## BeautifulSoup

One of the most popular HTML parsers for Python is called **BeautifulSoup**. It’s been around for quite some time and is known for being able to handle malformed HTML well. To install it for Python 3, all you need to do is the following:

```
pip install beautifulsoup4
```

If everything worked correctly, you should now have BeautifulSoup installed. When passing BeautifulSoup some HTML to parse, you can specify a tree builder. For this example we will use **html.parser**, because it is included with Python. If you’d like something faster, you can install lxml.

Let’s actually take a look at some code to see how this all works:

```
import requests
from bs4 import BeautifulSoup

url = 'http://www.blog.pythonlibrary.org/'

def get_articles():
    """
    Get the articles from the front page of the blog
    """
    req = requests.get(url)
    html = req.text
    soup = BeautifulSoup(html, 'html.parser')
    pages = soup.findAll('h1')

    articles = {i.a['href']: i.text.strip()
                for i in pages if i.a}

    for article in articles:
        s = '{title}: {url}'.format(title=articles[article].encode('utf-8'),url=article)
        print(s)

    return articles

if __name__ == '__main__':
    articles = get_articles()
```

Here we do out imports and set up what URL we are going to use. Then we create a function where the magic happens. We use the requests library to get the URL and then pull the HTML out as a string using the request object's **text** property. Then we pass the HTML to BeautifulSoup which turns it into a nice object. After that, we ask BeautifulSoup to find all the instances of **h1** and then use a dictionary comprehension to extract the title and URL. We then print that information to stdout and return the dictionary.

Let's try to scrape another website. This time we will look at Twitter and use my blog's account: mousevspython. We will try to scrape what I have tweeted recently. You will need to follow the same steps as before by right-clicking on a tweet and inspecting it to figure out what we need to do. In this case, we need to look for the 'li' tag and the js-stream-item class. Let's take a look:

```
import requests

from bs4 import BeautifulSoup

url = 'https://twitter.com/mousevspython'
req = requests.get(url)
html = req.text
soup = BeautifulSoup(html, 'html.parser')
tweets = soup.findAll('li', 'js-stream-item')
for item in range(len(soup.find_all('p', 'TweetTextSize'))):
    tweet_text = tweets[item].get_text().encode('utf-8')
    print(tweet_text)
    dt = tweets[item].find('a', 'tweet-timestamp')
    print('This was tweeted on ' + str(dt))
```

As before, we use BeautifulSoup's **findAll** command to grab all the instances that match our search criteria. Then we also look for the paragraph tag (i.e. 'p') and the 'TweetTextSize' class and loop over the results. You will note that we used **find_all** here. Just so we're clear, findAll is an alias of find_all, so they do the exact same thing. Anyway, we loop over those results and grab the tweet text and the tweet timestamp and print them out.

You would think that there might be an easier way to do this sort of thing and there is. Some websites provide a developer API that you can use to access their website's data. Twitter has a nice one that requires a consumer key and a secret. We will actually be looking at how to use that API and a couple of others in the next chapter.

Let’s move on and learn how to write a spider!