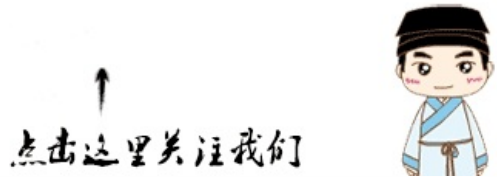
【零基础学爬虫】PyQuery解析器(保姆级别)



01 准备工作

> 如果你对Web编程有所了解，如果你比较喜欢CSS选择器，如果你对jquery有所了解，那么这里有一个更适合你的解析库pyquery。

```
pip3 install pyquery1
```

02 初始化

在用pyquery库解析HTML文本的时候，需要先将其初始化为一个PyQuery对象。
初始化有很多种方法，比如直接传入字符串，传入url、文件名称，等等。

- **字符串初始化**
  这种方式很多方式直接把HTML的内容当作初始化参数，来初始化pyquery对象。可以用一个实例来感受一下：

```
html = '''
<div>
<ul>
<li class="item-0">first item</li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>
</ul>
</div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
```

```
print(doc('li'))123456789101112131415
```

这里首先引用PyQuery这个对象，取别名为pq。然后声明一个长HTML字符串，并将其当作参数传递给PyQuery类，这样就成功完成了初始化。接着，将初始化的对象传入CSS选择器。在这个实例中，我们传入li节点，这样可以选择所有的li节点了。 运行结果如下：

```
<li class="item-0">first item</li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>12345
```

- URL初始化
  初始化的参数除了能以字符串形式传递，还能是网页的URL，此时只需要指定PyQuery对象参数为url即可：

```
from pyquery import PyQuery as pq

doc = pq(url="https://cuiqingcai.com")

print(doc('title'))12345
```

运行结果如下：

```
<title>静觅丨崔庆才的个人站点</title>1
```

这样的话PyQuery对象会首先请求这个URL，然后用得到的HTML内容完成初始化，其实就相当于把网页的源代码以字符串形式传递给PyQuery类，来完成初始化操作。 下面代码实现的功能是相同的：

```
from pyquery import PyQuery as pq
import requests

doc = pq(requests.get('https://cuiqingcai.com').text)
print(doc('title'))12345
```

- 文件初始化
  除了上面两种，还可以传递本地的文件名，此时将参数指定为filename即可：

```
from pyquery import PyQuery as pq
doc = pq(filename='demo.html')
print(doc('li'))123
```

本地这里需要有一个html文件demo.html,其内容是待解析的HTML字符串。这样PyQuery对象会首先读取本地的文件内容，然后将文件内容以字符串的形式传递给PyQuery类型进行初始化。

03 基本的CSS选择器

首先用一个实例感受一下pyquery库的css选择器的用法：

```
html = '''
<div id="container">
<ul class="list">
<li class="item-0">first item</li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>
</ul>
</div>
```

```
'''
from pyquery import PyQuery as pq
doc = pq(html)
print(doc('#container .list li'))
print(type(doc('#container .list li')))12345678910111213141 5
```

运行结果如下：

```
<li class="item-0">first item</li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>

<class 'pyquery.pyquery.PyQuery'>1234567
```

这里我们初始化PyQuery对象之后，传入一个css选择器 #container .list li,他的意思是先选取id为container的节点，在选取其内部class为list的节点内部的所有li节点，然后打印输出。运行结果可以看到，我们成功获取了符合条件的节点。
最后，将符合条件的节点的类型打印输出，可以看到依然是PyQuery类型。　下面，我们直接遍历获取的所有节点。然后调用text方法，就可以直接获取节点文本内容代码如下：

```
for item in doc('#container .list li').items():
print(item.text())12
```

运行结果如下：

```
first item
second item
third item
fourth item
fifth item12345
```

04　查找节点

- 子节点
  查找子节点时，需亚用到find方法，其参数是CSS选
  择器。这里我们还是以上面的HTML为例：

```
from pyquery import PyQuery as pq
doc = pq(html)
items = doc(".list")
print(type(items))
print(items)
lis = items.find("li")
print(type(lis))
print(lis)123456789
```

运行结果如下：

```
<class 'pyquery.pyquery.PyQuery'>
<ul class="list">
        <li class="item-0">first item</li>
        <li class="item-1"><a href="link2.html">second item</a></li>
        <li class="item-0 active"><a href="link3.html"><span class="bold">third item</span><
/a></li>
        <li class="item-1 active"><a href="link4.html">fourth item</a></li>
        <li class="item-0"><a href="link5.html">fifth item</a></li>
</ul>

<class 'pyquery.pyquery.PyQuery'>
<li class="item-0">first item</li>
```

```
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>123456789101112131415
```

