Basic Webpage Scraping

Webpage scraping consists of two steps: crawling and parsing. In this tutorial, we focus on parsing HTML data. Beautifulsoup is a powerful tool to process static HTML. More details can be found at https://www.crummy.com/software/BeautifulSoup/bs4/doc/

To simplify our learning, we will use a simple example from W3Schools: https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_example_website

This simple example contains in a single HTML for simplicity and has been saved in an html file, simple_page.html.

Parsing a webpage

First we read the content from simple_page.html file, store content in variable 'html'

```
with open('simple_page.html') as f:
    html = f.read()
```

```
html
```

```
'<!DOCTYPE html>\n<html lang="en">\n<head>\n<title>Page
Title</title>\n<meta charset="UTF-8">\n<meta name="viewport"
content="width=device-width, initial-scale=1">\n<style>\n* {\n  box-
sizing: border-box;\n}\n\n/* Style the body */\nbody {\n  font-family:
Arial, Helvetica, sans-serif;\n  margin: 0;\n}\n\n/* Header/logo Title
*/\n.header {\n  padding: 80px;\n  text-align: center;\n  background:
#1abc9c;\n  color: white;\n}\n\n/* Increase the font size of the heading
*/\n.header h1 {\n  font-size: 40px;\n}\n\n/* Sticky navbar - toggles
between relative and fixed, depending on the scroll position. It is
positioned relative until a given offset position is met in the viewport -
then it "sticks" in place (like position:fixed). The sticky value is not
supported in IE or Edge 15 and earlier versions. However, for these
versions the navbar will inherit default position */\n.navbar {\n
overflow: hidden;\n  background-color: #333;\n  position: sticky;\n
position: -webkit-sticky;\n  top: 0;\n}\n\n/* Style the navigation bar
links */\n.navbar a {\n  float: left;\n  display: block;\n  color:
white;\n  text-align: center;\n  padding: 14px 20px;\n  text-decoration:
none;\n}\n\n\n/* Right-aligned link */\n.navbar a.right {\n  float:
right;\n}\n\n/* Change color on hover */\n.navbar a:hover {\n  background-
color: #ddd;\n  color: black;\n}\n\n/* Active/current link */\n.navbar
a.active {\n  background-color: #666;\n  color: white;\n}\n\n/* Column
container */\n.row {  \n  display: -ms-flexbox; /* IE10 */\n  display:
```

```
flex;\n  -ms-flex-wrap: wrap; /* IE10 */\n  flex-wrap: wrap;\n}\n\n/*
Create two unequal columns that sits next to each other */\n/*
Sidebar/left column */\n.side {\n  -ms-flex: 30%; /* IE10 */\n  flex:
30%;\n  background-color: #f1f1f1;\n  padding: 20px;\n}\n\n/* Main column
*/\n.main {   \n  -ms-flex: 70%; /* IE10 */\n  flex: 70%;\n  background-
color: white;\n  padding: 20px;\n}\n\n/* Fake image, just for this example
*/\n.fakeimg {\n  background-color: #aaa;\n  width: 100%;\n  padding:
20px;\n}\n\n/* Footer */\n.footer {\n  padding: 20px;\n  text-align:
center;\n  background: #ddd;\n}\n\n/* Responsive layout - when the screen
is less than 700px wide, make the two columns stack on top of each other
instead of next to each other */\n@media screen and (max-width: 700px) {\n
.row {   \n    flex-direction: column;\n  }\n}\n\n/* Responsive layout -
when the screen is less than 400px wide, make the navigation links stack
on top of each other instead of next to each other */\n@media screen and
(max-width: 400px) {\n  .navbar a {\n    float: none;\n    width: 100%;\n
}\n}\n</style>\n</head>\n<body>\n\n<div class="header">\n  <h1>My
Website</h1>\n  <p>A <b>responsive</b> website created by me.
</p>\n</div>\n\n<div class="navbar">\n  <a href="#"
class="active">Home</a>\n  <a href="#">Link</a>\n  <a href="#">Link</a>\n
<a href="#" class="right">Link</a>\n</div>\n\n<div class="row">\n  <div
class="side">\n    <h2>About Me</h2>\n    <h5>Photo of me:</h5>\n    <div
id="my_photo" class="fakeimg" style="height:200px;">Image</div>\n
<p>Some text about me in culpa qui officia deserunt mollit anim..</p>\n
<h3>More Text</h3>\n    <p>Lorem ipsum dolor sit ame.</p>\n    <div
class="fakeimg" style="height:60px;">Image</div><br>\n    <div
class="fakeimg" style="height:60px;">Image</div><br>\n    <div
class="fakeimg" style="height:60px;">Image</div>\n  </div>\n  <div
class="main" id="div_1">\n    <h2>TITLE HEADING</h2>\n    <h5>Title
description, Dec 7, 2017</h5>\n    <div class="fakeimg"
style="height:200px;">Image</div>\n    <p id="more_text">Some text..</p>\n
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>\n    <br>\n    <h2>TITLE HEADING</h2>\n    <h5>Title
description, Sep 2, 2017</h5>\n    <div class="fakeimg"
style="height:200px;">Image</div>\n    <p>Some text..</p>\n    <p>Sunt in
culpa qui officia deserunt mollit anim id est laborum consectetur
```

