Exploring Selenium

In this lesson, a scraping and automation tool named selenium is discussed.

We'll cover the following

Selenium

Selenium is a browser automation tool. It is mostly used for automated testing purposes but in this course, this tool will be used to scrape information from websites.

While scraping information, all the information we need is never simply on a single page. Some buttons need to be clicked to traverse through the website to get to the required data, and then it is scraped and stored. Detailed info on selenium can be found here.

How it works

Selenium opens a browser and goes to the initial link given to it. Then, the user has the power to make it do anything as if they are controlling the browser behavior themselves. It locates specific elements in the *HTML DOM* structure and applies the action we want on that element. Selenium selects the elements on a web page through the following keywords:

- `id`: The id element of *HTML*
- `class`: The class element of *HTML*
- `css_selector`: Elements that affect the **CSS** of a webpage; all elements that let you change the `CSS` of a page can be used here. Some examples are `a` tag, `p` tag, etc

- `x_path` : Completes the path to certain elements in a webpage like the following one: `//[@id="something"]/div/div/div[2]/div[1]/div/div/div/div/div/div[4]/div/div[1]/div[1]/div/div[2]/div/div/div[2]/div[1]`. More on `x_path` and how to obtain it can be found here.

After selecting an element, it gets the complete structure of all the elements that are inside the selected element. The user can then choose from those nested elements as well. We'll mostly be using the `x_path` selector for locating elements.

Setup

To successfully run selenium with python, two things are needed: a *driver* that opens the browser of our choice and the selenium package for python.

- Every browser has a different driver that works with selenium to operate it. Our code will use *Google Chrome*, so we'll use the *chromedriver*. It can be downloaded from here.

- The selenium package can be installed by the following command:

```
pip install selenium
```

The following example demonstrates how selenium works.

```
# import the webdriver package from selenium
from selenium import webdriver

# Set properties of the browser
chrome_options=webdriver.ChromeOptions()
#chrome_options.add_argument("--headless")
chrome_options.add_argument("--disable-extensions")
chrome_options.add_argument("--disable-gpu")
chrome_options.add_argument("--no-sandbox")

# Initialize the browser object
driver=webdriver.Chrome("<Provide the path of chromedriver>", options=chrome_options)

# The initial url to be opened
url = "https://www.amazon.com/Searchlight-Comics-Comic-bundle-Marvel/dp/B00VIOHOAQ/ref=sr_1_2?keywo
# Open the given url in the browser
driver.get(url)

# Get the name of the comic book using its Xpath
name = driver.find_element_by_xpath('//*[@id="productTitle"]').text
print("The name of the Comic Book is:", name)
```

```
# Close the browser
driver.close()
```

On **lines 5-9**, the settings for the browser to appear are being set. The `--headless` is commented out; this setting makes the browser invisible while scraping data. If not used, the browser is visible while scraping.

