**Memoriam**

**Objective**

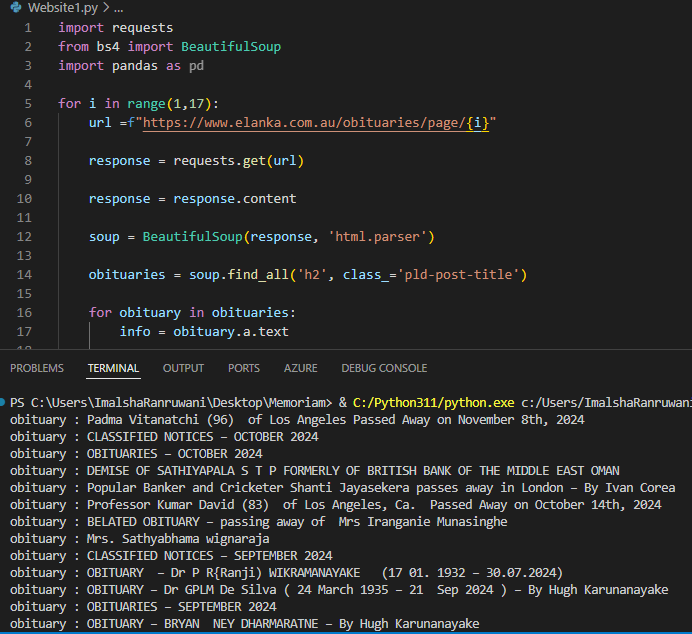
Web scraping - retrieve data from different targeted websites to a centralized location

(obituary notices - historical data) --- write a script to retrieve data

**Web Scraping with Python + Beautiful Soup library**

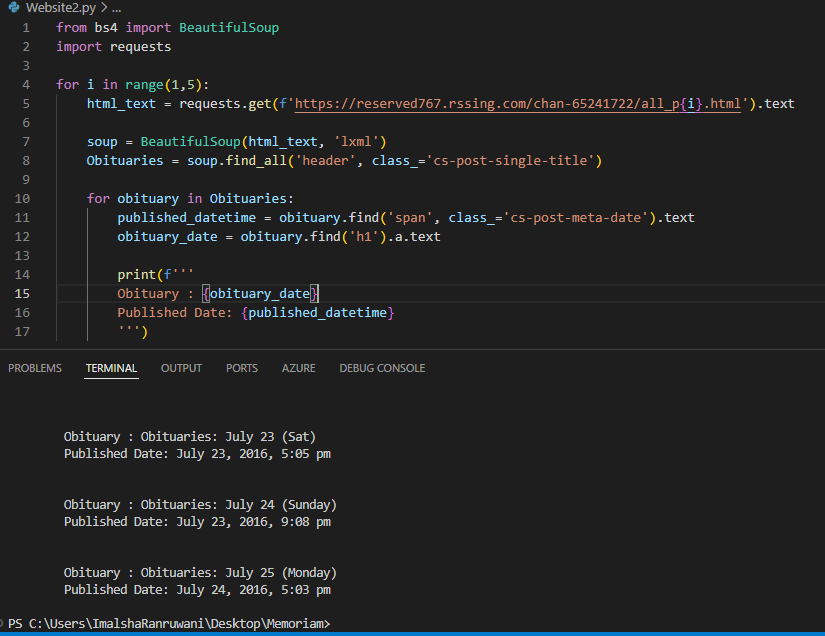
1. Australian - <https://www.elanka.com.au/obituaries/>

Time: around 1 minute



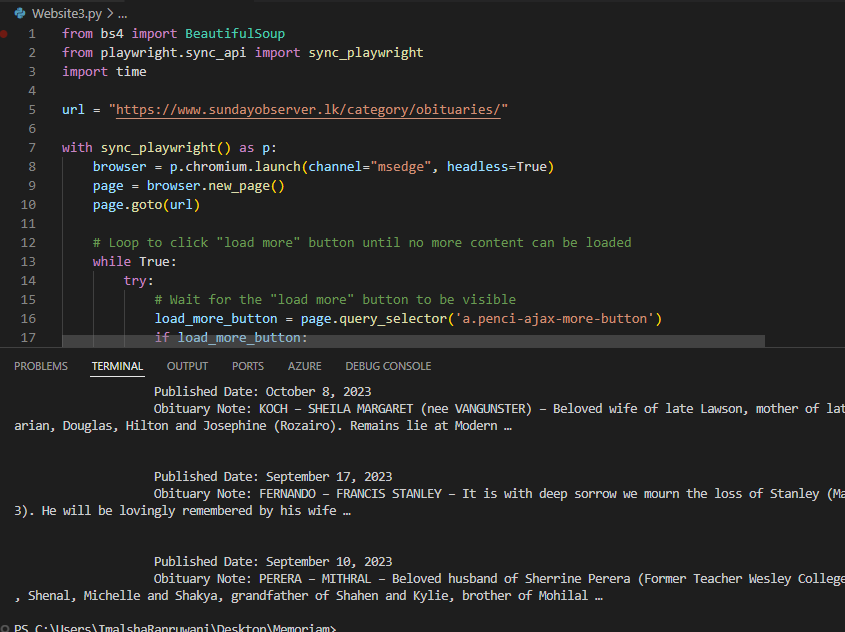
2 - Canada (Times of Lanka) - <https://reserved767.rssing.com/chan-65241722/all_p1.html>

