Python Libraries for Data Acquisition – BeautifulSoup, Scrapy, Requests

Lab Objective:

By the end of this lab, students will:

1. Learn how to use **Requests** to send HTTP requests and retrieve web pages.
2. Use **BeautifulSoup** for web scraping to parse HTML and extract specific data.
3. Get an introduction to **Scrapy**, a powerful web scraping framework for handling complex websites.

Prerequisites:

- Install required libraries:
  - **Requests**: `pip install requests`
  - **BeautifulSoup**: `pip install beautifulsoup4`
  - **Scrapy**: `pip install scrapy`

Part 1: Using Requests to Retrieve Web Pages

Objective:

- Learn how to use the `requests` library to fetch a webpage's HTML content.

Task 1: Sending a GET Request

- Fetch the HTML content of a webpage using the Requests library.

Code Example:

```
import requests

# Send a GET request to a webpage
url = "https://example.com"
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    print("Page fetched successfully!")
    print(response.text)  # Prints the HTML content of the page
else:
    print("Failed to retrieve the page.")
```

Exercise:

- Write a script that fetches the HTML content of any webpage (e.g., `https://www.wikipedia.org`) and prints the first 500 characters of the page.

Part 2: Web Scraping with BeautifulSoup

Objective:

- Use BeautifulSoup to parse the HTML content retrieved using Requests and extract specific information from it.

Task 1: Parsing HTML and Extracting Data

- Extract the title of a webpage using BeautifulSoup.

Code Example:

```
from bs4 import BeautifulSoup
import requests

# Send a request to fetch the HTML content of the page
url = "https://example.com"
response = requests.get(url)

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.text, 'html.parser')

# Extract the title of the page
title = soup.title.text
print("Page Title:", title)
```

Task 2: Extracting All Links

- Extract all the hyperlinks (`<a>` tags) from a webpage.

Code Example:

```
# Find all the anchor tags in the page and extract the links
links = soup.find_all('a')

for link in links:
    print(link.get('href'))
```

Exercise:

- Write a script that extracts all the headings (`<h1>`, `<h2>`, etc.) from a webpage and prints them.

Part 3: Introduction to Scrapy (for Advanced Web Scraping)

Objective:

- Understand the basics of Scrapy, a web crawling and scraping framework designed for large-scale scraping projects.

Setup:

- Scrapy is command-line based, so the following steps need to be run in the terminal.

Task 1: Create a New Scrapy Project

1. Open a terminal or command prompt.
2. Create a new Scrapy project by running:

   ```
   scrapy startproject example_project
   ```

3. Navigate into the project folder:

   ```
   cd example_project
   ```

4. Run the Scrapy spider:

   ```
   scrapy crawl example
   ```

Task 2: Creating a Simple Spider

- Create a spider that scrapes the titles of articles from a website.

Code Example (Spider Code):

In the `spiders` folder, create a new Python file (`example_spider.py`) and add the following code:

```
import scrapy

class ExampleSpider(scrapy.Spider):
    name = "example"
    start_urls = ['https://example.com']

    def parse(self, response):
        page_title = response.css('title::text').get()
        print("Page Title:", page_title)

        # Extracting all links on the page
        links = response.css('a::attr(href)').getall()
        for link in links:
            print(link)
```

Task 3: Running the Spider

- Run the Scrapy spider from the terminal to see the extracted data:

```
scrapy crawl example
```

Lab Exercises:

1. **Using Requests and BeautifulSoup:**
   - Write a Python script that uses **Requests** and **BeautifulSoup** to extract and print:
     - The title of the page.
     - All the `<img>` tags (images) on the page.
     - All the paragraphs (`<p>`) from a webpage.
2. **Using Scrapy:**
   - Create a Scrapy spider that crawls a news website (e.g., `https://news.ycombinator.com/`) and extracts the titles of the top 10 articles.