10.3.1 Use Splinter

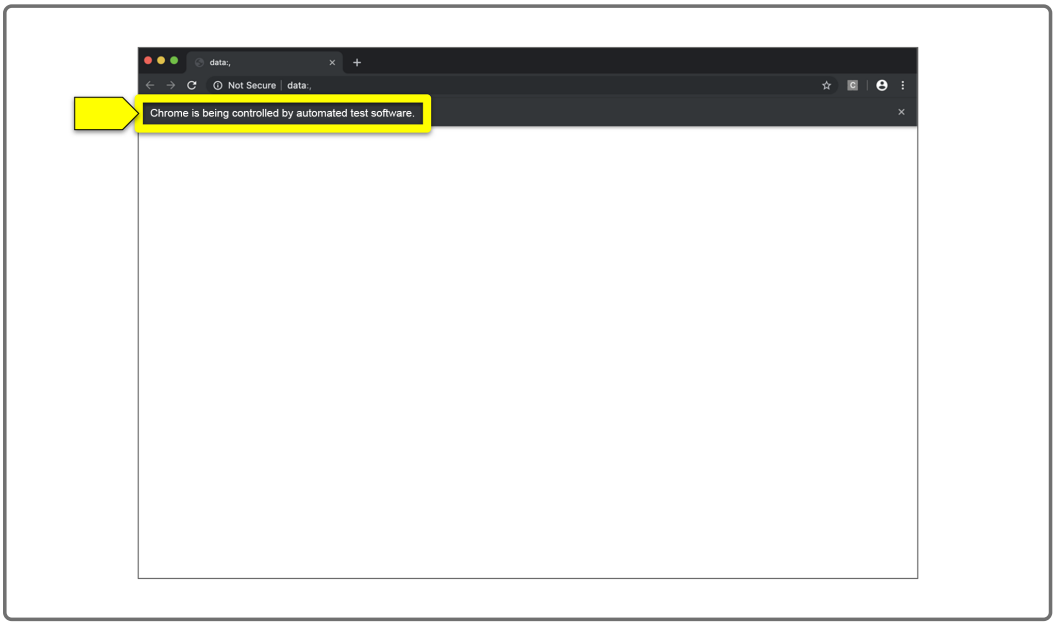
Robin is a bit more familiar with HTML tags and how they fit together to create a webpage, which is a great first step. She also has the necessary tools installed to get started with the scraping, so she's eager to dive in.

The next part is to use Splinter to automate a browser—this is pretty fun because we'll actually be able to watch a browser work without us clicking anywhere or typing in fields, such as using a search bar or next button.

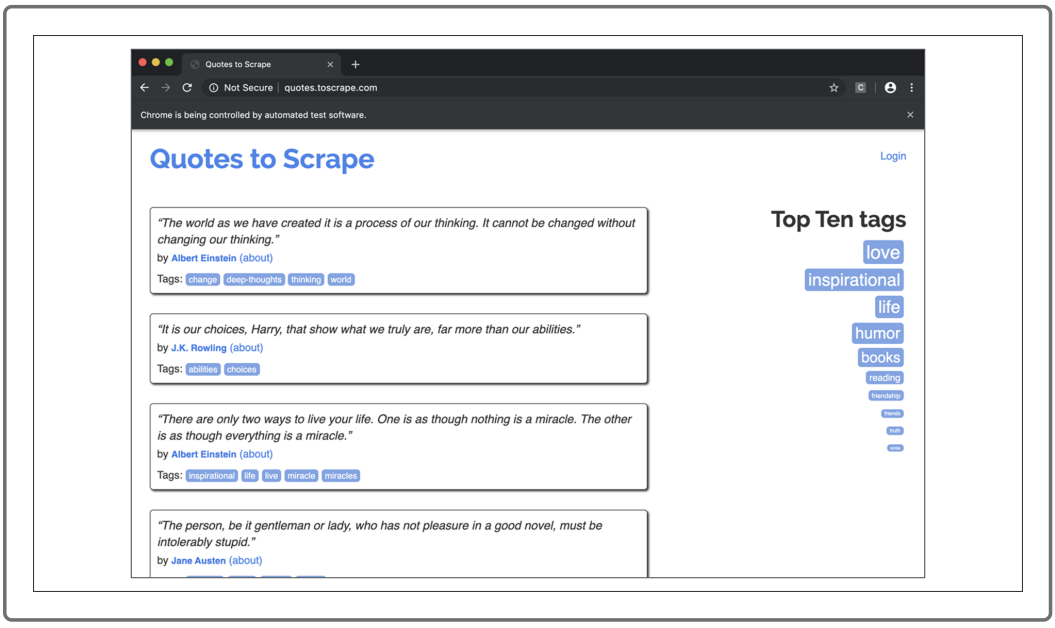
After we help Robin get Splinter rolling, we'll actually scrape data using BeautifulSoup. This is where our practice with HTML tags comes in. To scrape the data we want, we'll have to tell BeautifulSoup which HTML tag is being used and if it has an attribute such as a specific class or id.

One of the fun things about web scraping is the automation—watching your script at work.

