Automated Browsing

Data Boot Camp

Lesson 11.3

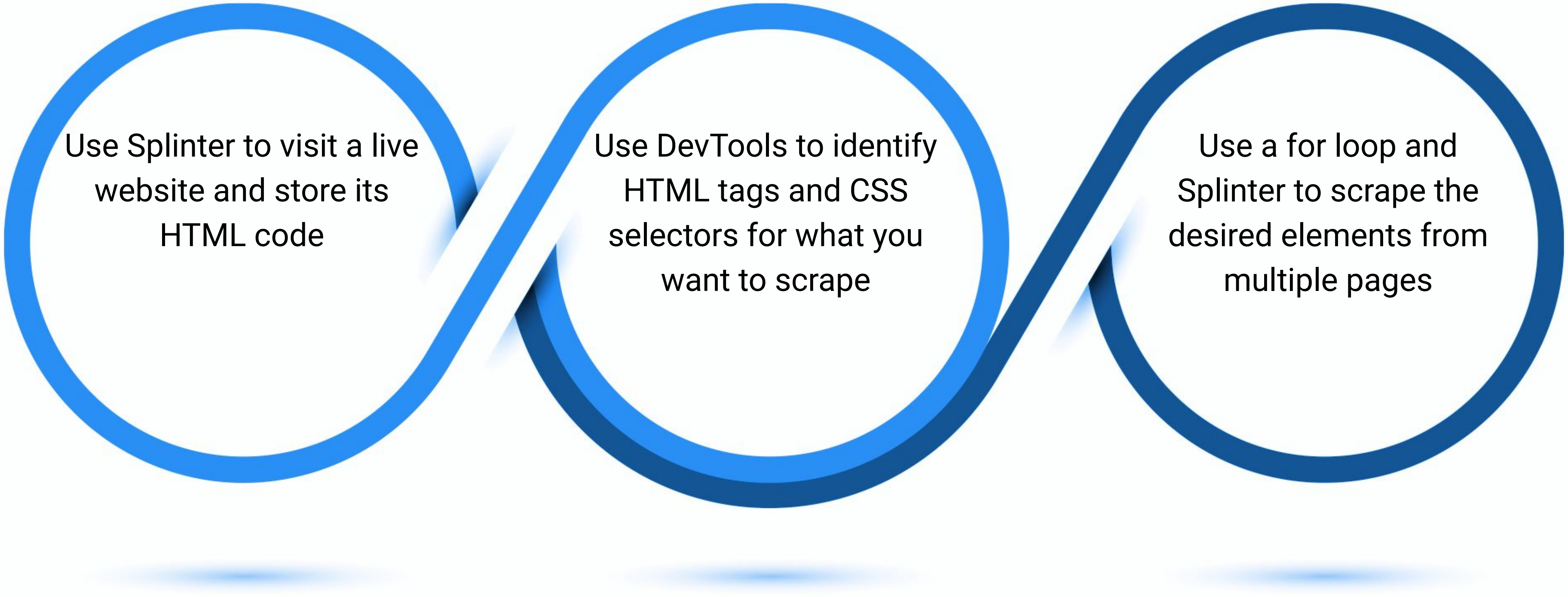
Class Objectives

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

- Use Splinter to perform automated browser actions.
- Automate the web scraping process by using Splinter and BeautifulSoup.
- Organize scraped information into a Python data structure.

Automated Web Scraping

Automated Web Scraping



Use Splinter to visit a live website and store its HTML code

Use DevTools to identify HTML tags and CSS selectors for what you want to scrape

Use a for loop and Splinter to scrape the desired elements from multiple pages

Ethics and Laws Around Web Scraping

Before collecting data from a website…

01

Always read a website's **Terms of Use** or **Terms of Service** before scraping the website

02

Look for sections that **prohibit automated scraping**, or directions on **using their data**

03

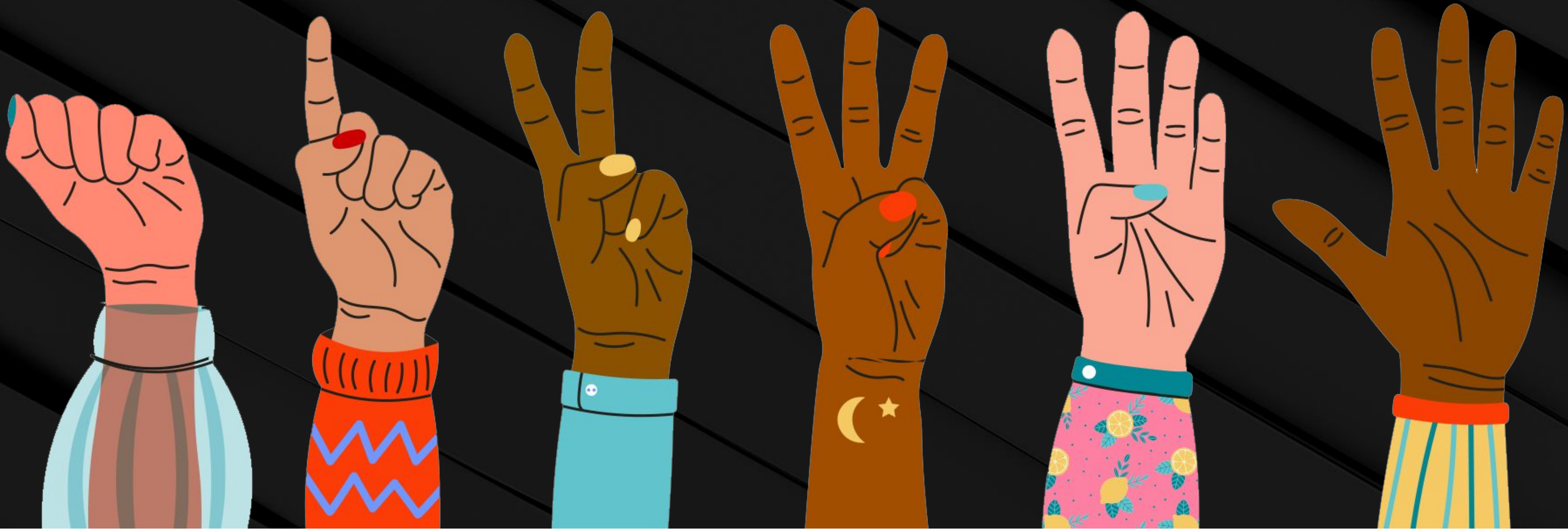
Research **recent legal cases** involving web scraping, since legal stances are evolving

