**What is Beautiful Soup?**

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.

**Installing Beautiful Soup**

Installing Beautiful Soup is easiest if you have pip. Run the following command in the terminal to install Beautiful Soup:

pip install beautifulsoup4

Additionally, you will need to install a “parser” for interpreting the HTML. To do so, run in the terminal:

pip install lxml.

**Accessing the HTML content from webpage**

# Python Requests get() Method

## Syntax

requests.get(url, params={key: value}, args)

*args* means zero or more of the *named* arguments in the parameter table below. Example:

requests.get(url, timeout=2.50)

### Example

Make a request to a web page, and return the status code:

import requests  
  
x = requests.get('https://w3schools.com')  
print(x.status\_code)

## Return Value

The get() method returns a [requests Response object](https://www.w3schools.com/python/ref_requests_response.asp).

|  |  |
| --- | --- |
| 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") BeautifulSoup(markup, "xml") |
| html5lib | BeautifulSoup(markup, "html5lib") |

To parse a document, pass it into the BeautifulSoup constructor. You can pass in a string or an open filehandle:

### Attributes

A tag may have any number of attributes. The tag <b id="boldest"> has an attribute “id” whose value is “boldest”. You can access a tag’s attributes by treating the tag like a dictionary:

tag['id']

# u'boldest'

You can access that dictionary directly as .attrs:

tag.attrs

# {u'id': 'boldest'}

You can add, remove, and modify a tag’s attributes. Again, this is done by treating the tag as a dictionary:

tag['id'] = 'verybold'

tag['another-attribute'] = 1

tag

# <b another-attribute="1" id="verybold"></b>

del tag['id']

del tag['another-attribute']

tag

# <b></b>

tag['id']

# KeyError: 'id'

print(tag.get('id'))

# None

## BeautifulSoup

The BeautifulSoup object represents the parsed document as a whole. For most purposes, you can treat it as a [Tag](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#tag) object. This means it supports most of the methods described in [Navigating the tree](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#navigating-the-tree) and [Searching the tree](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#searching-the-tree).

You can also pass a BeautifulSoup object into one of the methods defined in [Modifying the tree](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#modifying-the-tree), just as you would a [Tag](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#tag). This lets you do things like combine two parsed documents:

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

footer = BeautifulSoup("<footer>Here's the footer</footer>", "xml")

doc.find(text="INSERT FOOTER HERE").replace\_with(footer)

# u'INSERT FOOTER HERE'

print(doc)

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

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

Since the BeautifulSoup object doesn’t correspond to an actual HTML or XML tag, it has no name and no attributes.

**What is** [**Selenium**](http://www.geeksforgeeks.org/browser-automation-using-selenium/) **?**

[**Selenium**](http://www.geeksforgeeks.org/browser-automation-using-selenium/): Selenium Python bindings provide a convenient API to access Selenium Web Driver like Firefox, Chrome, etc. Selenium WebDriver is one of the most popular tools for Web UI Automation.

Web UI Automation means the automatic execution of the actions performed in a web browser window like navigating to a website, filling forms that include dealing with text boxes, radio buttons and drop downs, submitting the forms, browsing through web pages, handling pop-ups and so on. Selenium WebDriver is the one that can automate all these tasks. It can interact with all types of Web browsers available till date like Firefox, Internet Explorer, Safari, and Chrome, etc.

**What is webdriver?**  
Selenium WebDriver is an automation testing tool. When I say automation, it means it automates test scripts written in Selenium.

**Library Imported**

from selenium import webdriver

import time

(i) Selenium library:  
– Used for Automation  
– Control Webdriver  
– Perform actions like – element clicks, refresh page, goto website link, etc

(ii) Time library:  
-For using sleep function because selenium works only when the all the elements of the page is loaded.

**Driver setup:**  
**I am using Firefox driver:**  
firefoxdriver = webdriver.Firefox(executable\_path=”Path to Firefox driver”)

In the example, we used the *.find\_element\_by\_name()* method, which searches for the attribute name within the input HTML tag. We can also search for this term using other methods.

* CSS ID: *.find\_element\_by\_id(“id-search-field”)*
* DOM Path: *.find\_element\_by\_xpath(“//input[@id=’id-search-field’]”)*
* CSS class: *.find\_element\_by\_class\_name(“search-field”)*