An Introduction to Web Scraping and Beautiful Soup

Introduction

Now that you've gotten a taste of HTML and CSS, it's time to reverse engineer that knowledge to extract data from the web. In this lesson, you'll learn how to use Beautiful Soup, a Python package used for web scraping. You'll also get some hands-on scraping practice.

Objectives

You will be able to:

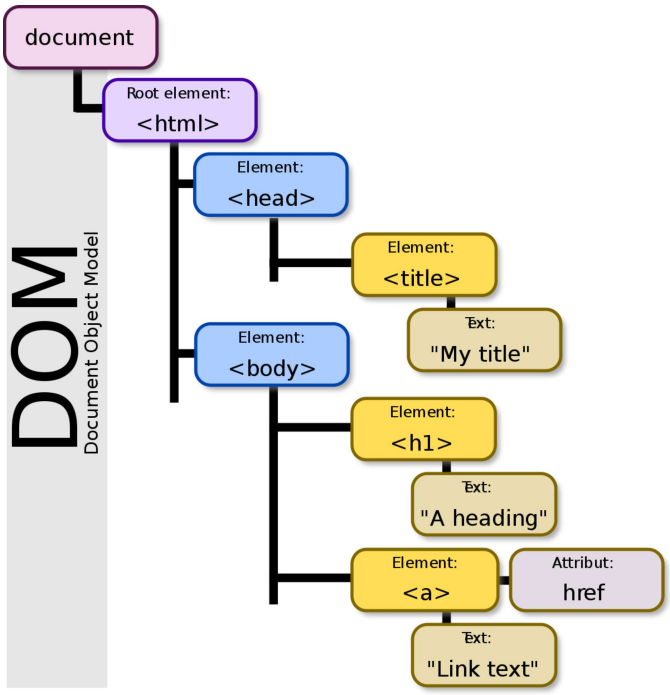
- Describe the DOM and its relationship to HTML
- Select specific elements from HTML using Beautiful Soup

Web Page Introduction: The DOM + HTML

Web pages can be represented by the objects that comprise their structure and content. This representation is known as the **Document Object Model (DOM)**. The purpose of the DOM is to provide an interface for programs to change the structure, style, and content of web pages. The DOM represents the document as nodes and objects. Amongst other things, this allows programming languages to interactively change the page and HTML!

What you'll see is the DOM and HTML create a hierarchy of elements. This structure and the underlying elements can be navigated similarly to a family tree which is one of Beautiful Soup's main mechanisms for navigation. Once you select a specific element within a page, you can then navigate to successive elements using methods to retrieve related tags including a tag's sibling, parent or descendants.

To learn more about the DOM see:
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)



Beautiful Soup

Beautiful Soup (https://www.crummy.com/software/BeautifulSoup/bs4/doc/) is a Python library designed for quick scraping projects. It allows you to select and navigate the tree-like structure of HTML documents, searching for particular tags, attributes or ids. It also allows you to then further traverse the HTML documents through relations like children or siblings. In other words, with Beautiful Soup, you could first select a specific div tag and then search through all of its nested tags.

In this lesson, you'll scrape a sample HTML document using Beautiful Soup! First, import it using the following code:

In [1]:
```
from bs4 import BeautifulSoup
```

An example webpage

Let's take a look at a very simple HTML page to see how you could select various elements using Beautiful Soup. First, open the `sample_page.html` file, then run the page through Beautiful Soup to get a `BeautifulSoup` object. `BeautifulSoup` objects are nested data structures that can be navigated easily!

In [2]:
```
with open('sample_page.html') as f:
    soup = BeautifulSoup(f, 'html.parser')
print(soup.prettify())
```

```
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link2">
    Tillie
   </a>
   ; and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

A few introductory Beautiful Soup Selections...

In [3]:
```
print(soup.title)
# <title>The Dormouse's story</title>

print(soup.title.name)
# u'title'

print(soup.title.string)
# u'The Dormouse's story'

print(soup.title.parent.name)
# u'head'

print(soup.p)
# <p class="title"><b>The Dormouse's story</b></p>

print(soup.p['class'])
# u'title'

print(soup.a)
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

print(soup.find_all('a'))
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

print(soup.find(id="link3"))
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

```
<title>
    The Dormouse's story
   </title>
title

    The Dormouse's story

head
<p class="title">
<b>
     The Dormouse's story
    </b>
</p>
['title']
<a class="sister" href="http://example.com/elsie" id="link1">
     Elsie
    </a>
[<a class="sister" href="http://example.com/elsie" id="link1">
     Elsie
    </a>, <a class="sister" href="http://example.com/lacie" id="link2">
     Lacie
    </a>, <a class="sister" href="http://example.com/tillie" id="link2">
     Tillie
    </a>]
None
```

Warnings and Precautions

While web scraping is a powerful tool, it can also lead you into ethical and legal gray areas. To start, it is possible to make hundreds of requests a second to a website. Browsing at superhuman speeds such as this is apt to get noticed. Large volumes of requests such as this are apt to bog down a website's servers and in extreme cases could be considered a denial of service attack. Similarly, any website requiring login may contain information that is thereby not considered public and scraping said websites could leave you in legal jeopardy. Use your best judgment when scraping and exercise precautions. Having your IP address blocked from your favorite website, for example, could prove to be quite an annoyance.

Additional Resources

Beautiful soup is the preliminary tool for web scraping. That said, there are more complex examples where you may wish to either scrape larger amounts of data through full-on web crawling or trickier examples involving javascript. For these and other scenarios, alternative tools such as Selenium and Scrapy are worth investigating.

Beautiful Soup - A good go-to tool for parsing the DOM

https://www.crummy.com/software/BeautifulSoup/ (https://www.crummy.com/software/BeautifulSoup/)?

Selenium - Browser automation (useful when you need to interact with javascript for more complex scraping)

https://www.seleniumhq.org/ (https://www.seleniumhq.org/)

Scrapy - Another package for scraping larger datasets at scale

https://scrapy.org/ (https://scrapy.org/)

Summary

You should now have a brief intro to web scraping! You have learned about the DOM and its relationship to HTML. You also saw how you can select specific elements from HTML using Beautiful Soup. The web scraping possibilities are nearly endless with what you can do. That said, be careful, as mentioned.