Web Scraping Amazon Product Data

Project Overview

This project demonstrates the use of Python to scrape product information from an Amazon webpage. The goal of this project was to automate the process of gathering product data, including product names, prices, reviews, ratings, quantities, and original prices. The scraped data is saved in a CSV format, which can be used for further analysis or reporting.

The script is designed to process a locally saved HTML file containing Amazon product listings and extract specific product details from the page. This approach eliminates the need to manually gather the data, enabling users to quickly compile and analyze large sets of product data.

Technologies Used

- **Python**: The primary programming language used for the script.
- **Libraries**:
  - **BeautifulSoup**: A library for parsing HTML and XML documents. Used to extract specific data from the webpage's HTML structure.
  - **Pandas**: A powerful data manipulation and analysis library, used here to organize the extracted data and save it as a CSV file.
- **Platform**:
  - **Jupyter Notebook**: The script was executed on Jupyter Notebook, which is ideal for data analysis and testing code in an interactive environment.

Features

- Extracts multiple key details from product listings, including:
  - **Product Name**
  - **Product Price**
  - **Product Reviews**
  - **Product Ratings**
  - **Quantity for Past Month**
  - **Original Price**
- Stores the extracted data into a CSV file for easy further use and analysis.
- Handles missing data gracefully by assigning default values when necessary.

Installation

Prerequisites

- Python 3.x (You can download it from)
- You will need to install the following Python libraries:
    - **BeautifulSoup**
    - **Pandas**

Installing Dependencies

You can install the required libraries using `pip`. Open a terminal or command prompt and run the following commands:

```
pip install beautifulsoup4
pip install pandas
```

Downloading the HTML File

For this project, the HTML file was manually downloaded from the Amazon product page. Here are the steps to manually download the HTML file:

- Open the Amazon product listing page in your browser.
- Right-click on the page and select **"Save As"**.
- Choose **"Web Page, HTML only"** as the file type and save the file to your local machine.

Ensure that the file path matches the one you reference in your script.

Usage

1. **Set up the Environment**:
   Install the required libraries (BeautifulSoup and Pandas) by running `pip install beautifulsoup4 pandas` in your terminal or command prompt.
2. **Download the HTML File**:
   Save the HTML content of the Amazon product page that you want to scrape. Ensure the file path is correct when specifying it in the script.
3. **Run the Script in Jupyter Notebook**:
   The script is designed to read the local HTML file, extract the product information, and save it into a CSV file. To run the script:
    - Open a new Jupyter Notebook and paste the Python code into a cell.
    - Update the file path to your local HTML file.
    - Run the cell to execute the script.
4. **Output**:
   After running the script, you will have a CSV file (`Amazon_volleyball40.csv`) containing the extracted data in the specified directory.

Code Explanation

Importing Libraries

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

5. **BeautifulSoup** is used for parsing and navigating the HTML.
6. **Pandas** is used for creating the DataFrame and saving the data into CSV.

Reading and Parsing the HTML File

```
with open(r"D:\DATA ANALYTICS\python 2\Amazon_volleyball40.html", 'r',
encoding='utf8') as file:
    soup = BeautifulSoup(file, 'html.parser')
```

- This part reads the HTML file and parses it using BeautifulSoup.

Extracting Data for Each Product

```
# Find all product containers
divs = soup.find_all('div', class_="sg-col-inner")
print(f"Found {len(divs)} product divs")  # Debugging line

# Extract details for each product
for div in divs:
    # Product Name
    name_element = div.find('span', class_="a-size-base-plus a-color-base a-text-
normal")
    product_name = name_element.get_text(strip=True) if name_element else " "
    product_names.append(product_name)
    print(f"Product Name: {product_name}")  # Debugging line

    # Product Price
    price_element = div.find('span', class_="a-price-whole")
    product_price = price_element.get_text(strip=True) if price_element else " "
    product_prices.append(product_price)
    print(f"Product Price: {product_price}")  # Debugging line

    # Product Reviews
    review_element = div.find('span', class_="a-size-base")
    product_reviews_text = review_element.get_text(strip=True) if review_element
else " "
    product_reviews.append(product_reviews_text)
    print(f"Product Reviews: {product_reviews_text}")  # Debugging line

    # Product Quantity
    quantity_element = div.find('span', class_="a-size-base a-color-secondary")
    product_quantity_text = quantity_element.get_text(strip=True) if
quantity_element else " "
    quantity_for_pastmonth.append(product_quantity_text)
    print(f"Product Quantity: {product_quantity_text}")  # Debugging line
```

```
    # Product Ratings
    rating_element = div.find('span', class_="a-icon-alt")
    product_rating_text = rating_element.get_text(strip=True) if rating_element
else " "
    ratings.append(product_rating_text)
    print(f"Product Ratings: {product_rating_text}")   # Debugging line

    # Product Original Price
    originalprice = div.find_all('span', class_="a-offscreen")
    originalprice_element = originalprice[1] if len(originalprice) > 1 else None
    product_original_text = originalprice_element.get_text(strip=True) if
originalprice_element else " "
    original_price.append(product_original_text)
    print(f"Product Original_price: {product_original_text}") # Debugging line
```

- **Div Selection:** `divs = soup.find_all('div', class_="sg-col-inner")` looks for all product containers on the webpage.
- **Product Details:** For each product, the script extracts the name, price, reviews, quantity, ratings, and original price. The `get_text(strip=True)` method is used to clean the extracted text.
- **Appends Data:** Each detail is appended to the corresponding list: `product_names`, `product_prices`, etc.
- **Print Debugging:** Debugging lines such as `print(f"Product Name: {product_name}")` are used to print each product's details during the extraction.

