

# Chapter 8

Data Crawling function

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. It just makes this **job easier and faster.**

# Why is Web Scraping?

- Web scraping is used to collect large information from websites.
- But why does someone have to collect such large data from websites?

# Why is Web Scraping?

- **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.*
- **Price Comparison:** Services such as Gocompare.com 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.
- **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?

- 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.



# Is Web Scraping Legal?

- 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, if we are scraping Flipkart website, we can see the "robots.txt" file, the URL is [www.flipkart.com/robots.txt](http://www.flipkart.com/robots.txt).

# Why is Python Good for Web Scrapping?

- **Ease of Use:** Python is simple to code.
- **Large Collection of Libraries:** Python has a huge collection of libraries such as [Numpy](#), [Matplotlib](#), [Pandas](#) etc., which provides methods and services for various purposes.
- **Dynamically typed:** In Python, you don't have to define datatypes for variables, you can directly use the variables wherever required.
- **Easily Understandable Syntax:** Python syntax is easily understandable mainly because reading a Python code is very similar to reading a statement in English.
- **Small code, large task:** In Python, you can write small codes to do large tasks. Hence, you save time even while writing the code.
- **Community:** Python community has one of the biggest and most active communities, where you can seek help from.

# How To 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.



# How To Scrape Data From A Website?

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

Let see how to extract data from the Flipkart website using Python.

# Libraries used for Web Scrapping

Please use the following libraries:

- **BeautifulSoup**: BeautifulSoup is a Python package for parsing HTML and XML documents. It creates parse trees that is helpful to extract the data easily.
- **Selenium**: Selenium is a web testing library. It is used to automate browser activities.
- **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.

# Let's get started!

## STEP 1

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

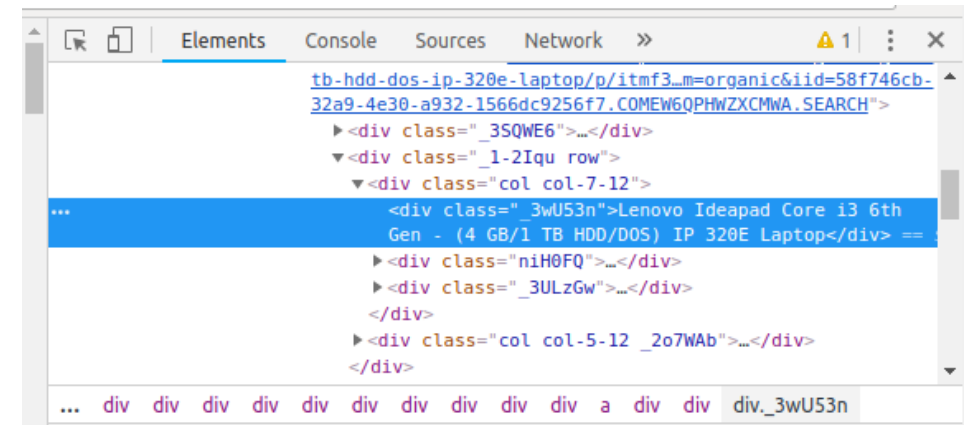
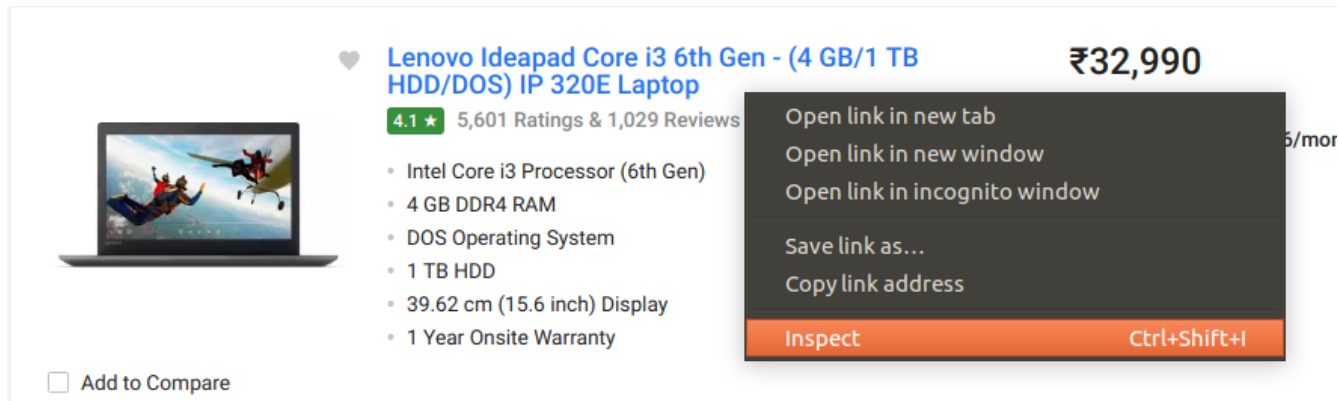
- For this example, we are going to scrape **Flipkart** website to extract the Price, Name, and Rating of Laptops.
- The URL for this page is:

[https://www.flipkart.com/laptops/~buyback-guarantee-on-laptops-/pr?sid=6bo%2Cb5g&uniqBStoreParam1=val1&wid=11.productCard.PMU\\_V2](https://www.flipkart.com/laptops/~buyback-guarantee-on-laptops-/pr?sid=6bo%2Cb5g&uniqBStoreParam1=val1&wid=11.productCard.PMU_V2).

# STEP 2

## 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

## **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

## 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.

Rename the file as "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 from selenium import webdriver
2 from BeautifulSoup import BeautifulSoup
3 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:

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:

# STEP 4

## Step 4: Write the code (cont')

Refer the below code to open the URL:

```
1 products=[] #List to store name of the product
2 prices=[] #List to store price of the product
3 ratings=[] #List to store rating of the product
4 driver.get("https://www.flipkart.com/laptops/~buyback-guarantee-on-laptops-
/pr?sid=6bo%2Cb5g&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:

```
1 content = driver.page_source
2 soup = BeautifulSoup(content)
3 for a in soup.findAll('a',href=True, attrs={'class':'_31qSD5'}):
4 name=a.find('div', attrs={'class':'_3wU53n'})
5 price=a.find('div', attrs={'class':'_1vC4OE _2rQ-NK'})
6 rating=a.find('div', attrs={'class':'hGSR34 _2beYZw'})
7 products.append(name.text)
8 prices.append(price.text)
9 ratings.append(rating.text)
```

# Step 5

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



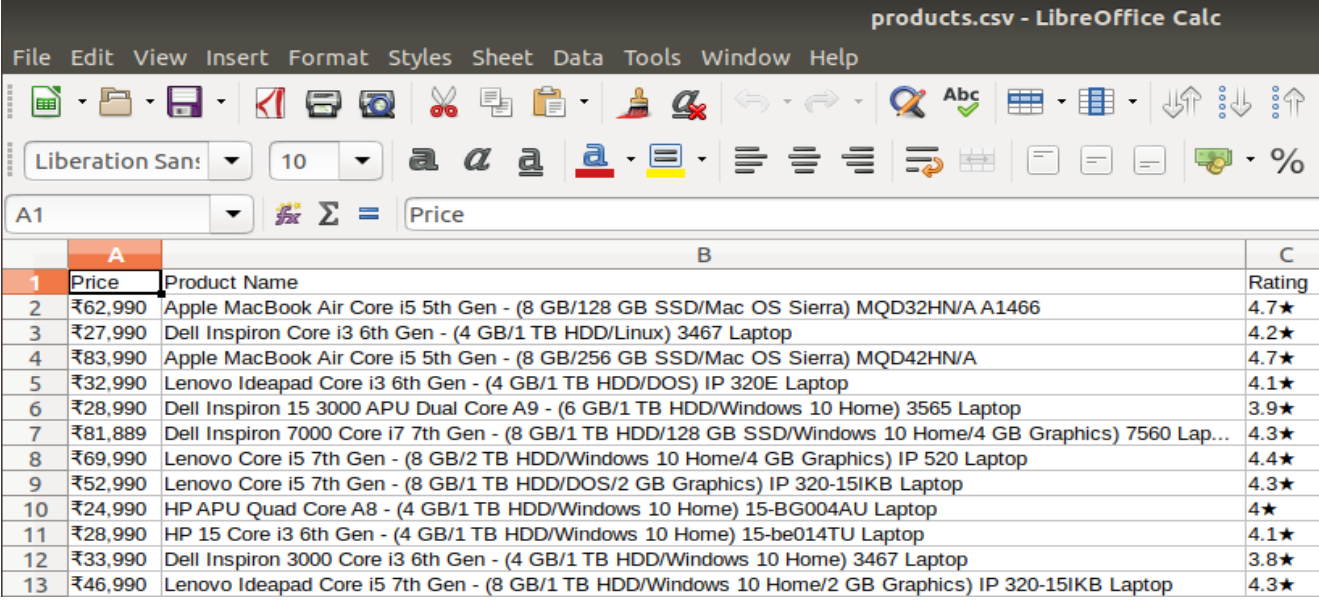
# Step 6

## 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, we can add the following lines to existing code:

```
1 df = pd.DataFrame({'Product Name':products,'Price':prices,'Rating':ratings})
2 df.to_csv('products.csv', index=False, encoding='utf-8')
```

- Now, we can run the whole code again.
- A file name “products.csv” is created and this file contains the extracted data.



	A	B	C
1	Price	Product Name	Rating
2	₹62,990	Apple MacBook Air Core i5 5th Gen - (8 GB/128 GB SSD/Mac OS Sierra) MQD32HN/A A1466	4.7★
3	₹27,990	Dell Inspiron Core i3 6th Gen - (4 GB/1 TB HDD/Linux) 3467 Laptop	4.2★
4	₹83,990	Apple MacBook Air Core i5 5th Gen - (8 GB/256 GB SSD/Mac OS Sierra) MQD42HN/A	4.7★
5	₹32,990	Lenovo Ideapad Core i3 6th Gen - (4 GB/1 TB HDD/DOS) IP 320E Laptop	4.1★
6	₹28,990	Dell Inspiron 15 3000 APU Dual Core A9 - (6 GB/1 TB HDD/Windows 10 Home) 3565 Laptop	3.9★
7	₹81,889	Dell Inspiron 7000 Core i7 7th Gen - (8 GB/1 TB HDD/128 GB SSD/Windows 10 Home/4 GB Graphics) 7560 Lap...	4.3★
8	₹69,990	Lenovo Core i5 7th Gen - (8 GB/2 TB HDD/Windows 10 Home/4 GB Graphics) IP 520 Laptop	4.4★
9	₹52,990	Lenovo Core i5 7th Gen - (8 GB/1 TB HDD/DOS/2 GB Graphics) IP 320-15IKB Laptop	4.3★
10	₹24,990	HP APU Quad Core A8 - (4 GB/1 TB HDD/Windows 10 Home) 15-BG004AU Laptop	4★
11	₹28,990	HP 15 Core i3 6th Gen - (4 GB/1 TB HDD/Windows 10 Home) 15-be014TU Laptop	4.1★
12	₹33,990	Dell Inspiron 3000 Core i3 6th Gen - (4 GB/1 TB HDD/Windows 10 Home) 3467 Laptop	3.8★
13	₹46,990	Lenovo Ideapad Core i5 7th Gen - (8 GB/1 TB HDD/Windows 10 Home/2 GB Graphics) IP 320-15IKB Laptop	4.3★