Implementing Web Scraping in Python with BeautifulSoup

Difficulty Level : Medium • Last Updated : 15 May, 2021

There are mainly two ways to extract data from a website:

- Use the API of the website (if it exists). For example, Facebook has the Facebook Graph API which allows retrieval of data posted on Facebook.
- Access the HTML of the webpage and extract useful information/data from it. This technique is called web scraping or web harvesting or web data extraction.

This article discusses the steps involved in web scraping using the implementation of a Web Scraping framework of Python called Beautiful Soup.

Steps involved in web scraping:

1. Send an HTTP request to the URL of the webpage you want to access. The server responds to the request by returning the HTML content of the webpage. For this task, we will use a third-party HTTP library for python-requests.
2. Once we have accessed the HTML content, we are left with the task of parsing the data. Since most of the HTML data is nested, we cannot extract data simply through string processing. One needs a parser which can create a nested/tree structure of the HTML data. There are many HTML parser libraries available but the most advanced one is html5lib.
3. Now, all we need to do is navigating and searching the parse tree that we created, i.e. tree traversal. For this task, we will be using another third-party python library, Beautiful Soup. It is a Python library for pulling data out of HTML and XML files.

Step 1: Installing the required third-party libraries

```
pip install requests
pip install html5lib
pip install bs4
```

- Another way is to download them manually from these links:
  - requests
  - html5lib
  - beautifulsoup4

Step 2: Accessing the HTML content from webpage

```
import requests
URL = "https://www.geeksforgeeks.org/data-structures/"
r = requests.get(URL)
print(r.content)
```

Let us try to understand this piece of code.

- First of all import the requests library.
- Then, specify the URL of the webpage you want to scrape.
- Send a HTTP request to the specified URL and save the response from server in a response object called r.
- Now, as print r.content to get the **raw HTML content** of the webpage. It is of 'string' type.

Step 3: Parsing the HTML content

```
#This will not run on online IDE
import requests
from bs4 import BeautifulSoup

URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)
```

A really nice thing about the BeautifulSoup library is that it is built on the top of the HTML parsing libraries like html5lib, lxml, html.parser, etc. So BeautifulSoup object and specify the parser library can be created at the same time.

In the example above,

```
soup = BeautifulSoup(r.content, 'html5lib')
```

We create a BeautifulSoup object by passing two arguments:

- **r.content** : It is the raw HTML content.
- **html5lib** : Specifying the HTML parser we want to use.

Now **soup.prettify()** is printed, it gives the visual representation of the parse tree created from the raw HTML content.

Step 4: Searching and navigating through the parse tree

Now, we would like to extract some useful data from the HTML content. The soup object contains all the data in the nested structure which could be programmatically extracted. In our example, we are scraping a webpage consisting of some quotes. So, we would like to create a program to save those quotes (and all relevant information about them).

```
#Python program to scrape website
#and save quotes from website
import requests
from bs4 import BeautifulSoup
import csv

URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

quotes=[]  # a list to store quotes

table = soup.find('div', attrs = {'id':'all_quotes'})
```

```
    quote['lines'] = row.img['alt'].split(" #")[0]
    quote['author'] = row.img['alt'].split(" #")[1]
    quotes.append(quote)

filename = 'inspirational_quotes.csv'
with open(filename, 'w', newline='') as f:
    w = csv.DictWriter(f,['theme','url','img','lines','author'])
    w.writeheader()
    for quote in quotes:
        w.writerow(quote)
```

Before moving on, we recommend you to go through the HTML content of the webpage which we printed using soup.prettify() method and try to find a pattern or a way to navigate to the quotes.

- It is noticed that all the quotes are inside a div container whose id is 'all_quotes'. So, we find that div element (termed as table in above code) using **find()** method :

  ```
  table = soup.find('div', attrs = {'id':'all_quotes'})
  ```

  The first argument is the HTML tag you want to search and second argument is a dictionary type element to specify the additional attributes associated with that tag. **find()** method returns the first matching element. You can try to print **table.prettify()** to get a sense of what this piece of code does.