Import all necessary packages

```python
from bs4 import BeautifulSoup
from bs4.element import Tag
```

Parse HTML content in 'html' variable. Note that prettify is a bs4 method that returns a string with formatted HTML content.

```python
soup = BeautifulSoup(html, "lxml")
print(soup.prettify())
```

```
  * {
  box-sizing: border-box;
}

/* Style the body */
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #1abc9c;
  color: white;
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}

/* Sticky navbar - toggles between relative and fixed, depending on the scr
.navbar {
  overflow: hidden;
  background-color: #333;
  position: sticky;
  position: -webkit-sticky;
  top: 0;
}

/* Style the navigation bar links */
.navbar a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
}


/* Right-aligned link */
.navbar a.right {
  float: right;
}

/* Change color on hover */
.navbar a:hover {
  background-color: #ddd;
  color: black;
}
```

```
/* Active/current link */
.navbar a.active {
  background-color: #666;
  color: white;
```

Beautiful Soup DOM Tree

The structure of Beautiful Soup bases on the concept of DOM, which is used in all web browsers. DOM is a tree of all elements in the webpage. Each element node consists of:

- tag
- innerHTML/outerHTML
- id
- attributes
- parent and children

DOM Element Node

Tag **'title'** is a tag of one of the element node in the example; we can refer to the node by using the tag name

Note that title is not a root node. The root node is 'html'. We can access nodes via the root node. We can also refer to the tag directly. In this case, bs4 will return the first node with the refered tag.

```
type(soup.title)
```

```
bs4.element.Tag
```

```
# we can get tag of a node with 'name'
soup.title.name
```

```
'title'
```

```
# we can get outerHTML by converting node to string
str(soup.title)
```

```
'<title>Page Title</title>'
```

```
# direct reference to a node leads to the same result
soup.title
```

```
<title>Page Title</title>
```

```
# we can get innerHTML with 'string'
soup.title.string
```

```
'Page Title'
```

```
# we can get id with 'id' (it is empty in this example)
soup.title.id
```

```
# we can get attribute values with 'attrs'
soup.title.attrs
```

```
{}
```

```
# getting the parent node with 'parent'
soup.title.parent.name
```

```
'head'
```

```
# referring to children
soup.title.children
```

```
<list_iterator at 0x107a0b220>
```