Saving Data to CSV

```
# Create a DataFrame and save to CSV
data = {
    'Product_Name': product_names,
    'Product_Price': product_prices,
    'Product_Reviews': product_reviews,
    'Product_Quantity': quantity_for_pastmonth,
    'Product_Ratings': ratings,
    'Original_Price': original_price
}

df = pd.DataFrame(data)
output_path = r'D:\DATA ANALYTICS\Amazon files2\Amazon_volleyball40.csv'
df.to_csv(output_path, index=False)
```

- The extracted data is stored in a Pandas DataFrame and then saved as a CSV file.

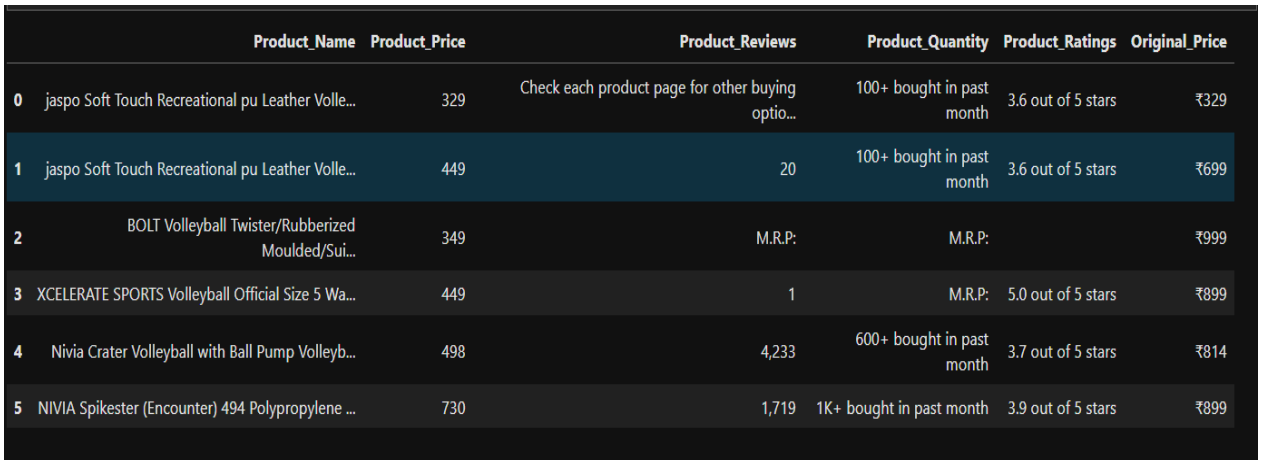
Challenges Faced

- **Handling Missing Data**:
  During the scraping process, some products may not have all the data points available. For instance, not all products may have reviews, ratings, or original prices listed. The script handles this by checking for the presence of each data element and assigning a default value (empty string) if the data is missing.

- **Dynamic Page Structure**:
  Websites like Amazon often update their page structures or HTML tags. In such cases, the scraping code would need to be adjusted to account for any changes in the structure or class names.

Example Output

Here is a sample of the data extracted by the script:



	Product_Name	Product_Price	Product_Reviews	Product_Quantity	Product_Ratings	Original_Price
0	jaspo Soft Touch Recreational pu Leather Volle...	329	Check each product page for other buying optio...	100+ bought in past month	3.6 out of 5 stars	₹329
1	jaspo Soft Touch Recreational pu Leather Volle...	449	20	100+ bought in past month	3.6 out of 5 stars	₹699
2	BOLT Volleyball Twister/Rubberized Moulded/Sui...	349	M.R.P:	M.R.P:		₹999
3	XCELERATE SPORTS Volleyball Official Size 5 Wa...	449	1	M.R.P:	5.0 out of 5 stars	₹899
4	Nivia Crater Volleyball with Ball Pump Volleyb...	498	4,233	600+ bought in past month	3.7 out of 5 stars	₹814
5	NIVIA Spikester (Encounter) 494 Polypropylene ...	730	1,719	1K+ bought in past month	3.9 out of 5 stars	₹899

Conclusion

This web scraping project allows for automated extraction of product data from Amazon using Jupyter Notebook. By leveraging Python libraries such as BeautifulSoup and Pandas, the process is simplified, enabling the extraction and storage of large volumes of data for analysis.