Worksheet 3: String Pattern Matching and Web Crawling

Learning Objectives

- Understand how to perform string pattern matching using Python's `re` module.
- Learn the basics of web crawling using Python's `requests` and `BeautifulSoup` libraries.
- Gain hands-on experience in extracting useful information from web pages.

Activity 1: Understanding String Pattern Matching

Objective: Learn how to use Python’s `re` module for pattern matching.

Instructions:

1. Review the following sample code that demonstrates string pattern matching.
2. Answer the questions about the code provided.
3. Write your own regular expression to solve the given problems.

Sample Code:

```
import re

# Sample text
text = "John's phone number is 123-456-7890. Call him at 987-654-3210."

# Pattern to match phone numbers
pattern = r"\d{3}-\d{3}-\d{4}"

# Find all matches in the text
matches = re.findall(pattern, text)

# Display the matches
print("Phone numbers found:", matches)
```

Questions:

1. What does the pattern `\d{3}-\d{3}-\d{4}` represent?
2. How many phone numbers are found in the sample text?
3. Modify the code to extract only phone numbers starting with `987`.

Problem:
Write a Python script to find and extract all email addresses from the following text:

```
plaintext
Contact us at support@example.com, sales@company.org, or info@domain.net.
```

Activity 2: Extracting Data with Web Crawling

Objective: Learn how to fetch a webpage and extract specific information using `BeautifulSoup`.

Instructions:

1. Use the provided code snippet to fetch and parse a webpage.
2. Identify the key components of the code.
3. Modify the code to extract specific elements from a webpage.

Sample Code:

```
import requests
from bs4 import BeautifulSoup

# URL to crawl
url = "https://example.com"

# Fetch the webpage
response = requests.get(url)

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

# Extract all links
links = soup.find_all('a')

# Display the links
print("Links found on the webpage:")
for link in links:
    print(link.get('href'))
```

Questions:

1. What is the purpose of the `soup.find_all('a')` method?
2. How would you modify the code to extract all paragraphs (`<p>` tags) instead of links?

Problem:
Write a Python script to fetch the title (`<title>`) of a webpage and display it.

Activity 3: Building a Simple Web Crawler

Objective: Build a simple web crawler that extracts specific content from multiple pages.

Instructions:

1. Use the provided code to crawl multiple pages of a website.
2. Answer the questions about the code.
3. Write your own script to extract a list of article titles from a blog.

Sample Code:

```
import requests
from bs4 import BeautifulSoup

# Base URL of the blog
base_url = "https://example-blog.com"

# List to store article titles
titles = []

# Crawl the first 3 pages
for page in range(1, 4):
    # Construct the URL
    url = f"{base_url}/page/{page}"

    # Fetch the page
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Extract article titles
    articles = soup.find_all('h2', class_='article-title')
    for article in articles:
        titles.append(article.text)

# Display the titles
print("Article Titles Found:")
for title in titles:
    print(title)
```

Questions:

1. How does the `for` loop help in crawling multiple pages?
2. What is the role of the `class_='article-title'` argument in `soup.find_all()`?
3. Modify the script to save the extracted titles to a file named `titles.txt`.