Traversing simple HTML's DOM Tree

In our example, the structure is as followed:

```
html
+-- head
|   +-- title
|   +-- meta
|   +-- meta
|   +-- style
+-- body
    +-- div
    |   +-- h1
    |   +-- p
    |       +-- b
    +-- div
    |   +-- a
    |   +-- a
    |   +-- a
    |   +-- a
    +-- div
    |   +--div
    |   |   +-- h2
    |   |   +-- h5
    |   |   +-- ...
    |   +--div
    |       +-- h2
    |       +-- h5
    |       +-- ...
    +-- div
        +-- h2
```

Let refer to the first anchor (tag = a) in this structure. Notice that even if we have 4 anchors in this HTML, bs4 returns only the first one.

```
soup.a
```

```
<a class="active" href="#">Home</a>
```

Assign this node to a variable 'a_node'

```
a_node = soup.a
```

A node can have multiple (or 0) attributes. We can access all attributes using get method.

```
a_node.attrs
```

```
{'href': '#', 'class': ['active']}
```

```
print(a_node.get('href'))
print(a_node.get('class'))
```

```
#
['active']
```

Relative Reference

Let's navigate based on a_node using parent and children

```
a_parent = a_node.parent
a_parent
```

```
<div class="navbar">
<a class="active" href="#">Home</a>
<a href="#">Link</a>
<a href="#">Link</a>
<a class="right" href="#">Link</a>
</div>
```

```
a_parent.name
```

```
'div'
```

Note that bs4 turns any spaces in between elements into NavigableString

```
for node in a_parent.children:
    print('[{}] {}'.format(type(node), node))
```

```
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a class="active" href="#">Home</a>
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a href="#">Link</a>
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a href="#">Link</a>
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a class="right" href="#">Link</a>
[<class 'bs4.element.NavigableString'>]
```

```
for node in a_node.previous_siblings:
    print('[{}] {}'.format(type(node), node))
```

```
[<class 'bs4.element.NavigableString'>]
```

```
for node in a_node.next_siblings:
    print('[{}] {}'.format(type(node), node))
```

```
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a href="#">Link</a>
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a href="#">Link</a>
[<class 'bs4.element.NavigableString'>]

[<class 'bs4.element.Tag'>] <a class="right" href="#">Link</a>
[<class 'bs4.element.NavigableString'>]
```

Let's try with first div node in this HTML

```
soup.div
```

```
<div class="header">
<h1>My Website</h1>
<p>A <b>responsive</b> website created by me.</p>
</div>
```

```
soup.div.parent.name
```

```
'body'
```

Here is the list of DOM navigation:

- **node.children**: iterator for all children of a node
- **node.descendants**: iterator for all of a tag’s children, recursively: its direct children, the children of its direct children, and so on
- **node.parent**: parent of the existing node
- **node.parents**: iterator for all of an element’s parents to the root of the DOM tree
- **node.next_sibling / node.previous_sibling**: navigate between page elements that are on the same level of the DOM tree
- **node.next_element / node.previous_element**: navigate between page elements in the DOM tree, regardless of the level

Direct Reference

We can refer to any node via the root node, html, with dot separated. Refer to our DOM tree structure in the following examples.

```
soup.head
```

```
<head>
<title>Page Title</title>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<style>
* {
  box-sizing: border-box;
}

/* Style the body */
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #1abc9c;
  color: white;
```

```
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}

/* Sticky navbar - toggles between relative and fixed, depending on the
scroll position. It is positioned relative until a given offset position
is met in the viewport - then it "sticks" in place (like position:fixed).
The sticky value is not supported in IE or Edge 15 and earlier versions.
However, for these versions the navbar will inherit default position */
.navbar {
  overflow: hidden;
  background-color: #333;
  position: sticky;
  position: -webkit-sticky;
  top: 0;
}

/* Style the navigation bar links */
.navbar a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
}


/* Right-aligned link */
.navbar a.right {
  float: right;
}

/* Change color on hover */
.navbar a:hover {
```

