ENSF-381: Full Stack Web Development Laboratory

Ahmad Abdellatif and Novarun Deb
Department of Electrical & Software Engineering
University of Calgary
Lab 8

Objectives

Welcome to the ENSF381 course lab! In this lab, we will create a Python tool that analyzes the structure and content of web pages. Along the way, we will revisit some key Python basics while working with BeautifulSoup, a powerful library for web scraping. You will get to extract text, find common words, and even visualize the data you gather.

Groups

Lab instructions must be followed in groups **of two students**.

Submission

You must submit the complete source code, ensuring it can be executed without any modifications. Also, if requested by the instructor, you may need to submit the corresponding documentation file (e.g., word and image). Only one member of the group needs to submit the assignment, but the submission must include the names and UCIDs of all group members at the top of the code.

Deadline

Lab exercises must be submitted by **11:55 PM on the same day as the lab session**. Submissions made within 24 hours after the deadline will receive a maximum of 50% of the mark. Submissions made beyond 24 hours will not be evaluated and will receive a grade of zero.

Academic Misconduct

Academic Misconduct refers to student behavior which compromises proper assessment of a student's academic activities and includes: cheating; fabrication; falsification; plagiarism; unauthorized assistance; failure to comply with an instructor's expectations regarding conduct required of students completing academic assessments in their courses; and failure to comply with exam regulations applied by the Registrar.

For more information on the University of Calgary Student Academic Misconduct Policy and Procedure and the SSE Academic Misconduct Operating Standard,

please visit: https://schulich.ucalgary.ca/current-students/undergraduate/student-resources/policies-and-procedures

Introduction to BeautifulSoup

BeautifulSoup is a Python library used for parsing HTML and XML documents. It creates a parse tree for the parsed web page, making it easy to extract data from HTML tags like <h1>, <a>, and <p>. It's widely used in web scraping projects.

BeautifulSoup is a powerful and versatile library used for parsing HTML due to its many advantages. One of its key benefits is **its simple syntax**, which makes it easy to extract and manipulate data from HTML documents. Furthermore, BeautifulSoup **offers powerful search features** that allow users to efficiently find specific tags and extract relevant data. BeautifulSoup's flexibility allows it to easily handle imperfect or poorly formatted HTML, making it an ideal tool for web scraping tasks where the source code may not always be well-structured.

Installing BeautifulSoup

To install `BeautifulSoup` and `requests`, follow these steps:

1. Open your terminal (Command Prompt, PowerShell, or any command-line tool).
2. Run the following command to install both libraries:

```
pip install beautifulsoup4 requests
```

3. If you encounter issues with installation, consider upgrading `pip` with:

```
pip install --upgrade pip
```

4. To verify the installation, open a Python shell by typing `python` and run:

```
import requests
from bs4 import BeautifulSoup
print("BeautifulSoup and requests are installed correctly!")
```

If no errors appear, the libraries are successfully installed and ready to use.

1. **Creating a Python Project Using VS Code**
    - Open Visual Studio Code (VS Code).
    - Create a new folder for your project (e.g., `WebAnalyzer`).
    - Open this folder in VS Code by selecting **File → Open Folder**.
    - Inside the folder, create a new file called `web_analyzer.py`.

2. **Crawl the UoC Wikipedia webpage**

We need to write the following code to 1) ask the user for a URL, 2) fetch the webpage content using `requests`, and 3) parse the content using `BeautifulSoup`:

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/University_of_Calgary"

try:
    response = requests.get(url)
    response.raise_for_status()  # Ensures the request was successful
    soup = BeautifulSoup(response.text, 'html.parser')
    print(f"Successfully fetched content from {url}")
except Exception as e:
    print(f"Error fetching content: {e}")
```

The soup variable is a BeautifulSoup object that represents the parsed HTML content of the fetched web page. It allows you to easily extract and manipulate HTML elements like headings, links, and paragraphs. Print the content of soup variable using the `prettify()` method:

```
print(soup.prettify())
```

3. **Data Analysis**

    Write a Python code in `web_analyzer.py` to count and display the:
    - Number of headings (`<h1>` to `<h6>`) in the HTML content.
    - Number of links (`<a>` tags).
    - Number of paragraphs (`<p>` tags).

    Hint: You can use `find_all` method.

4. **Keywords Analysis**

    Write a Python code that asks the user for a keyword and counts how many times the keyword appears in the webpage's content. Display the result in a clear format. Hint: you need to convert the soup output into text before performing the search.

5. **Word Frequency Analysis**

   Analyze the webpage's content to identify the most frequently used words. To achieve this:
   - Extract all the text content from the web page using `soup.get_text()`.
   - Split the text into individual words.
   - Converting all words to lowercase.
   - Count the frequency of each word.
   - Display the top 5 most frequently occurring words with their counts.

6. **Finding the Longest Paragraph**

   Write a Python code that displays the longest paragraph in the webpage, and the number of words does it contain. Ignore empty paragraphs or paragraphs with fewer than 5 words.

7. **Visualizing Results**

   The next step is to visualize the extracted data using matplotlib library. First, we need to install `matplotlib` by running the following command:

   ```
   pip install matplotlib
   ```

   Next, create a bar chart showing the counts of headings, links, and paragraphs computed in step 3:

```python
import matplotlib.pyplot as plt

labels = ['Headings', 'Links', 'Paragraphs']
values = [headings, links, paragraphs]

plt.bar(labels, values)
plt.title('Put your Group# Here')
plt.ylabel('Count')
plt.show()
```

Submission:

Fill out the names and UCIDs of all group members in Answer_sheet. Also, upload both the completed Answer_sheet.docx and the code to D2L.