BeautifulSoup(response.text, 'html.parser')
# Assuming the structure of the HTML on the page (you may need to inspect the actual HTML
cars_listings = soup.find_all('div', class_='article-widget--list-item')
data = []
for car in cars_listings[:50]:
car_name = (car.find('a', class_='article-widget--list-item-title').text.strip()
car_price = car.find('div', class_='article-widget--list-item-details').text.strip()
data.append({
'Car Name': car_name,
'Price': car_price
})
# main scraping logic
base_url = "https://www.motor1.com/"
search_query = "50 most expensive cars"
search_url = (f"search/?q={search_query.replace(' ', '+')}")
motor1_url = base_url + search_url
expensive_cars_data = scrape_expensive_cars(data.motor1_url)
# create a DataFrame
df = pd.DataFrame(expensive_cars_data)
# display the DataFrame
print(df)
```