这里我们先通过.list参数选取class为list的节点。然后调用find方法，并给其传入CSS选择器，选取其内部li节点，最后打印输出。可以发现，find方法会将所有符合条件的节点选择出来，结果为PyQuery类型。　其实find方法的查找范围的的节点的所有子孙节点。如果只想查找子节点，那么可以用children方法：

```
items = doc(".list")
lis = items.children()
print(type(lis))
print(lis)1234
```

运行结果如下：

```
<class 'pyquery.pyquery.PyQuery'>
<li class="item-0">first item</li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>123456
```

如果要筛选所有的子节点中符合条件的节点，例如想筛选出子节点中class为active的节点，则可以向children方法传入class选择器 .active,代码如下：

```
items = doc(".list")
lis = items.children(".active")
print(lis)1234
```

运行结果如下:

```
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>12
```

- 父节点
  我们可以用parent方法获取某个节点的父节点，下面
  用一个实例感受一下：

```
html = '''
<div id="container">
        <div class="wrap">
            <ul class="list">
                 <li class="item-0">first item</li>
                 <li class="item-1"><a href="link2.html">second item</a></li>
                 <li class="item-0 active"><a href="link3.html"><span class="bold">third ite
m</span></a></li>
                 <li class="item-1 active"><a href="link4.html">fourth item</a></li>
                 <li class="item-0"><a href="link5.html">fifth item</a></li>
             </ul>
        </div>
 </div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
items = doc(".list")
container = items.parent()
print(type(container))
print(container)12345678910111213141516171819
```

运行结果如下：

```
<class 'pyquery.pyquery.PyQuery'>
```

```
<div class="wrap">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
    </div>12345678910
```

这个parent方法不会找到祖先节点如果想找到祖先节需要用parents()
如果想要在某个祖先节点查找，可以在parents()中传入CSS选择器，这样就会返回祖先节点中符合CSS选择器的节点：

```
parent = items.parents(".wrap")
print(parent)123
```

结果如下：

```
<div class="wrap">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
    </div>123456789
```

- 兄弟节点
  获取兄弟节点可以用siblings方法。这里还是以上面
  的HTML文本为例：

```
from pyquery import PyQuery as pq
doc = pq(html)
items = doc(".list .item-0.active")
print(items.siblings())1234
```

这里首先选择class为list的节点内部的class为item-0和active的节点，获取结果为除了此节点的所有同级别节点：

```
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-0">first item</li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>1234
```

如果想要筛选某个节点，再往siblings方法传入css选择器例如：

```
from pyquery import PyQuery as pq
doc = pq(html)
items = doc(".list .item-0.active")
print(items.siblings(".active"))1234
```

如下：

```
<li class="item-1 active"><a href="link4.html">fourth item</a></li>12
```

05 遍历节点

pyquery库的选择结果可能是多个节点，也可能是单个节点，类型都是pyquery类型，并没有像beautifulsoup那样返回列表 如果结果是单个节点，既可以直接打印输出，也可以直接转成字符串：

```
from pyquery import PyQuery as pq
doc = pq(html)
items = doc(".item-0.active")
print(items)
print(str(items))12345
```

结果如下：

```
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>

<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
123
```

如果结果是多个节点，就需要遍历获取了。调用items方法

```
from pyquery import PyQuery as pq
doc = pq(html)
lis = doc("li").items()
for li in lis:
    print(li, type(li))12345
```

这里把所有li节点遍历一遍，运行结果如下：

```
<li class="item-0">first item</li>
<class 'pyquery.pyquery.PyQuery'>
<li class="item-1"><a href="link2.html">second item</a></li>
<class 'pyquery.pyquery.PyQuery'>
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
<class 'pyquery.pyquery.PyQuery'>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<class 'pyquery.pyquery.PyQuery'>
<li class="item-0"><a href="link5.html">fifth item</a></li>
<class 'pyquery.pyquery.PyQuery'>12345678910
```

可以发现，**调用items方法后，会得到一个生成器**，对其进行遍历，就可以逐个得到li节点对象，类型也是pyquery。**还可以调用前面所说的方法对li节点进行选择，继续查询子节点或者祖先节点**

- 获取属性
  提取到某个pyquery类型的节点后，可以调用attr方
  法获取其属性

```
html = '''
<div class="wrap">
    <div id="container">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
     </div>
 </div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
a = doc('.item-0.active a')
print(a, type(a))
print(a.attr('href'))12345678910111212131415161718
```

运行结果为：

```
<a href="link3.html"><span class="bold">third item</span></a> <class 'pyquery.pyquery.PyQuery'>
link3.html12
```