We can access the css style at node:

html -> head -> style

Note that we do not have to include html in the reference

```
soup.head.style
```

```
<style>
* {
  box-sizing: border-box;
}
```

```
        /* Style the body */
        body {
          font-family: Arial, Helvetica, sans-serif;
          margin: 0;
        }

        /* Header/logo Title */
        .header {
          padding: 80px;
          text-align: center;
          background: #1abc9c;
          color: white;
        }

        /* Increase the font size of the heading */
        .header h1 {
          font-size: 40px;
        }

        /* Sticky navbar - toggles between relative and fixed, depending on the
        scroll position. It is positioned relative until a given offset position
        is met in the viewport - then it "sticks" in place (like position:fixed).
        The sticky value is not supported in IE or Edge 15 and earlier versions.
        However, for these versions the navbar will inherit default position */
        .navbar {
          overflow: hidden;
          background-color: #333;
          position: sticky;
          position: -webkit-sticky;
          top: 0;
        }

        /* Style the navigation bar links */
        .navbar a {
          float: left;
          display: block;
          color: white;
          text-align: center;
          padding: 14px 20px;
          text-decoration: none;
        }



        /* Right-aligned link */
        .navbar a.right {
          float: right;
        }

        /* Change color on hover */
        .navbar a:hover {
          background-color: #ddd;
          color: black;
        }
```

We do not have to include everythin in the path. Let access the first h2 node inside body.

```
soup.body.h2
```

```
<h2>About Me</h2>
```

```
soup.body.h2.parent
```

```
<div class="side">
<h2>About Me</h2>
<h5>Photo of me:</h5>
<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>
<p>Some text about me in culpa qui officia deserunt mollit anim..</p>
<h3>More Text</h3>
<p>Lorem ipsum dolor sit ame.</p>
<div class="fakeimg" style="height:60px;">Image</div><br/>
<div class="fakeimg" style="height:60px;">Image</div><br/>
<div class="fakeimg" style="height:60px;">Image</div>
</div>
```

We can jump back and forth between nodes, parents, and children

```
soup.body.h2.parent.p
```

```
<p>Some text about me in culpa qui officia deserunt mollit anim..</p>
```

Finding Nodes

We usually have more than one element with the same tag. We can get all those nodes using find_all method. Note that find method is also avaiable and will return the first node that match the criteria.

```
all_div = soup.find_all('div')
```

```
len(all_div)
```

```
12
```

```
n = 0
for div in all_div:
    print('-- {} --'.format(n))
    print(div)
    n += 1
```

```
-- 0 --
<div class="header">
<h1>My Website</h1>
<p>A <b>responsive</b> website created by me.</p>
</div>
-- 1 --
<div class="navbar">
<a class="active" href="#">Home</a>
<a href="#">Link</a>
<a href="#">Link</a>
<a class="right" href="#">Link</a>
</div>
-- 2 --
<div class="row">
<div class="side">
<h2>About Me</h2>
<h5>Photo of me:</h5>
<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>
<p>Some text about me in culpa qui officia deserunt mollit anim..</p>
<h3>More Text</h3>
<p>Lorem ipsum dolor sit ame.</p>
<div class="fakeimg" style="height:60px;">Image</div><br/>
<div class="fakeimg" style="height:60px;">Image</div><br/>
<div class="fakeimg" style="height:60px;">Image</div>
</div>
<div class="main" id="div_1">
<h2>TITLE HEADING</h2>
<h5>Title description, Dec 7, 2017</h5>
<div class="fakeimg" style="height:200px;">Image</div>
<p id="more_text">Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetu
<br/>
<h2>TITLE HEADING</h2>
<h5>Title description, Sep 2, 2017</h5>
<div class="fakeimg" style="height:200px;">Image</div>
<p>Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetu
</div>
</div>
-- 3 --
<div class="side">
<h2>About Me</h2>
<h5>Photo of me:</h5>
<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>
<p>Some text about me in culpa qui officia deserunt mollit anim..</p>
<h3>More Text</h3>
<p>Lorem ipsum dolor sit ame.</p>
<div class="fakeimg" style="height:60px;">Image</div><br/>
<div class="fakeimg" style="height:60px;">Image</div><br/>
```

```
<div class="fakeimg" style="height:60px;">Image</div>
</div>
-- 4 --
<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>
-- 5 --
<div class="fakeimg" style="height:60px;">Image</div>
-- 6 --
<div class="fakeimg" style="height:60px;">Image</div>
-- 7 --
<div class="fakeimg" style="height:60px;">Image</div>
```

```
div8 = all_div[8]
div8
```

```
<div class="main" id="div_1">
<h2>TITLE HEADING</h2>
<h5>Title description, Dec 7, 2017</h5>
<div class="fakeimg" style="height:200px;">Image</div>
<p id="more_text">Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>
<br/>
<h2>TITLE HEADING</h2>
<h5>Title description, Sep 2, 2017</h5>
<div class="fakeimg" style="height:200px;">Image</div>
<p>Some text..</p>
<p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>
</div>
```

```
div8.attrs
```

```
div8.get('class')
```

