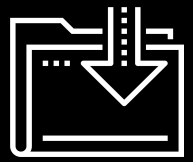
Web Scraping with CSS Selectors

Data Boot Camp

Lesson 11.2



What’s the difference between HTML and CSS?

What are some examples of HTML elements?

What is an HTML tag?

What is a CSS selector?

What is BeautifulSoup used for?

What does the BeautifulSoup `find()` method return?

Class Objectives

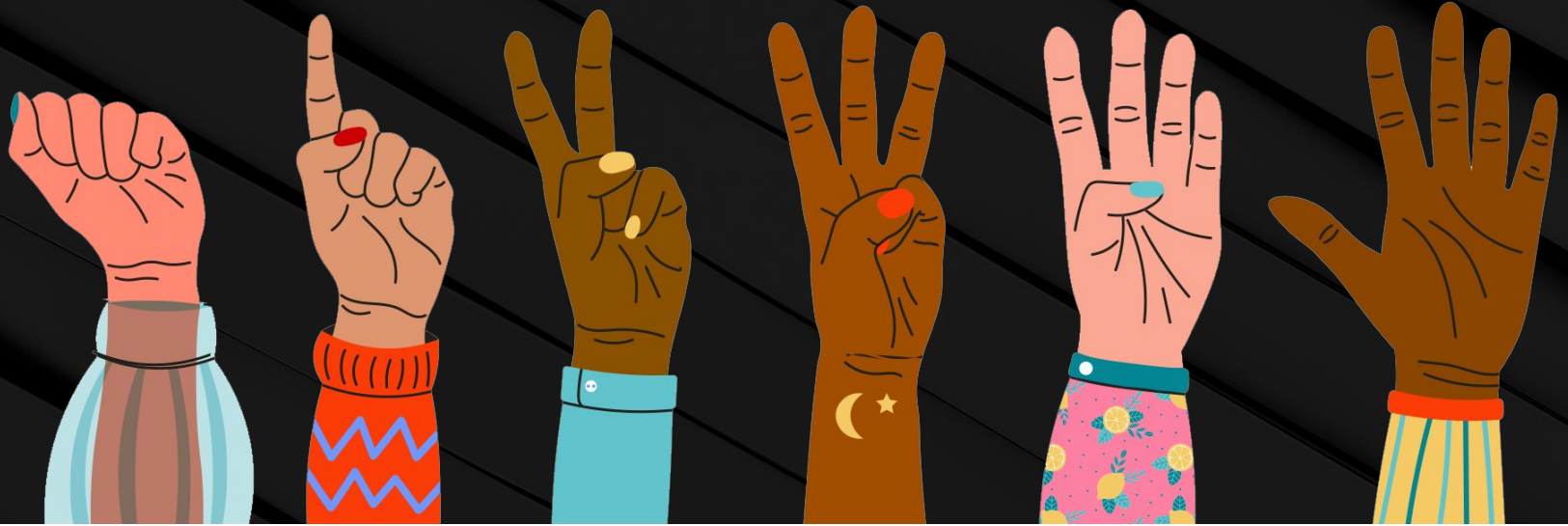
By the end of today's class you will be able to:

Use CSS selectors to scrape targeted elements.

Use Chrome DevTools to identify elements and their CSS selectors.

FIST TO FIVE:

How comfortable do you feel with this topic?



Instructor Demonstration

CSS Scraping

Activity: CSS Case Study

In this activity, you'll use CSS selectors in a basic web scraping example.

Suggested Time:

15 minutes

Activity: CSS Case Study

Instructions	
	Import BeautifulSoup.
	Save the provided HTML code as a Python string.
	Convert the HTML string into a BeautifulSoup object.
	Use the `find_all()` function to retrieve **all** of the elements that belong to the “odd” class.
	Save the results to a variable.
	Display the results by using a `for` loop.
	Extract the text from a paragraph element that has an `id` of “first”.
	Print your result.

Time's Up! Let's Review.

Activity: Pandas Scrape

In this activity, you will use BeautifulSoup to extract information from a simplified version of the official Pandas website.

Suggested Time:

15 minutes

Activity: Pandas Scrape

Instructions	Follow these steps to set up for web scraping: • Create a new Jupyter notebook and import BeautifulSoup. • Copy and paste the code of the provided HTML file into a Python string. • Create an instance of BeautifulSoup to parse the HTML. Use BeautifulSoup to retrieve the following: • The text of **all** `h3`-level headers. • The text and URLs of only the first section ("Getting Started"). Use the id of this section to scrape the data. • **All** URLs on the page.
Bonus	Each subsection on the page consists of a header, e.g. "Getting Started" and links. Write Python code to automate the collection of headers and link URLs in a logical structure.

Time's Up! Let's Review.


Break

Instructor Demonstration

DevTools

Activity: Stack Scrape

In this activity, you will scrape Stack Overflow's Python page and store the results in a Python list of dictionaries.

Suggested Time:

20 minutes

Activity: Stack Scrape

Instructions

Visit Stack Overflow's Python page with Splinter's automated browser. Scrape information from all the questions on the first page. For each question, use BeautifulSoup to scrape the following pieces of information:

- The summary (e.g. 'What does the "yield" keyword do?' for the first question)
- The number of votes for that question
- The excerpt (the longer text below the summary)

As you scrape, use Chrome's DevTools to identify elements and their CSS selectors, which you will then use in BeautifulSoup.

Next, organize your information into a Python list of dictionaries. That is, information from each question will be organized into a dictionary like the below, and a list will contain all these dictionaries.

Don't forget to close your automated browser!

Example dictionary

```
{'summary': 'What does the "yield" keyword do?',
  'votes': '11692',
  'excerpt': "What is the use of the yield keyword in Python? What does it do?\nFor example, I'm trying to understand this code1:\ndef _get_child_candidates(self, distance, min_dist, max_dist):\n    if self._leftchild ..."}
```

Time's Up! Let's Review.

Activity: Mars News Scrape

In this activity, you'll get additional web scraping practice by collecting data from a website based on NASA's Mars News.

Suggested Time:

20 minutes

Activity: Mars News Scrape

Instructions

Open up https://static.bc-edx.com/data/web/mars_news/index.html in Chrome and become familiar with the layout. Use DevTools to find the class of the title and summary of a news article.

Begin the scraping process by importing the necessary libraries and setting up Splinter.

Use Splinter to visit the the website and collect the html. Create a BeautifulSoup parser to parse the html from the website.

Use the `select_one` method from BeautifulSoup to search the html for a `<div>` tag that has the class associated with the news articles. Store your result in a variable.

Find the title of the news article you selected by using the `find` method and the class associated with the article titles. Print your result.

Use the `get_text` method to extract the text of the news article title that you scraped, and print your result.

Find the summary of the news article you selected by using the `find` method and the class associated with the article summary. Print your result.

Use the `get_text` method to extract the text of the news article summary that you scraped, and print your result.

Time's Up! Let's Review.


Recap

Questions?


The End