12.2 Lesson Summary - Kiss My Fist and Scrape the Sky

The Internet contains a wealth of data. Some of this data is neatly packaged for computer consumption in the form of web APIs but a lot of meaningful web data is formatted for human consumption in the form of HTML/CSS/JavaScript that is used to create the webpages we view in our browsers. The process of extracting data from a webpage is called web scraping. Scraping data from a webpage isn't quite as easy as getting data from a web API because it requires you to learn the HTML of the target webpage. Fortunately, Python provides a number of tools to assist you.

Concept: **Beautiful Soup** is a Python library that can convert HTML documents into a BeautifulSoup object that enables you to extract data from a webpage. To display the contents of the BeautifulSoup object and access the data contained within the *p* HTML tags you could use the following code:

*from bs4 import BeautifulSoup as bs*

*my\_html\_string = "<html><body><p>Simple HTML document</p></body></html>"*

*soup = bs(my\_html\_string, 'html.parser')*

*print(soup.prettify())*

*print(soup.p)*

*print(soup.p.text)*

* Activity: 01-Ins\_SoupIntro
* Suppl link: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

Concept: Beautiful Soup provides a ***find***and ***findall***method that allow you to locate HTML tags within a document. For example:

*my\_p\_with\_class\_tags = soup.find\_all('p', class\_=’my-class’)*

Beautiful Soup's *find* and *findall* methods allow you to locate HTML tags using the element's class. For example:

*my\_p\_with\_class\_tags = soup.find\_all('p', class\_='my-class')*

* Activity: 01-Ins\_SoupIntro, 02-Stu\_CNNSoup

Concept: To access data from live websites you can pair Beautiful Soup with Python's **requests** library. For example:

*from bs4 import BeautifulSoup*

*import requests*

*response = requests.get("https://www.google.com/")*

*soup = BeautifulSoup(response.text, 'html.parser')*

*print(soup.prettify())*

* Activity: 03-Ins\_Craigslist

Concept: Beautiful Soup let's you parse HTML but sometimes in order to scrape web data you must interact with the website in order to be able to access the data that you want. You may need to click on specific options or enter login information each time before you access a webpage. When you need to perform these kinds of interactions the Python **Splinter** module provides you with a way to automate your web browser's activity. You can use Splinter to click links and perform the interactions required to access the data you need. To simply browse to a page and read its HTML with Splinter you could use the following code:

*from splinter import Browser*

*from bs4 import BeautifulSoup*

*executable\_path = {'executable\_path': 'chromedriver.exe'}*

*browser = Browser('chrome', \*\*executable\_path, headless=False)*

*url = 'http://quotes.toscrape.com/'*

*browser.visit(url)*

*html = browser.html*

*soup = BeautifulSoup(html, 'html.parser')*

* Activity: 07-Ins\_Splinter, 08-Stu\_Splinter

Concept: Pandas provides you with the ability to extract data from HTML tables directly from a website. To extract the tables on Wikipedia's webpage about US capitals you can use the following code:

*import pandas as pd*

*tables = pd.read\_html(‘https://en.wikipedia.org/wiki/List\_of\_capitals\_in\_the\_United\_States’)*

* Activity: 09-Ins\_Pandas\_Scraping, 10-Stu\_Doctor\_Decoder