find_all method can be used at any node. This effectly limits the finding scope.

```
div8.find_all('div')
```

```
div8.find('div')
```

Find with Criteria

find and find_all methods accept criteria for searching. For example, we can find all nodes with specific id or with specific attributes.

```
soup.find_all(id='more_text')
```

```
[<p id="more_text">Some text..</p>]
```

```
soup.find_all(attrs={'class': 'fakeimg'})
```

```
[<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:200px;">Image</div>,
 <div class="fakeimg" style="height:200px;">Image</div>]
```

```
soup.find_all(attrs={'style': 'height:60px;'})
```

```
[<div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>]
```

```
soup.find_all('div', attrs={'class': 'main'})
```

```
[<div class="main" id="div_1">
 <h2>TITLE HEADING</h2>
 <h5>Title description, Dec 7, 2017</h5>
 <div class="fakeimg" style="height:200px;">Image</div>
 <p id="more_text">Some text..</p>
 <p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>
 <br/>
 <h2>TITLE HEADING</h2>
 <h5>Title description, Sep 2, 2017</h5>
 <div class="fakeimg" style="height:200px;">Image</div>
 <p>Some text..</p>
 <p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>
 </div>]
```

CSS Selector

CSS Selector is very powerful for node searching. We can search by tag name, id, class, and combination of criteria. The CSS Selector includes:

- **string**: select node with the specific *tag* e.g. div for node with tag 'div'
- **.class**: select node with the specific *class*
- **#id**: select node with the specific *id*
- **tag[attr]**: select node with the specific *tag* and *attr*

```
soup.select('p')
```

```
[<p>A <b>responsive</b> website created by me.</p>,
 <p>Some text about me in culpa qui officia deserunt mollit anim..</p>,
 <p>Lorem ipsum dolor sit ame.</p>,
 <p id="more_text">Some text..</p>,
 <p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>,
 <p>Some text..</p>,
 <p>Sunt in culpa qui officia deserunt mollit anim id est laborum
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco.</p>]
```

```
soup.select('h2')
```

```
[<h2>About Me</h2>,
 <h2>TITLE HEADING</h2>,
 <h2>TITLE HEADING</h2>,
 <h2>Footer</h2>]
```

```
soup.select('#more_text')
```

```
[<p id="more_text">Some text..</p>]
```

```
soup.select('.fakeimg')
```

```
[<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:60px;">Image</div>,
 <div class="fakeimg" style="height:200px;">Image</div>,
 <div class="fakeimg" style="height:200px;">Image</div>]
```

```
soup.select('#my_photo.fakeimg')
```

```
[<div class="fakeimg" id="my_photo" style="height:200px;">Image</div>]
```

```
node = soup.select('div div h2')
node
```

```
[<h2>About Me</h2>, <h2>TITLE HEADING</h2>, <h2>TITLE HEADING</h2>]
```