**A Beginner’s Guide to learn web scraping with python!**

**Web Scraping with Python**

Imagine you have to pull a large amount of data from websites and you want to do it as quickly as possible. How would you do it without manually going to each website and getting the data? Well, “Web Scraping” is the answer. Web Scraping just makes this job easier and faster.

In this article on **Web Scraping with Python**, you will learn about web scraping in brief and see how to extract data from a website with a demonstration. I will be covering the following topics:

* + - [Why Web Scraping?](https://www.edureka.co/blog/web-scraping-with-python/#whywebscraping)
    - [What Is Web Scraping? Is Web Scraping Legal?](https://www.edureka.co/blog/web-scraping-with-python/#whyweb)
    - [Why Python For Web Scraping?](https://www.edureka.co/blog/web-scraping-with-python/#whypython)
    - [How Does Web Scraping work?](https://www.edureka.co/blog/web-scraping-with-python/#steps)
    - [Libraries used for Web Scraping](https://www.edureka.co/blog/web-scraping-with-python/#lib)
    - [Demo: Scraping Flipkart Website](https://www.edureka.co/blog/web-scraping-with-python/#demo)

## ****Why Web Scraping?****

Web scraping is used to collect large information from websites. But why does someone have to collect such large data from websites? To know about this, let’s look at the applications of web scraping:

* **Price Comparison:** Services such as ParseHub use web scraping to collect data from online shopping websites and use it to compare the prices of products.
* **Email address gathering:** Many companies that use email as a medium for marketing, use web scraping to collect email ID and then send bulk emails.
* **Social Media Scraping:** Web scraping is used to collect data from Social Media websites such as Twitter to find out what’s trending.
* **Research and Development:** Web scraping is used to collect a large set of data (Statistics, General Information, Temperature, etc.) from websites, which are analyzed and used to carry out Surveys or for R&D.
* **Job listings:** Details regarding job openings, interviews are collected from different websites and then listed in one place so that it is easily accessible to the user.

## ****What is Web Scraping? Is Web Scraping legal?****

Web scraping is an automated method used to extract large amounts of data from websites. The data on the websites are unstructured. Web scraping helps collect these unstructured data and store it in a structured form. There are different ways to scrape websites such as online Services, APIs or writing your own code. In this article, we’ll see how to implement web scraping with python.

[](https://www.edureka.co/blog/wp-content/uploads/2018/11/Untitled-1.jpg)Talking about whether web scraping is legal or not, some websites allow web scraping and some don’t. To know whether a website allows web scraping or not, you can look at the website’s “robots.txt” file. You can find this file by appending “/robots.txt” to the URL that you want to scrape. For this example, I am scraping Flipkart website. So, to see the “robots.txt” file, the URL is [www.flipkart.com/robots.txt.](http://www.flipkart.com/robots.txt)

## ****Why Python for Web Scraping?****

You’ve probably heard of how awesome Python is. But, so are other languages too. Then why should we choose Python over other languages for web scraping?

Here is the list of features of Python which makes it more suitable for web scraping.

* **Ease of Use:** Python is simple to code. You do not have to add semi-colons “;” or curly-braces “{}” anywhere. This makes it less messy and easy to use.
* **Large Collection of Libraries:** Python has a huge collection of libraries such as [Numpy](https://www.edureka.co/blog/python-numpy-tutorial/), [Matlplotlib](https://www.edureka.co/blog/python-matplotlib-tutorial/), [Pandas](https://www.edureka.co/blog/python-pandas-tutorial/) etc., which provides methods and services for various purposes. Hence, it is suitable for web scraping and for further manipulation of extracted data.
* **Dynamically typed:** In Python, you don’t have to define datatypes for variables, you can directly use the variables wherever required. This saves time and makes your job faster.
* **Easily Understandable Syntax:** Python syntax is easily understandable mainly because reading a Python code is very similar to reading a statement in English. It is expressive and easily readable, and the indentation used in Python also helps the user to differentiate between different scope/blocks in the code.
* **Small code, large task:** Web scraping is used to save time. But what’s the use if you spend more time writing the code? Well, you don’t have to. In Python, you can write small codes to do large tasks. Hence, you save time even while writing the code.
* **Community:** What if you get stuck while writing the code? You don’t have to worry. Python community has one of the biggest and most active communities, where you can seek help from.

## ****How does Web Scraping work?****

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

Now let us see how to extract data from the Flipkart website using Python.

## ****Libraries used for Web Scraping****

As we know, Python is used for 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.

## ****Demo: Scraping Flipkart Website****

Pre-requisites:

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

Let’s get started!

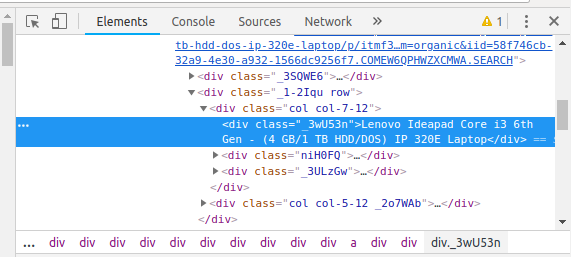
### ****Step 1: Find the URL that you want to scrape****

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 nested in the “div” tag respectively.

### ****Step 4: Write the code****

First, let’s create a Python file. To do this, open the terminal in Ubuntu and type gedit <your file name> with .py extension.

I am going to name my file “web-s”. Here’s the command:

|  |  |
| --- | --- |
| 1 | gedit web-s.py |

Now, let’s write our code in this file.

First, let us import all the necessary libraries:

|  |  |
| --- | --- |
| 1  2  3 | 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

|  |  |
| --- | --- |
| 1 | driver = webdriver.Chrome("/usr/lib/chromium-browser/chromedriver") |

Refer the below code to open the URL:

|  |  |
| --- | --- |
| 1  2  3  4 | 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&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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 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:

|  |  |
| --- | --- |
| 1 | python web-s.py |

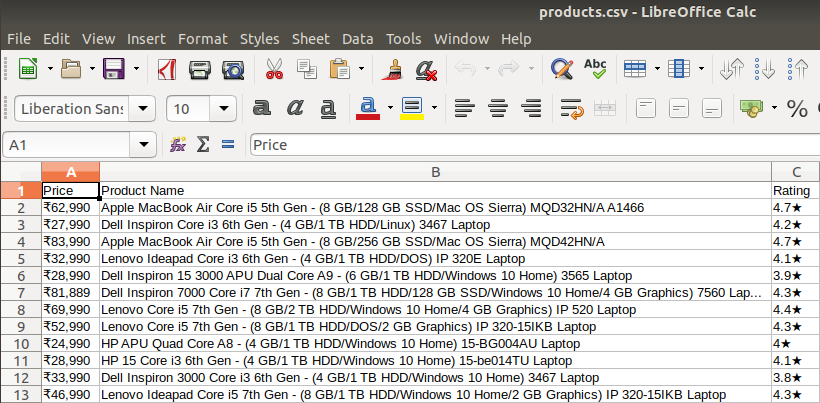
### ****Step 6: Store the data in a required format****

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:

|  |  |
| --- | --- |
| 1  2 | 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)

I hope you guys enjoyed this article on “Web Scraping with Python”. I hope this blog was informative and has added value to your knowledge. Now go ahead and try Web Scraping. Experiment with different modules and [applications of Python](https://www.edureka.co/blog/python-tutorial/#pythonApplication).