Amazon Web Scraping with BeautifulSoup

This Python script scrapes product data from Amazon's search results using BeautifulSoup and saves it to a CSV file. The script allows you to specify a user agent to mimic a web browser.

Prerequisites

Before running the script, make sure you have the necessary libraries installed:

- `requests`: To make HTTP requests.
- `beautifulsoup4`: To parse the HTML content.
- `pandas`: To create and manipulate data in tabular form.
- `fake_useragent`: To generate user-agent strings for HTTP headers.

You can install these libraries using `pip`:

```
pip install requests beautifulsoup4 pandas fake_useragent
```

Usage (Single-Page Scraping)

1. Clone or download this repository.
2. Open the Python script, `amazon_web_scraping.py`, in your preferred code editor.
3. Modify the `url` variable to the Amazon search results page you want to scrape.
4. Run the script using Python.

The script will scrape product data and save it to a CSV file named `amazon_products_headphone.csv`.

Code Snippets (Single-Page Scraping)

Sending an HTTP GET Request

```
import requests

url = "https://www.amazon.in/s?rh=n%3A1388921031&fs=true&ref=lp_1388921031_sar"
ua = UserAgent()
headers = {'User-Agent': ua.chrome}
response = requests.get(url, headers=headers)
```

Parsing HTML with BeautifulSoup

```
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(response.text, "html.parser")
results = soup.find("div", {"class": "s-main-slot s-result-list s-search-results
sg-row"})
```

Extracting Product Data

```
all_products = results.find_all("div", {"class": "sg-col-inner"})

for item in all_products:
    name = item.find("span", {"class": "a-size-base-plus"})
    price = item.find("span", {"class": "a-price-whole"})

    name_text = name.text.strip if name else "N/A"
    price_text = price.text.strip if price else "N/A"
```

Saving Data to CSV

```
import pandas as pd

# Create a DataFrame using Pandas
df = pd.DataFrame(product_data, columns=['Name', 'Price'])

# Save the DataFrame to a CSV file
df.to_csv('amazon_products_headphones.csv', index=False, encoding='utf-8')
```

Usage (Multipage Scraping)

1. Clone or download this repository.
2. Open the Python script, `amazon_multipage_web_scraping.py`, in your preferred code editor.
3. Modify the `url` variable to the Amazon search results page you want to scrape.
4. Adjust the `num_pages` variable to specify the number of pages you want to scrape.
5. Run the script using Python.

The script will scrape product data from the specified number of pages and save it to a CSV file named `amazon_products_headphones_multi.csv`.

Code Explanation (Multipage Scraping)

Sending HTTP GET Requests for Multiple Pages

The script iterates through multiple pages to scrape data by sending HTTP GET requests to each page and parsing the HTML content. Here's how it works:

```
def scrape_amazon_product_data(url, headers, num_pages):
    all_product_data = []

    for page in range(1, num_pages + 1):
        current_url = f"{url}&page={page}"
        response = requests.get(current_url, headers=headers)
        response.raise_for_status()

        soup = BeautifulSoup(response.text, "html.parser")

        # ... (parsing and extracting data)

    return all_product_data
```

Organizing and Saving Data

The scraped data is organized into a Pandas DataFrame, which allows you to manipulate and analyze it easily. The data is then saved to a CSV file as shown here:

```
if product_data:
    # Create a DataFrame using Pandas
    df = pd.DataFrame(product_data, columns=['Name', 'Price'])

    item_count = len(df)  # Get the count of items

    print(f"Total items found: {item_count}")

    # Save the DataFrame to a CSV file
    df.to_csv('amazon_products_headphones_multi.csv', index=False, encoding='utf-
8')

    print("Data has been saved to amazon_products_headphones_multi.csv")
else:
    print("No results found on the pages.")
```