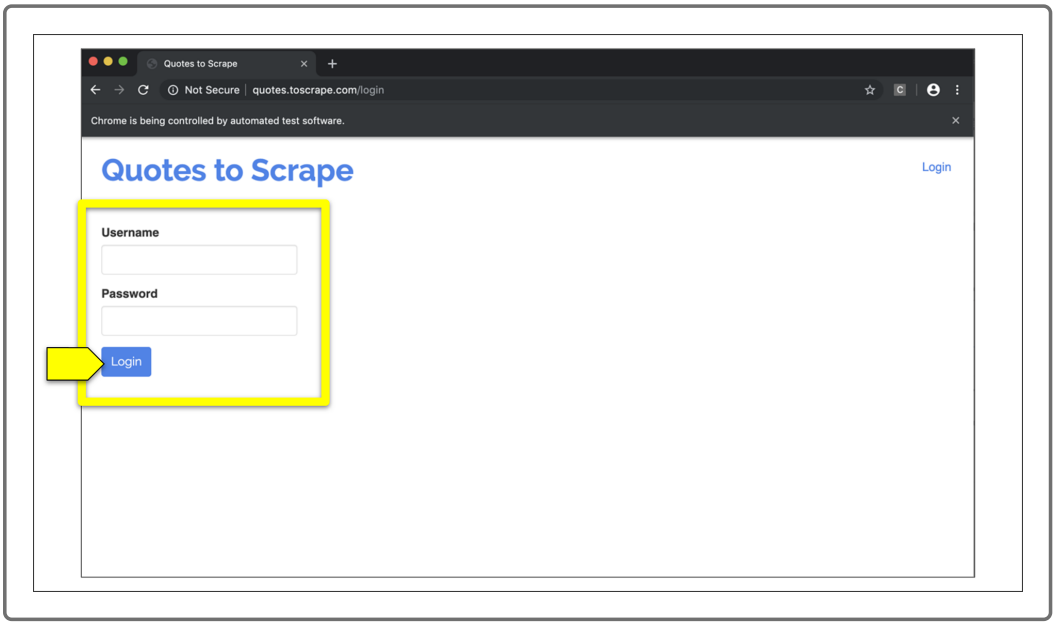
1. Once you execute your completed scraping script, a new Chrome web browser will pop up with a banner across the top that says "Chrome is being controlled by automated test software."



2. This message lets you know that your Python script is directing the browser. The browser will visit websites and interact with them on its own.



3. Depending on how you've programmed your script, your browser will click buttons, use a search bar, or even log in to a website.



Navigate to your Mission-to-Mars folder using the terminal. Then go ahead and activate Jupyter Notebook. Create a new `.ipynb` file to get started—this is where we'll begin our web scraping work. Let's name it "Practice." It can be deleted when we're done, or used as a reference later on. It's not necessary, but you can add it to your GitHub repo and to your `.gitignore` file so that it's hidden from public view.

REWIND

`.gitignore` is a ext file that contains the names of files you don't want the public to see, such as configuration files, or files that aren't necessary for the completed project, but you want to keep for reference.

In the very first cell, we'll import our scraping tools: the Browser instance from splinter, the BeautifulSoup object, and the driver object for Chrome, ChromeDriverManager.

```
from splinter import Browser
from bs4 import BeautifulSoup as soup
from webdriver_manager.chrome import ChromeDriverManager
```

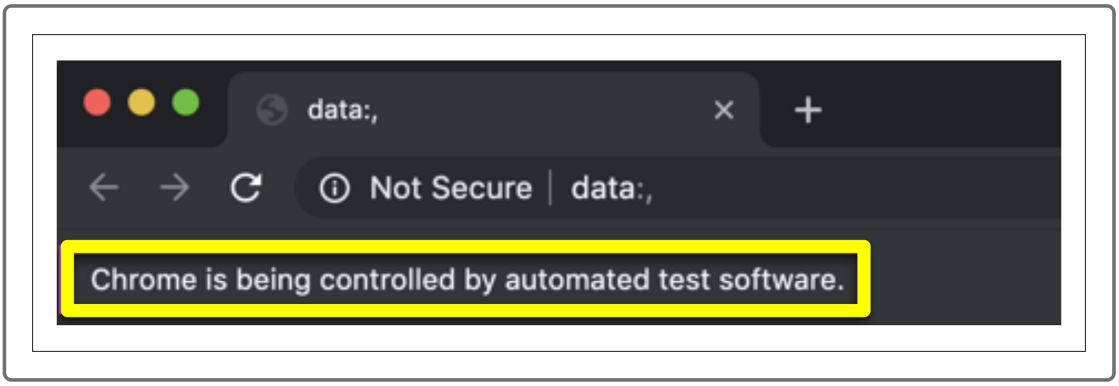
We're using an alias, "soup," to simplify our code a bit when we reference it later.

Next, we'll set the executable path and initialize a browser:

```
# Set up Splinter
executable_path = {'executable_path': ChromeDriverManager().install()}
browser = Browser('chrome', **executable_path, headless=False)
```

With these two lines of code, we are creating an instance of a Splinter browser. This means that we're prepping our automated browser. We're also specifying that we'll be using Chrome as our browser. `**executable_path` is unpacking the dictionary we've stored the path in – think of it as unpacking a suitcase. `headless=False` means that all of the browser's actions will be displayed in a Chrome window so we can see them.

You should have three cells ready to be run; go ahead and execute them. The third cell that initiates a Splinter browser may take a couple of seconds to finish, but an empty webpage should automatically open, ready for instructions. You'll know that it's an automated browser because it'll have a special message stating so, right under the tab, as shown below:



This browser now belongs to Splinter (for the duration of our coding, anyway). It's a lot of fun to watch Splinter do its thing and navigate through webpages without us physically interacting with any components. It's also a great way to make sure our code is working as we want it to. While the window can be closed at any time, it's generally not a good idea to shut down the browser without ending the session properly – there's an excellent chance your code will fail or an error will be generated.

NOTE

> Splinter provides us with many ways to interact with webpages. It can input terms into a Google search bar for us and click the Search button, or even log us into our email accounts by inputting a username and password combination.