- Now, in the table element, one can notice that each quote is inside a div container whose class is quote. So, we iterate through each div container whose class is quote. Here, we use findAll() method which is similar to find method in terms of arguments but it returns a list of all matching elements. Each quote is now iterated using a variable called **row.**
  Here is one sample row HTML content for better understanding:

Now consider this piece of code:

```
for row in table.find_all_next('div', attrs = {'class': 'col-6 col-lg-3 t
    quote = {}
    quote['theme'] = row.h5.text
    quote['url'] = row.a['href']
    quote['img'] = row.img['src']
    quote['lines'] = row.img['alt'].split(" #")[0]
    quote['author'] = row.img['alt'].split(" #")[1]
    quotes.append(quote)
```

We create a dictionary to save all information about a quote. The nested structure can be accessed using dot notation. To access the text inside an HTML element, we use **.text :**

```
quote['theme'] = row.h5.text
```

We can add, remove, modify and access a tag's attributes. This is done by treating the tag as a dictionary:

```
quote['url'] = row.a['href']
```

Lastly, all the quotes are appended to the list called **quotes.**

- Finally, we would like to save all our data in some CSV file.

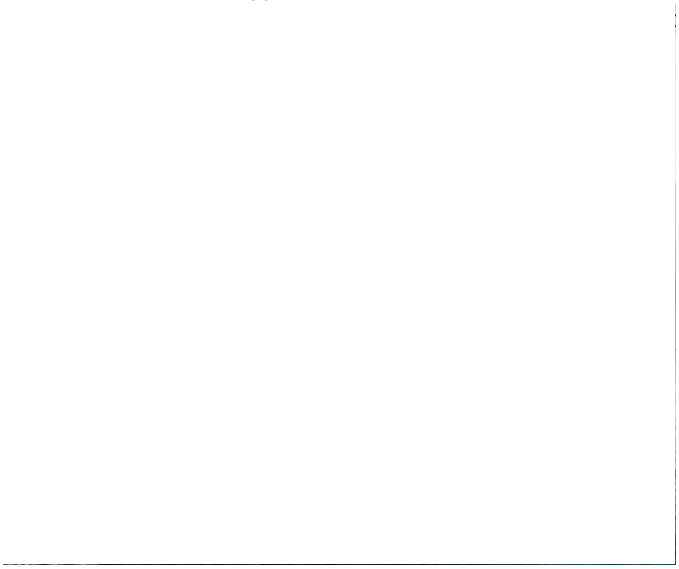
```
filename = 'inspirational_quotes.csv'
with open(filename, 'w', newline='') as f:
    w = csv.DictWriter(f,['theme','url','img','lines','author'])
    w.writeheader()
    for quote in quotes:
        w.writerow(quote)
```

can try to scrap any other website of your choice. In case of any queries, post them below in comments section.

Note : Web Scraping is considered as illegal in many cases. It may also cause your IP to be blocked permanently by a website.

This blog is contributed by **Nikhil Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



RECOMMENDED ARTICLES

Page : 1 2 3

01 **Web Scraping using Beautifulsoup and scrapingdog API**
19, Aug 20

02 **Implementing web scraping using lxml in Python**
07, Mar 18

03 **Implementing Web Scraping in Python with Scrapy**
08, May 18

04 **Scraping Reddit with Python and BeautifulSoup**
27, May 21

05 **BeautifulSoup - Scraping List from HTML**
22, Jan 21

06 **BeautifulSoup - Scraping Paragraphs from HTML**
22, Jan 21

07 **Scraping Covid-19 statistics using BeautifulSoup**
07, May 20

08 **BeautifulSoup - Scraping Link from HTML**
21, Jan 21

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : the_galaxy_hunter, Shamiul Hasan

Article Tags : GBlog, Project, Python

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments