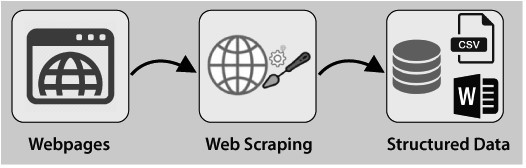




Web Scraping is a technique to extract a large amount of data from several websites. The term **"scraping"** refers to obtaining the information from another source (webpages) and saving it into a local file. For example: Suppose you are working on a project called **"Phone comparing website,"** where you require the price of mobile phones, ratings, and model names to make comparisons between the different mobile phones. If you collect these details by checking various sites, it will take much time. In that case, web scrapping plays an important role where by writing a few lines of code you can get the desired results.



Web Scrapping extracts the data from websites in the unstructured format. It helps to collect these unstructured data and convert it in a structured form.

Startups prefer web scrapping because it is a cheap and effective way to get a large amount of data without any partnership with the data selling company.



Here the question arises **whether the web scrapping is legal or not**. The answer is that some sites allow it when used legally. Web scraping is just a tool you can use it in the right way or wrong way

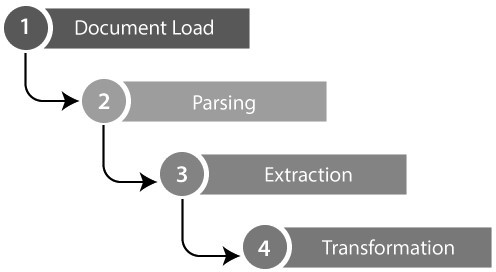
Web scrapping is illegal if someone tries to scrap the nonpublic data. Nonpublic data is not reachable to everyone; if you try to extract such data then it is a violation of the legal term.

There are several tools available to scrap data from websites, such as:

* Scrapping-bot
* Scrapper API
* Octoparse
* Import.io
* Webhose.io
* Dexi.io
* Outwit
* Diffbot
* Content Grabber
* Mozenda
* Web Scrapper Chrome Extension









There are other popular programming languages, but why we choose the Python over other programming languages for web scraping? Below we are describing a list of Python's features that make the most useful programming language for web scrapping.



In Python, we don't need to define data types for variables; we can directly use the variable wherever it requires. It saves time and makes a task faster. Python defines its classes to identify the data type of variable.



Python comes with an extensive range of libraries such as **NumPy, Matplotlib, Pandas, Scipy, etc**., that provide flexibility to work with various purposes. It is suited for almost every emerging field and also for web scrapping for extracting data and do manipulation.



The purpose of the web scrapping is to save time. But what if you spend more time in writing the code? That's why we use Python, as it can perform a task in a few lines of code.





The web scrapping consists of two parts: **a web crawler and a web scraper**. In simple words, the web crawler is a horse, and the scrapper is the chariot. The crawler leads the scrapper and extracts the requested data. Let's understand about these two components of web scrapping:

# The crawler





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

First, you should understand the requirement of data according to your project. A webpage or website contains a large amount of information. That's why scrap only relevant information. In simple words, the developer should be familiar with the data requirement.

# Step - 2: Inspecting the Page

The data is extracted in raw HTML format, which must be carefully parsed and reduce the noise from the raw data. In some cases, data can be simple as name and address or as complex as high dimensional weather and stock market data.

# Step - 3: Write the code

Write a code to extract the information, provide relevant information, and run the code.

# Step - 4: Store the data in the file

Store that information in required csv, xml, JSON file format.





Python has a vast collection of libraries and also provides a very useful library for web scrapping. Let's understand the required library for Python.

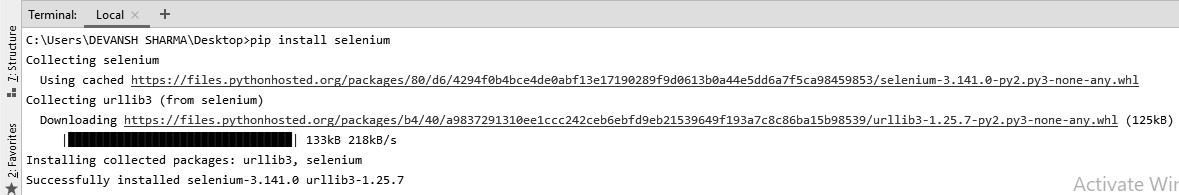
# Library used for web scrapping





pip install selenium





# Pandas

Pandas library is used for **data manipulation and analysis**. It is used to extract the data and store it in the desired format.

# BeautifulSoup

BeautifulSoup is a Python library that is used to pull data of HTML and XML files. It is mainly designed for web scrapping. It works with the parser to provide a natural way of navigating, searching, and modifying the parse tree. The latest version of BeautifulSoup is 4.8.1.