Time: around 20 seconds



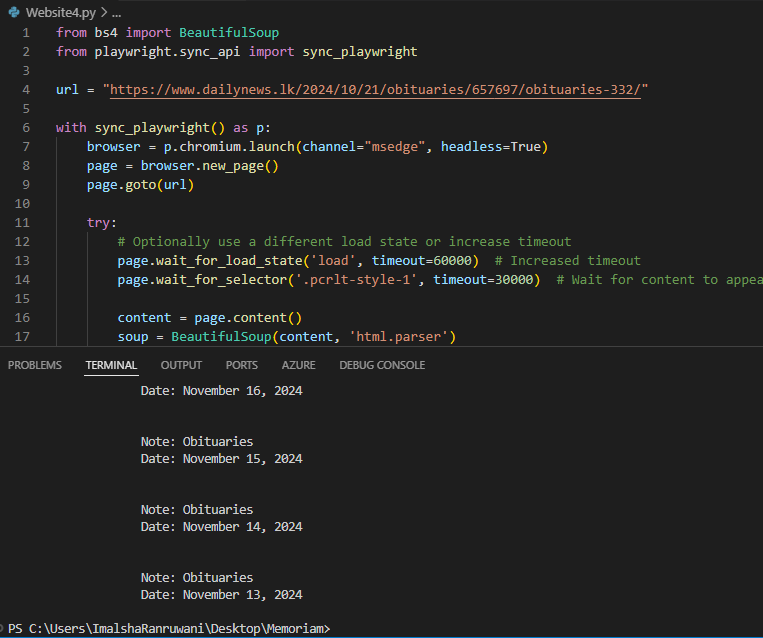
3 - LK (Sunday Times) - <https://www.sundayobserver.lk/category/obituaries/>

Time: around 35 seconds



4 - LK (Daily News Archive) - <https://www.dailynews.lk/2024/10/21/obituaries/657697/obituaries-332/>

Time: around 10 seconds



**Advantages**

1. **Easy to Learn and Use**
   * It requires minimal code to extract elements from an HTML page using CSS selectors, tags, classes, and IDs.
2. **Flexible HTML Parsing**
   * It can handle poorly structured or "broken" HTML pages and convert them into a navigable parse tree, making it robust for web scraping tasks involving unpredictable page structures.
3. **Integration with Other Libraries**
   * Beautiful Soup can be combined with other libraries like requests (for fetching web content) and pandas (for data manipulation), providing a powerful data extraction and processing workflow.
4. **Efficient for Static Pages**
   * Ideal for extracting data from static web pages where content doesn’t change dynamically after initial loading, avoiding the overhead of dealing with JavaScript-heavy content.
5. **Open Source and Community Support**
   * Beautiful Soup is free, open-source, and has a strong community of users and contributors who provide extensive documentation, examples, and support.

**Disadvantages**

1. **Limited Handling of JavaScript Content**
   * Beautiful Soup does not support JavaScript execution, making it ineffective for scraping dynamically loaded content (e.g., pages that use AJAX calls). This may require using tools like Selenium, Playwright, or Scrapy for JavaScript-heavy sites.
2. **Slower Performance for Large-scale Scraping**
   * Beautiful Soup is slower compared to more advanced libraries like lxml or frameworks like Scrapy when handling large datasets or performing concurrent scraping tasks.
   * It works best for small to medium-sized tasks but may not scale as effectively as more comprehensive solutions.
3. **Requires Additional Libraries for HTTP Requests**:
   * Beautiful Soup is purely an HTML/XML parsing library and does not handle HTTP requests directly. You'll need to use requests or similar libraries to fetch content from the web.
4. **Potential for Website Blocking**:
   * Like all web scraping methods, frequent or large-scale scraping attempts can result in being blocked by websites that detect and prevent automated access, requiring careful rate-limiting, user-agent rotation, and respect for robots.txt directives.
5. **Manual Handling of Complex Structures**:
   * While Beautiful Soup makes HTML parsing easier, complex web pages with nested structures may require detailed manual handling and inspection of the DOM, which can be time-consuming.
6. **Compliance and Legal Risks**:
   * Web scraping using Beautiful Soup does not inherently comply with website terms of service or legal regulations (e.g., terms prohibiting scraping). Careful attention is needed to ensure ethical and lawful scraping practices.