Create Soup and Search for Tag

At step 3, after have HTML content downloaded with selenium, we use `beautiful soup` to parse the HTML and start search for tags which contain our wanted data.

Install Beautiful Soup

We need to install `beautiful soup` package before using it.

```
pip install beautifulsoup4
```

Create Soup Object

For easy of explain, we will use a predefine HTML content as following code. When we do actual web scraping, step 2 will provide HTML content with `driver.page_source`

```
# import beautifulsoup object
from bs4 import BeautifulSoup

# predefine a html content
html = """
<html>
<head>
    <title>The Dormouse's story</title>
</head>
<body>
    <p class="title">
        <b>The Dormouse's story</b>
    </p>
    <p class="story">
        Once upon a time there were three little sisters. And their names were
        <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
        <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a>
and
        <a href="http://example.com/tillie" class="sister"
id="link3">Tillie</a>;
        and they lived at the bottom of a well.
    </p>
    <p class="story">...</p>
/</body>
</html>
"""
# create soup object by HTML content
soup = BeautifulSoup(html, 'lxml')
```

Search for First Tag by Name

To search the first tag by name, we use function `find` and pass in the tag name. For example following code return the first `p` tag

```
first_p_tag = soup.find('p')
print(first_p_tag)
```

Result print out

```
"""
<p class="title">
    <b>The Dormouse's story</b>
</p>
"""
```

Search for All Tags by Name

To search for all tags by name, we use function `find_all` then pass in the tag name. For example following code return all `a` tags in a list.

```
a_tags = soup.find_all('a')
print(a_tags)
```

Result when print out

```
"""
[
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
]
"""
```

Search with Tag Name and Class Attribute

In real project, many time we want to search for a tag with specific class name. To do that we put in tag name and `class_` parameter.

```
p_tag = soup.find('p', class_ = 'story')
print(p_tag)
```

Above code will return the first `p` tag which have `class` is "story"

```
"""
<p class="story">
    Once upon a time there were three little sisters; and their names were
    <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
    <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
    <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
    and they lived at the bottom of a well.
</p>
"""
```

Search with Tag Name and Other Attribute

Beside of using `class` attribute for search, we could use any other attribute. To do that we put in tag name and attribute parameter with grammar `{'attribute_name': attribute_value}`

For example following code will find first `a` tag which have `id` is "link1"

```python
a = soup.find('a', {'id':'link1'})
print(a)
```

And it will return

```
"""
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
"""
```

Search with Tag Name and Text Inside

Another way to search is using tag name and text inside that tag. To do that we use `string` parameter.

For example following code will return all `a` tag which has text inside is "Elsie"

```python
a_elsie = soup.find_all('a', string = 'Elsie')
print(a_elsie)
```

And it will return

```
"""
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]
"""
```

Search Scope

You could see that HTML content page is structured in tree. So after find out the parent tag, we can find out child inside that parent.

For example following code will return the first `b` tag inside the first `p` tag.

```python
first_p = soup.find('p')
first_b_inside_first_p = first_p.find('b')

print(first_b_inside_first_p)
```

And it print out

```
"""
<b>The Dormouse's story</b>
"""
```