Testing BeautifulSoup - Report

Erik Bertse
Sara Gustavsson
Moa Marklund
Henrik Thorsell

December 7, 2017

Abstract

The BeautifulSoup library is a Python library designed to easily search and manipulate data associated with HTML/XML markup. In this project, we test the functionality of this library. Most of the available API functions have been subject to black-box testing. One of the functions was selected to undergo white-box testing. Our results are: Here will be the results.

1 Introduction

1.1 What is BeautifulSoup?

BeatifulSoup is a library for Python which provides functionality for navigating and extracting data from HTML and XML markup. It does so by using external parsers, specifiable by the users of the library, whose output is assembled by BeautifulSoup to generate a tree structure, the nodes of which are tags in the HTML/XML markup. Thus, children of any given tag are nested tags.

The BeautifulSoup library is compatible with Python 2.7 and Python 3.2, but we have limited the scope of our testing to using version 2.7. The most recent version of BeautifulSoup is 4, abbreviated BS4. Previous (and no longer unsupported) versions of BeautifulSoup are available, but are not subject to any tests in this project.

Given some markup, the library creates an ‘soup’ object, containing all the information in the markup. In the following example, we call the BeautifulSoup constructor with some markup, using the HTML parser that comes with Python:

```
a_simple_soup = BeautifulSoup(
    '''
    <html>
      <head>
      </head>
      <body>
        <p>
          <a>An anchor tag</a>
        </p>
      </body>
    </html>
    ''', "html.parser")
```

The soup object created is a rather complex. Each tag in the markup gives rise to an object inside the soup, simply referred to as *Tag* object. the *Tag* objects are named according to the name of the tag. These are then organized in a tree structure, such that any inner tag as a child of its parent tag. We can access these objects directly. For example, we can access the `head` object inside the soup show above like so:

```
a_simple_soup.head
```

Note that we are not required to first access its parent, i.e. we do not have to specify

```
a_simple_soup.html.head
```

although this is perfectly valid also. If the tree structure contains several tags with the same name, then the first one in markup will be accessed.

The library contains a number of functions for navigating and modifying the tree, many of which were tested in this project. For example, the following call:

```
a_simple_soup.find_all("a")
```

will return a list of all *Tag* objects it finds in the tree structure.

Many other functions are available. We refer the curious reader to the BeautifulSoup documentation, for a complete list. The documentation is available here: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`

2 Method

This is a section describing what we did. This includes how we wrote the tests, what tools we used, and how we structured the code. How is the implementation of the black-box different than the white-box? why?

2.1 Blackbox testing

This section is details the black-box testing

2.1.1 Navigation

Navigating the soup tree by directly accessing the subojects of the soup, (i.e. soup.html.title)

2.1.2 Search

Searching the tree (i.e. find, find_all, find_parent)

2.1.3 Insert/Delete/Edit

Editing the tree (i.e. insert, append, wrap)

2.2 Whitebox testing

Here all our white box will go.

3 Results

This is a section that presents what we found out. Did we find bugs in the library?

4 Discussion

This section discusses the results. What can we infer from the tests? What did we learn? What did we omit? Why did we pick the tests we picked?

5 Conclusion

This section is a summary of the project

6 Appendix

Here we insert all our code