Python Assignment

Web Scraping

Harshal Rajput

8826639815

harshalrajput9000@gmail.com

I have divided the assignment into two parts which I thought would be efficient and better in terms of readability and understandability.

One part focuses on overall assignment structure whereas the other part is focused on retrieving data from product list page.

1. Product_Details.py

```
# importing libraries
from bs4 import BeautifulSoup
import requests


def main(URL):
    # opening our output file in append mode
    File = open("out.csv", "a")

    # specifying user agent, You can use other user agents
    # available on the internet
    HEADERS = ({'User-Agent':'Mozilla/5.0 (X11Linux
x86_64)AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/44.0.2403.157 Safari/537.36', 'Accept-Language': 'en-US,
en;q=0.5'})

    # Making the HTTP Request
    webpage = requests.get(URL, headers=HEADERS)

    # Creating the Soup Object containing all data
    soup = BeautifulSoup(webpage.content, "lxml")
```

```
    # retrieving product url
    try:
        a = soup.find('a', attrs={"class": 'a-size-base a-link-
normal s-no-hover s-underline-text s-underline-link-text s-link-
style a-text-normal'})
        if(a):
            url = a['href']
        else:
            url = "NA"
    except AttributeError:
        url = "NA"
    print(f"Product Url = https://www.amazon.in{url}")
    File.write(f"{url},")

    # retrieving product name
    try:
        name = soup.find("span", attrs={"class": 'a-size-medium
a-color-base a-text-normal'}).string.strip().replace(',', '')
    except AttributeError:
        name = "NA"
    print("Product Name = ", name)
    File.write(f"{name},")

    # retrieving product price
    try:
        a = soup.find('a', attrs={"class": 'a-size-base a-link-
normal s-no-hover s-underline-text s-underline-link-text s-link-
style a-text-normal'})
        if(a):
            url = a['href']
        if(url):
            price = a.find("span", attrs={"class": 'a-price-
whole'}).text
        else:
            price = soup.find("span", attrs={"class": 'a-price-
whole'}).text
    except AttributeError:
        price = "NA"
    print("Products Price = ", price)
    File.write(f"{price},")

    # retrieving product rating
```

```
    try:
        b = soup.find('div', attrs={"class": 'a-row a-size-
small'})
        if(b):
            rating = b.find("span", attrs={"class": 'a-size-base
puis-normal-weight-text'}).text
        else:
            rating = soup.find("span", attrs={"class": 'a-size-
base puis-normal-weight-text'}).text
    except AttributeError:
        rating = "NA"
    print("Products Rating = ", rating)
    File.write(f"{rating},")

    # retrieving product reviews
    try:
        reviews = soup.find("span", attrs={"class": 'a-size-base
s-underline-text'}).text
    except AttributeError:
        reviews = "NA"
    print("Products Reviews = ", reviews)
    File.write(f"{reviews},")

    # closing the file
    File.close()


if __name__ == '__main__':
    main('https://www.amazon.in/s?k=bags&page=3&crid=2M096C61O4ML
T&qid=1694448990&sprefix=ba%2Caps%2C283&ref=sr_pg_3')
```

2. Overall Structure Code:

```
import csv
import requests
from bs4 import BeautifulSoup
import time

#Scraping Product Details from Product Listing Page
def scrape_product_listing_page(url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (X11Linux
x86_64)AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/44.0.2403.157 Safari/537.36', 'Accept-Language': 'en-US,
en;q=0.5',
    }
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        product_details = []

        # Extract product information from the page (e.g.,
product URL, name, price, rating, number of reviews)

        #This would be done using Product_Details.py

        #This way is for example to showcase further steps
        a = soup.find('a', attrs={"class": 'a-size-base a-link-
normal s-no-hover s-underline-text s-underline-link-text s-link-
style a-text-normal'})
        product_urls = a['href']
        product_names = soup.find("span", attrs={"class": 'a-
size-medium a-color-base a-text-
normal'}).string.strip().replace(',', '')
        product_prices = a.find("span", attrs={"class": 'a-price-
whole'}).text
        product_ratings = soup.find("span", attrs={"class": 'a-
size-base puis-normal-weight-text'}).text
```

```
        product_reviews = soup.find("span", attrs={"class": 'a-
size-base s-underline-text'}).text

        for i in range(len(product_urls)):
            product_details.append({
                'Product URL': product_urls[i],
                'Product Name': product_names[i],
                'Product Price': product_prices[i],
                'Rating': product_ratings[i],
                'Number of Reviews': product_reviews[i]
            })

        return product_details

    else:
        print(f"Failed to fetch data from URL: {url}")
        return []

# Scraping Additional Details from Product Page
def scrape_product_page(product_url):
    headers = {
        'User-Agent': 'Mozilla/5.0 (X11Linux
x86_64)AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/44.0.2403.157 Safari/537.36', 'Accept-Language': 'en-US,
en;q=0.5',
    }
    response = requests.get(product_url, headers=headers)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        # Extracting additional product information (description,
ASIN, product description, manufacturer)

        product_description =
soup.select_one('#productDescription').get_text(strip=True)
        asin = soup.find('th',
text='ASIN').find_next('td').text.strip()
        manufacturer = soup.find('th',
text='Manufacturer').find_next('td').text.strip()

        return {
            'Description': product_description,
```

```
            'ASIN': asin,
            'Product Description': product_description,
            'Manufacturer': manufacturer
        }
    else:
        print(f"Failed to fetch data from URL: {product_url}")
        return {}

# Main function
def main():
    base_url =
"https://www.amazon.in/s?k=bags&crid=2M096C61O4MLT&qid=1653308124
&sprefix=ba%2Caps%2C283&ref=sr_pg_"

    all_product_details = []

    # Scraping 20 pages of product listing
    for page_number in range(1, 21):
        url = f"{base_url}{page_number}"
        product_details = scrape_product_listing_page(url)
        all_product_details.extend(product_details)

        # Adding a delay to avoid overloading Amazon's servers
        time.sleep(2)

    # Scraping additional product details from individual product
pages
    for product_detail in all_product_details[:200]:
        product_url = product_detail['Product URL']
        additional_details = scrape_product_page(product_url)
        product_detail.update(additional_details)

        # Adding a delay to avoid overloading Amazon's servers
        time.sleep(2)

    # Writing data to a CSV file
    with open('output.csv', 'w', newline='', encoding='utf-8') as
csvfile:
        fieldnames = ['Product URL', 'Product Name', 'Product
Price', 'Rating', 'Number of Reviews', 'Description', 'ASIN',
'Product Description', 'Manufacturer']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
```

```
        writer.writeheader()
        for product_detail in all_product_details[:200]:
            writer.writerow(product_detail)

if __name__ == "__main__":
    main()
```

I have read and learned from many sources present there and have done the assignment the best way I could have done.

There are some issues which I would like to mention:

- Firstly, the link for the amazon website takes few number of tries before we could actually scrape data from it.
- There is some sort of security terms the site has applied in the URL to avoid scraping, thus the link pretends to be changing in a different manner for each new page.
- Even after using all thr right selectors some of the things may not work at once but could work on the other turn.
- The program needs a few tries before it could actually load the result.

I have tried to solve each problem with atleast 2 different solutions so as to check for inconsisitency in results and found the same for each one of them.

Therefore few things may not seem to work at first but they are conceptually correct as per my knowledge and just the issue is with optimisation and other things.

I have learned a lot while working on this assignment and have gained decent knowledge regarding the same.

Although I tried a lot to go through this task as far as I could, but I feel the time wasn't enough but still I tried my best to provide the best possible solution.

Looking forward to learn many new things further and work with them.