Project Report – BookScraper

Project Overview

BookScraper is a Python-based web scraping project designed to collect data from BooksToScrape.com, a sandbox website for practicing web scraping. The tool navigates all available pages and extracts structured book information into a CSV file, which can be used for data analysis, machine learning, or cataloging tasks.

Link

www.github.com/stevenpyae/BookScraper

Technology Stack

Component	Description
Python 3.x	Core programming language
`requests`	(optional) Alternative for HTTP GET
`BeautifulSoup4`	HTML parsing and DOM navigation
`csv`	Writing structured data to CSV files
`re`	Regex for data sanitization

Design & Approach

1. **URL Navigation**
   The scraper starts at the homepage and automatically follows pagination links until the last page.
2. **HTML Parsing**
   Each page is parsed using BeautifulSoup. Book data is located within `<article class="product_pod">` blocks.
3. **Data Extraction**
   For each book, the following details are extracted:
   - **Title** (cleaned of commas and quotes)
   - **Price** (converted to float, GBP)
   - **Rating** (parsed from class name, mapped to integer)
   - **Availability** (cleaned string)
4. **Data Cleaning**
   Special characters like commas double quotes and Pound price are removed to ensure CSV compatibility and consistency.
5. **Data Storage**
   All records are written to a single file: `output/books.csv`

✅ Output

- Over **400 book records** extracted across multiple pages.
- Data saved to: `output/books.csv`
- Format:

```
title,price_gbp,rating,availability
"Book Title", price, rating, "In stock"
```

🧪 Challenges & Solutions

Challenge	Solution
Commas and quotes breaking CSV formatting	Cleaned titles using `str.replace()` and ensured proper quoting
Currency symbols	Cleaned it using clean_price(price) function and regex to remove the special characters
Star ratings stored in CSS class names	Extracted class and mapped manually (`One` → `1`, etc.)
Pagination without explicit "last" page	Used loop to follow `next` links until no longer present

Key Takeaways

- Knowledge BeautifulSoup for structured scraping
- Handling real-world HTML inconsistencies
- Data pre-processing for CSV output
- Writing modular and reusable Python scraping code

Potential for Improvements

- User defined number of books/pages to retrieve
  Simplicity of the target site made the project relatively easier to implement
- The target site lacked scraping protection such as CloudFlare protection or CAPTCHA
- Writing Test Codes to avoid manually testing the output of the program
- Targeting websites with stronger anti-scraping measures
- Implementing session rotation and user-agent spoofing => Basic code demonstrated in www.github.com/stevenpyae/propertyguru_scraper