Pro Tip:

Be extra cautious about personal data!

FIST TO FIVE:

How comfortable do you feel with this topic?



Practice Performing an Automated Web Scrape

In this activity, we'll scrape data from a website that was created specifically for practicing our skills: Quotes to Scrape

Suggested Time:

15 minutes

Activity: Practice Performing an Automated Web Scrape

Instructions	Open up Quotes to Scrape in Chrome and familiarize yourself with the page layout.
	Use DevTools to review the details of the “Top Ten tags” line.
	Scrape the "Top Ten tags" heading.
	Extract all of the Top Ten tags from the webpage.
	End the session by using `browser.quit()`

Time’s Up! Let’s Review.

Instructor Demonstration

Automated Web Scraping Across Multiple Pages

Scrape Book Links

In this activity, you'll practice automated scraping on another website that was created specifically for practicing our skills: Books to Scrape

Suggested Time:

15 minutes

Activity: Scrape Book Links

Inspect	Open up Books to Scrape in Chrome and familiarize yourself with the page layout. Where can you find a link to more details about each book?
	Inspect the element where the additional book details are linked. What class is the link contained within?
Scrape from One	Run the cells to import the necessary libraries and to set up the browser.
	Use the `browser.visit(url)` function to visit the website.
	Create a BeautifulSoup parser to parse the HTML code from the page you visited.
	Use the `find_all` function to locate and store all of the website content associated with the class you previously identified by using DevTools.
	Create a blank list, then use a `for` loop to extract each of the links associated with the books and store them in that list.
	Print the list to confirm that you were successful.

Activity: Scrape Book Links

Scrape from Multiple	Since you have confirmed that your code works and you are able to scrape the book links, now you can automate the scraping process by scraping multiple website pages.
	Look back at the page, and find the button that allows you to click the next page. How is it labeled?
	Use the label text that you found, the `browser.links.find_by_partial_text` method, and the `click` method to automatically move to the next page. Run your code and use the open browser window to confirm that you successfully advanced to the next page.
	Now use a for loop and the code you wrote for the previous section to scrape the book links from the first three pages of the website. Display your results by printing the page number, followed by the links on that page.
	Congratulations, you have automated the process of scraping multiple website pages! Now end the session by using `browser.quit()`.

Time’s Up! Let’s Review.

Countdown timer

15:00

(with alarm)

News Headers

In this activity, you will scrape news summaries across multiple pages from the Global Voices news site

Suggested Time:

20 minutes

Activity: News Headers

Instructions	
	Open page 2 of Global Voices news in Chrome. Use DevTools to identify the elements that contain the data you need to scrape.
	Within your Jupyter notebook, use Splinter to click on any popup or lightbox to make it disappear.
	Create an empty list to store your article summaries.
	Create a function that will perform your web scraping. The function should perform the following tasks: ● Collect the HTML from the browser. ● Parse the HTML and save it as a BeautifulSoup object. ● On any given page, scrape the **title** and the **date** of each article. ● For each article summary, the title and the date should be structured as a dictionary. All dictionaries should be contained in a Python list.
	Create a `for` loop that will: ● Call your web scraping function to scrape the article summaries on a page. ● Click the button to go to the next page of older articles. ● Repeat the process for a total of five pages.
	Import the scraped data (a list of dictionaries) into a Pandas dataframe. Using Python, Pandas, or both, convert the dates into a useable datetime type.

Time’s Up! Let’s Review.

Mars Fact Scrape

In this activity, you'll practice scraping data that was stored in a table on a website.

Suggested Time:

15 minutes

Activity: Mars Fact Scrape

Instructions

Open up the [Mars Facts]() website in Chrome and become familiar with the layout.

You'll scrape the data from the table labeled "Mars Planet Profile." Use Chrome DevTools to inspect that element. What is the class of the table you want to scrape?

In Jupyter Notebook, Import the necessary libraries and set up Splinter. Use Splinter to visit the Mars Facts site and collect the html. Create a BeautifulSoup parser to parse the html from the website.

Scrape and store the table by using the `find` method alongside the class you identified by using Chrome DevTools.

Store the table information in a dictionary by following these steps:

- Create an empty dictionary.
- Use the `find_all` method to store all the rows of the table in a variable.
- Use a `for` loop to iterate through each of the rows in the table.
- For each row in the table, use the `find` method to extract the text for each header and to extract the text for the data associated with that header.
- Add the extracted header and associated data in the dictionary, with the header as the key and the data as the value.

Display your scraped table dictionary to confirm the process was successful.

Quit your browsing session.

Time’s Up! Let’s Review.

Instructor Demonstration

Scrape a Table with Pandas

This Week's Challenge

Scraping Mars Websites

Questions?

Recap

The End