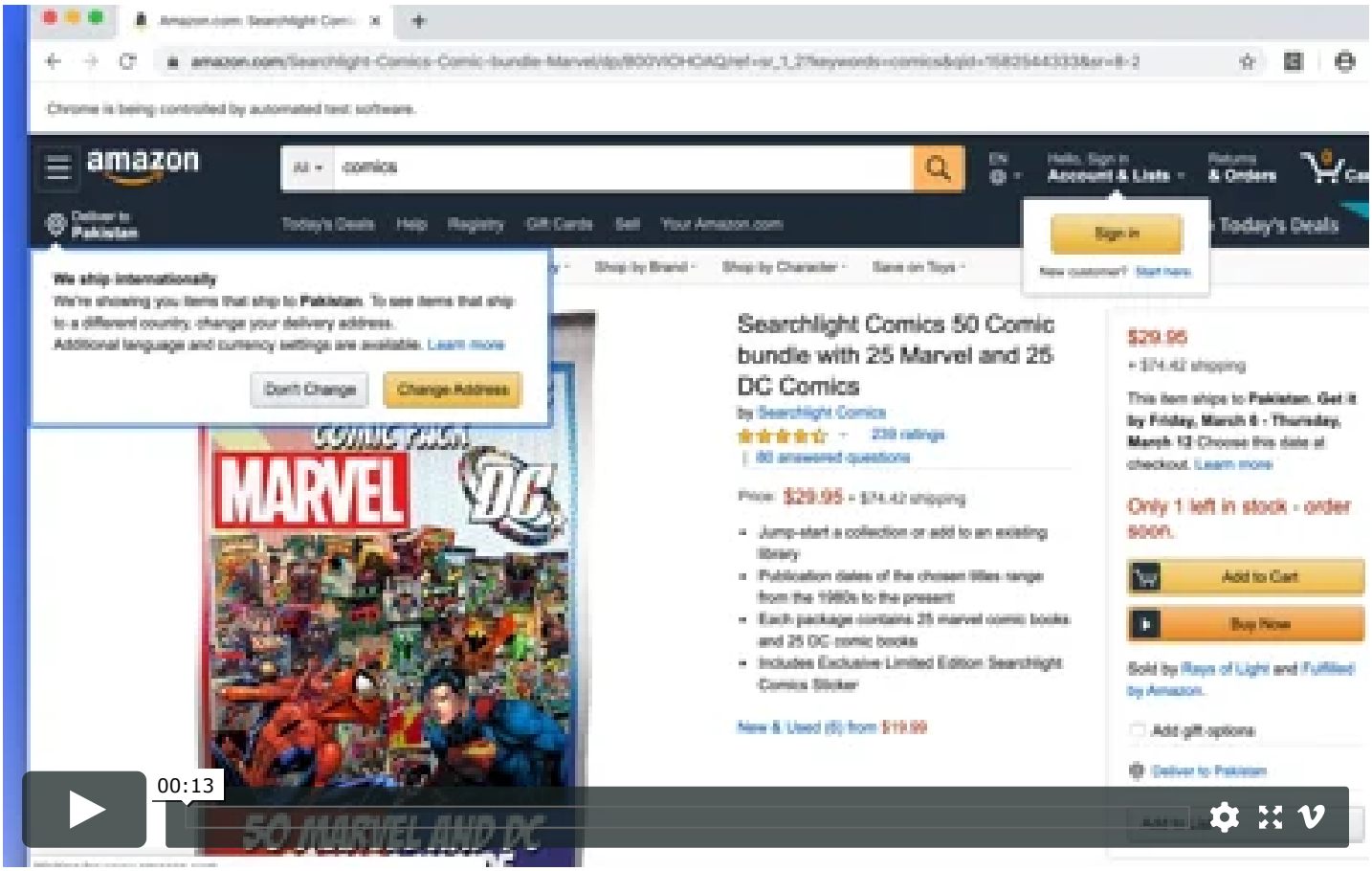
On **line 12**, the browser instance is declared, which takes the path to the `chromedriver` and the settings as arguments.

On **lines 15 & 17**, the URL to go is provided, and the browser instance is used to open the page linked to that URL.

On **line 20**, the `find_element_by_xpath` function of the browser instance is used to search for the name of the comic book on the entire page, and the `text` is used to extract all the textual information from it.

On **line 24**, the opened browser is closed using the `close` function of the browser instance.

The following video shows how a browser opens, navigates to the provided URL, fetches the name of the comic book, displays it on the console, and then closes itself.



Usage

Now, just imagine what kind of tasks can be automated using this remarkable python package. Customized bots can be developed to scrape data from sites like **Amazon**, **eBay**, **Yelp**, **Yellow Pages**, etc. Any number of useful analyses can be performed on that kind of data.

The selenium package has countless functions that help in scraping data from all kinds of web elements. Go through the selenium documentation here to learn about different scenarios and functions provided by the selenium package of python.

In the next lesson, a practical example of scraping is displayed.