**Web Scrapping**

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
3. Beautiful Soup sits on top of popular Python parsers like [lxml](http://lxml.de/) and [html5lib](http://code.google.com/p/html5lib/), allowing you to try out different parsing strategies or trade speed for flexibility.

Request library: <http://docs.python-requests.org/en/master/>

**Requests** allow you to send organic, grass-fed HTTP/1.1 requests, without the need for manual labor. There’s no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic

http://docs.python-requests.org/en/master/api/

requests.**get**(url, params=None, \*\*kwargs)

Sends a GET request.

**status\_code***= None*

Integer Code of responded HTTP Status, e.g. 404 or 200.

**text**

Content of the response, in unicode. If Response.encoding is None, ncoding will be guessed using chardet. The encoding of the response content is determined based solely on HTTP headers, following RFC 2616 to the letter. If you can take advantage of non-HTTP knowledge to make a better guess at the encoding, you should set r.encoding appropriately before accessing this property.

**content**

Content of the response, in bytes.

The prettify() method will turn a Beautiful Soup parse tree into a nicely formatted Unicode string, with a separate line for each HTML/XML tag and string:

The BeautifulSoup object itself has children. In this case, the <html> tag is the child of the BeautifulSoup object.:

len(soup.contents)

*# 1*

soup.contents[0].name

*# u'html'*

get\_text()

If you only want the text part of a document or tag, you can use the get\_text() method. It returns all the text in a document or beneath a tag, as a single Unicode string.

## find\_all()

find\_all() is the most popular method in the Beautiful Soup search API, you can use a shortcut for it. If you treat the BeautifulSoup object or a Tag object as though it were a function, then it’s the same as calling find\_all() on that object.

## find()

find([name](https://www.crummy.com/software/BeautifulSoup/bs4/doc/" \l "id11), [attrs](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#attrs), [recursive](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#recursive), [string](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#id12), [\*\*kwargs](https://www.crummy.com/software/BeautifulSoup/bs4/doc/#kwargs))

The find\_all() method scans the entire document looking for results, but sometimes you only want to find one result. If you know a document only has one <body> tag, it’s a waste of time to scan the entire document looking for more. Rather than passing in limit=1 every time you call find\_all, you can use the find() method.