**Web Scraping Assignment: National Parks Data**

**Learning Objectives**

By the end of this assignment, students will be able to:

* Use the requests library to fetch web pages
* Parse HTML using Beautiful Soup
* Extract specific data from HTML elements
* Store scraped data in a Pandas DataFrame
* Export data to CSV format

**Assignment Overview**

You will scrape book information from a practice website designed for web scraping education. This will teach you the fundamentals that you can apply to real-world scenarios like scraping national park data.

**Website:** http://books.toscrape.com/

This is a safe, legal website specifically created for practicing web scraping skills.

**Part 1: Basic Scraping (50 points)**

**Task 1.1: Scrape Book Titles (15 points)**

Write a Python script that:

1. Fetches the homepage of books.toscrape.com
2. Extracts ALL book titles from the first page
3. Prints the titles to the console

**Expected Output:** A list of 20 book titles

**Hints:**

* Book titles are in <h3> tags inside <a> tags
* Use .find\_all() to get all matching elements
* Use .text or .get\_text() to extract the text content

**Task 1.2: Scrape Book Prices (15 points)**

Extend your script to also extract:

* Book prices

**Hints:**

* Prices are in elements with class price\_color
* Use .find() or .find\_all() with the class\_ parameter

**Task 1.3: Scrape Star Ratings (20 points)**

Further extend your script to extract:

* Star ratings (One, Two, Three, Four, or Five stars)

**Hints:**

* Star ratings are in the class attribute of <p> tags
* Look for classes like "star-rating Three"
* You'll need to parse the class attribute

**Part 2: Create a DataFrame (30 points)**

**Task 2.1: Organize Data (20 points)**

Store all scraped data in a Pandas DataFrame with columns:

* Title
* Price
* Rating

Print the first 10 rows using .head(10)

**Task 2.2: Save to CSV (10 points)**

Export your DataFrame to a CSV file named books\_data.csv

**Part 3: Extension Challenge (20 points)**

Choose ONE of the following challenges:

**Option A: Data Cleaning**

* Remove the "£" symbol from prices and convert to float
* Convert ratings from words (e.g., "Three") to numbers (e.g., 3)
* Add a column showing if a book is "Highly Rated" (4-5 stars) or not

**Option B: Multiple Pages**

* Modify your script to scrape the first 3 pages of books
* The website has pagination - figure out the URL pattern
* Combine all data into a single DataFrame

**Option C: Data Analysis**

* Calculate the average price of all books
* Find the most common rating
* Create a simple bar chart showing the distribution of ratings

**Rubric**

| **Component** | **Points** | **Criteria** |
| --- | --- | --- |
| **Part 1: Basic Scraping** | 50 |  |
| - Book titles extracted correctly | 15 | All 20 titles scraped accurately |
| - Book prices extracted correctly | 15 | All prices scraped with currency symbol |
| - Star ratings extracted correctly | 20 | All ratings extracted (One-Five) |
| **Part 2: DataFrame** | 30 |  |
| - Data organized in DataFrame | 20 | Proper column names, all rows included |
| - CSV export working | 10 | File created and readable |
| **Part 3: Extension** | 20 |  |
| - Challenge completed | 20 | One option fully implemented |
| **Code Quality** | Bonus |  |
| - Comments and documentation | +5 | Clear comments explaining each step |
| - Error handling | +5 | try/except blocks used appropriately |

**Total: 100 points (+ 10 bonus)**

**Submission Requirements**

Submit the following:

1. Modularized Python script (.py files)
2. The generated CSV file (books\_data.csv)
3. A short README explaining:
   * Which extension challenge you chose (if any)
   * Any difficulties you encountered
   * What you learned

**Tips for Success**

1. **Inspect the HTML first**: Right-click on the webpage and select "Inspect" to see the HTML structure
2. **Test incrementally**: Get titles working first, then add prices, then ratings
3. **Use print statements**: Print variables to check what you're extracting
4. **Handle errors**: Some books might be missing data - handle these cases
5. **Be polite**: This site doesn't need it, but add time.sleep(1) between requests as good practice

**Advanced: Real National Parks Data**

Once you master this assignment, you can apply these skills to scrape actual National Park data from:

* Individual state pages: https://www.nps.gov/state/XX/list.htm (replace XX with state code)
* Or use the official NPS API: https://www.nps.gov/subjects/developer/

The API is actually the **recommended** approach for real projects!

**Resources**

* [Beautiful Soup Documentation](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)
* [Pandas Documentation](https://pandas.pydata.org/docs/)
* [HTTP Status Codes](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status)