**Lab. 7: Data Crawling using Python**

**How Do You Scrape Data From A Website?**

When you run the code for web scraping, a request is sent to the URL that you have mentioned. As a response to the request, the server sends the data and allows you to read the HTML or XML page. The code then, parses the HTML or XML page, finds the data and extracts it.

To extract data using web scraping with python, you need to follow these basic steps:

1. Find the URL that you want to scrape
2. Inspecting the Page
3. Find the data you want to extract
4. Write the code
5. Run the code and extract the data
6. Store the data in the required format

**Libraries used for Web Scraping**

As we know, Python is has various applications and there are different libraries for different purposes. In our further demonstration, we will be using the following libraries:

* **Selenium**:  Selenium is a web testing library. It is used to automate browser activities.
* **BeautifulSoup**: Beautiful Soup is a Python package for parsing HTML and XML documents. It creates parse trees that is helpful to extract the data easily.
* **Pandas**: Pandas is a library used for data manipulation and analysis. It is used to extract the data and store it in the desired format.

**Web Scraping Example : Scraping Flipkart Website**

Pre-requisites:

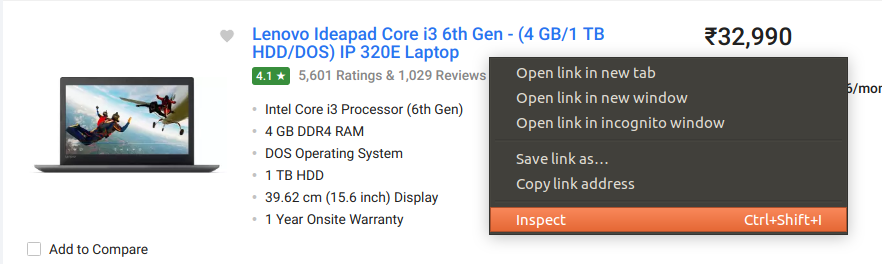
* Python 2.x or Python 3.x with **Selenium**, **BeautifulSoup, pandas** libraries installed
* Google-chrome browser
* Ubuntu Operating System

**Step 1: Find the URL that you want to scrape**

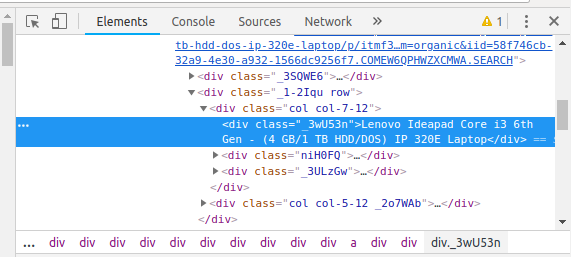
For this example, we are going scrape **Flipkart** website to extract the Price, Name, and Rating of Laptops. The URL for this page is <https://www.flipkart.com/laptops/~buyback-guarantee-on-laptops-/pr?sid=6bo%2Cb5g&uniqBStoreParam1=val1&wid=11.productCard.PMU_V2>.

**Step 2: Inspecting the Page**

The data is usually nested in tags. So, we inspect the page to see, under which tag the data we want to scrape is nested. To inspect the page, just right click on the element and click on “Inspect”.



When you click on the “Inspect” tab, you will see a “Browser Inspector Box” open.



**Step 3: Find the data you want to extract**

Let’s extract the Price, Name, and Rating which is in the “div” tag respectively.

**Step 4: Write the code**

First, let us import all the necessary libraries:

|  |
| --- |
| **from** selenium **import** webdriver  **from** BeautifulSoup **import** BeautifulSoup  **import** pandas as pd |

To configure webdriver to use Chrome browser, we have to set the path to chromedriver

|  |
| --- |
| driver **=** webdriver.Chrome("/usr/lib/chromium-browser/chromedriver") |

Refer the below code to open the URL:

|  |
| --- |
| products**=**[] #List to store name of the product  prices**=**[] #List to store price of the product  ratings**=**[] #List to store rating of the product  driver.get("<a href="https://www.flipkart.com/laptops/">https://www.flipkart.com/laptops/</a>~buyback-guarantee-on-laptops-/pr?sid=6bo%2Cb5g&amp;amp;amp;amp;amp;amp;amp;amp;amp;uniq") |

Now that we have written the code to open the URL, it’s time to extract the data from the website. As mentioned earlier, the data we want to extract is nested in <div> tags. So, I will find the div tags with those respective class-names, extract the data and store the data in a variable. Refer the code below:

|  |
| --- |
| content **=** driver.page\_source  soup **=** BeautifulSoup(content)  **for** a **in** soup.findAll('a',href**=**True, attrs**=**{'class':'\_31qSD5'}):  name**=**a.find('div', attrs**=**{'class':'\_3wU53n'})  price**=**a.find('div', attrs**=**{'class':'\_1vC4OE \_2rQ-NK'})  rating**=**a.find('div', attrs**=**{'class':'hGSR34 \_2beYZw'})  products.append(name.text)  prices.append(price.text)  ratings.append(rating.text) |

**Step 5: Run the code and extract the data**

To run the code, use the below command:

|  |
| --- |
| python web**-**s.py |

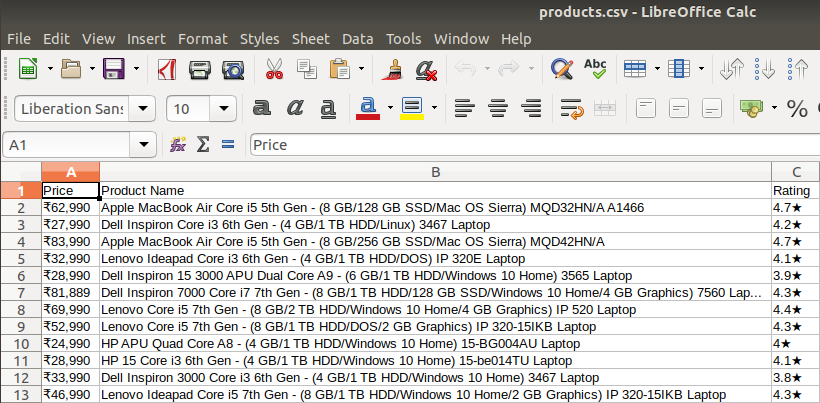
**Step 6: Store the data in a required format**

After extracting the data, you might want to store it in a format. This format varies depending on your requirement. For this example, we will store the extracted data in a CSV (Comma Separated Value) format. To do this, I will add the following lines to my code:

|  |
| --- |
| df **=** pd.DataFrame({'Product Name':products,'Price':prices,'Rating':ratings})  df.to\_csv('products.csv', index**=**False, encoding**=**'utf-8') |

Now, I’ll run the whole code again.

A file name “products.csv” is created and this file contains the extracted data.

[](https://www.edureka.co/blog/wp-content/uploads/2018/11/output-2-1.png)