# Installation of BeautifulSoup

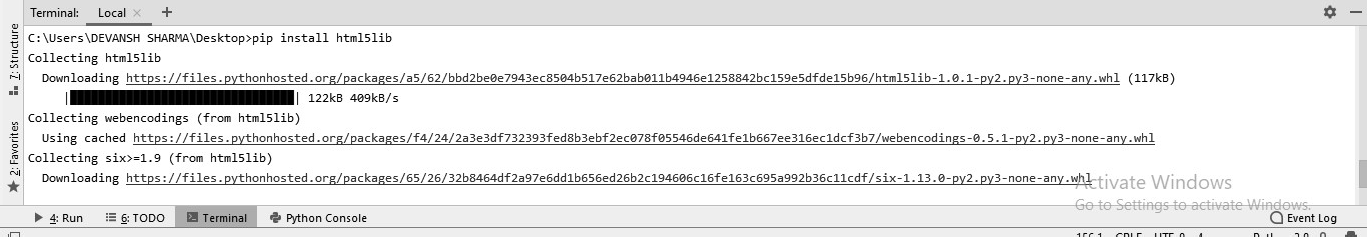
You can install BeautifulSoup by typing the following command: pip install bs4

# Installing a parser

BeautifulSoup supports HTML parser and several third-party Python parsers. You can install any of them according to your dependency. The list of BeautifulSoup's parsers is the following:

|  |  |
| --- | --- |
| **Parser** | **Typical usage** |
| Python's html.parser | BeautifulSoup(markup,"html.parser") |
| lxml's HTML parser | BeautifulSoup(markup,"lxml") |
| lxml's XML parser | BeautifulSoup(markup,"lxml-xml") |
| Html5lib | BeautifulSoup(markup,"html5lib") |

Type the following command in your terminal: pip install html5lib



BeautifulSoup is used to transform a complex HTML document into a complex tree of Python objects. But there are a few essential types object which are mostly used:

# Tag

A **Tag** object corresponds to an XML or HTML original document.

soup = bs4.BeautifulSoup("<b class = "boldest">Extremely bold</b>) tag = soup.b

type(tag)

# Output:

<class "bs4.element.Tag">

Tag contains lot of attributes and methods, but most important features of a tag are name and attribute.

# Name

Every tag has a name, accessible as **.name:**

1. tag.name

# Attributes

A tag may have any number of attributes. The tag <b id = "boldest"> has an attribute "id" whose value is "boldest". We can access a tag's attributes by treating the tag as dictionary.

tag[id]

We can add, remove, and modify a tag's attributes. It can be done by using tag as dictionary.

# add the element tag['id'] = 'verybold'

tag['another-attribute'] = 1 tag

# delete the tag del tag['id']

# Multi-valued Attributes

In HTML5, there are some attributes that can have multiple values. The class (consists more than one css) is the most common multivalued attributes. Other attributes are **rel, rev, accept-charset, headers,** and **accesskey**.

class\_is\_multi= { '\*' : 'class'}

xml\_soup = BeautifulSoup('<p class="body strikeout"></p>', 'xml', multi\_valu ed\_attributes=class\_is\_multi)

xml\_soup.p['class']

# [u'body', u'strikeout']

# NavigableString

A string in BeautifulSoup refers text within a tag. BeautifulSoup uses the **NavigableString** class to contain these bits of text.

tag.string

# u'Extremely bold' type(tag.string)

# <**class** 'bs4.element.NavigableString'>

A string is immutable means it can't be edited. But it can be replaced with another string using **replace\_with()**.

tag.string.replace\_with("No longer bold") tag

In some cases, if you want to use a **NavigableString** outside the BeautifulSoup, the **unicode()** helps it to turn into normal Python Unicode string.

# BeautifulSoup object

The BeautifulSoup object represents the complete parsed document as a whole. In many cases, we can use it as a Tag object. It means it supports most of the methods described in navigating the tree and searching the tree.

doc=BeautifulSoup("**<document><content/>**INSERT FOOTER HERE**</docu ment**","xml")

footer=BeautifulSoup("**<footer>**Here's the footer**</footer>**","xml") doc.find(text="INSERT FOOTER HERE").replace\_with(footer)

print(doc)

# Output:

?xml version="1.0" encoding="utf-8"?>

# <document><content/><footer>Here's the footer</footer></document>



1. #importing the BeautifulSoup Library 2.

1. importbs4
2. **import** requests
3. #Creating the requests 7.
4. res = requests.get("https://en.wikipedia.org/wiki/Machine\_learning")
5. print("The object type:",type(res)) 10.
6. # Convert the request object to the Beautiful Soup Object
7. soup = bs4.BeautifulSoup(res.text,'html5lib')
8. print("The object type:",type(soup)



The object type <class 'requests.models.Response'> Convert the object into: <class 'bs4.BeautifulSoup'>

In the following lines of code, we are extracting all headings of a webpage by class name. Here front-end knowledge plays an essential role in inspecting the webpage.

1. soup.select('.mw-headline')
2. **for** i in soup.select('.mw-headline'):
3. print(i.text,end = ',')

**Output:**

Overview,Machine learning tasks,History and relationships to other fields,Relation to data mining



