HTML Parsing

All Versions

Examples

Using CSS selectors in BeautifulSoup

BeautifulSoup has a limited support for CSS selectors , but covers most commonly used ones. Use select() method to find multiple elements and select_one() to find a single element.

Basic example:

```
from bs4 import BeautifulSoup

data = """
<ul>
    <li class="item">item1</li>
    <li class="item">item2</li>
    <li class="item">item3</li>
</ul>
"""

soup = BeautifulSoup(data, "html.parser")

for item in soup.select("li.item"):
    print(item.get_text())
```

Prints:

```
item1
item2
item3
```

Locate a text after an element in BeautifulSoup

Imagine you have the following HTML:

```
<div>
    <label>Name:</label>
    John Smith
</div>
```

And you need to locate the text "John Smith" after the label element.

In this case, you can locate the label element by text and then use .next_sibling property :

```
from bs4 import BeautifulSoup

data = """
<div>
    <label>Name:</label>
    John Smith
</div>
"""

soup = BeautifulSoup(data, "html.parser")

label = soup.find("label", text="Name:")
print(label.next_sibling.strip())
```

Prints John Smith .

PyQuery

pyquery is a jquery-like library for python. It has very well support for css selectors.

```
from pyquery import PyQuery

html = """
<h1>Sales</h1>
<table id="table">
<tr>
    <td>Lorem</td>
    <td>46</td>
```

```
</tr>
<tr>
    <td>Ipsum</td>
    <td>12</td>
</tr>
<tr>
    <td>Dolor</td>
    <td>27</td>
</tr>
<tr>
    <td>Sit</td>
    <td>90</td>
</tr>
</table>
"""

doc = PyQuery(html)

title = doc('h1').text()

print title

table_data = []

rows = doc('#table > tr')
for row in rows:
    name = PyQuery(row).find('td').eq(0).text()
    value = PyQuery(row).find('td').eq(1).text()
```

Syntax

Parameters

Remarks