先选择item0 active a节点，在调用attr获取属性内容 如果用print(a.attr.href)结果也是一样的
如果多个节点在调用attr方法看看会怎么样：

```
from pyquery import PyQuery as pq
doc = pq(html)
a = doc('a')
print(a, type(a))
print(a.attr.href)12345
```

运行结果为：

```
<a href="link2.html">second item</a><a href="link3.html"><span class="bold">third item</span></a><a href="link4.html">fourth item</a><a href="link5.html">fifth item</a> <class 'pyquery.pyquery.PyQuery'>
link2.html12
```

我们得到的节点四个结果，但是attr只有一个属性，调用attr方法，只会得到第一个节点的属性
如果想要获取全部属性则需要遍历了：

```
from pyquery import PyQuery as pq
doc = pq(html)
a = doc('a')
for i in a.items():
    print(i, type(a))
    print(i.attr.href)123456
```

结果如下：

```
<a href="link2.html">second item</a> <class 'pyquery.pyquery.PyQuery'>
link2.html
<a href="link3.html"><span class="bold">third item</span></a> <class 'pyquery.pyquery.PyQuery'>
link3.html
<a href="link4.html">fourth item</a> <class 'pyquery.pyquery.PyQuery'>
link4.html
<a href="link5.html">fifth item</a> <class 'pyquery.pyquery.PyQuery'>
link5.html12345678
```

此时无论是多个还是单个都可以获取到了

- 获取文本
  获取节点之后的另一个主要操作就是获取其中内部的
  文本，此时可以调用text方法实现：

```
html = '''
<div class="wrap">
    <div id="container">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
     </div>
 </div>
'''
from pyquery import PyQuery as pq
```

```
doc = pq(html)
a = doc('.item-0.active a')
print(a)
print(a.text())12345678910111213141516171819
```

结果如下：

```
<a href="link3.html"><span class="bold">third item</span></a>
third item12
```

这里首先选中a节点，调用text方法，就可以获取其中的文本信息。此时text方法会忽略节点内部的所有HTML，只返回纯文字内容。
要想获取节点内部的HTML文本，需要用html方法：

```
from pyquery import PyQuery as pq
doc = pq(html)
li = doc('.item-0.active')
print(li)
print(li.html())12345
```

返回li节点内部的所有html文本：

```
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>

<a href="link3.html"><span class="bold">third item</span></a>123
```

这里有一个问题，如果我们选中多个节点，那么text或html方法会返回什么内容？ 用一个实例来看一看：

```
html = '''
<div class="wrap">
    <div id="container">
        <ul class="list">
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
     </div>
 </div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
li = doc('li')
print(li.html())
print(li.text())12345678910111213141516171617
```

结果如下：

```
<a href="link2.html">second item</a>
second item third item fourth item fifth item12
```

结果出乎意料，html方法返回的第一个li内部的html文本，而text返回了所有的li节点内部的文本，个节点用一个空格分隔开，返回结果是一个字符串

06 节点操作

pyquery库提供了一系列方法对节点进行动态修改，例如为某个节点添加一个class，移除某个节点等，有时候这些操作会为提供信息带来极大的便利！
节点操作的方法太多，下面仅举几个典型的来说明用法：

- addClass 和 removeClass

我们来感受一下：

```
   html = '''
<div class="wrap">
    <div id="container">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
         </ul>
     </div>
 </div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
li = doc('.item-0.active')
print(li)
li.removeClass('active')
print(li)
li.addClass('active')
print(li)123456789101112131415161718192021 22
```

首先选取第三个li节点，然后调用removeClass方法，将其active这个class移除，然后又调用addclass方法，把他添加回来结果如下：

```
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>

<li class="item-0"><a href="link3.html"><span class="bold">third item</span></a></li>

<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>
12345
```

addclass方法和removeclass方法可以动态改变节点的class属性、

- attr 、 text 、和html
  当然除了上面两个方法也可以用，attr方法对属性进行
  操作。此外text方法可以用来改变节点内部的内容示
  例如下：

```
html = '''
<ul class="list">
     <li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a>
</li>
</ul>
'''
from pyquery import PyQuery as pq

doc = pq(html)
li = doc('.item-0.active')
print(li)
li.attr('name', 'link')
print(li)
li.text('changed item')
print(li)
li.html('<span>changed item</span>')
print(li)12345678910111213141516
```

结果如下：

```
<li class="item-0 active"><a href="link3.html"><span class="bold">third item</span></a></li>

<li class="item-0 active" name="link"><a href="link3.html"><span class="bold">third item</sp
an></a></li>

<li class="item-0 active" name="link">changed item</li>

<li class="item-0 active" name="link"><span>changed item</span></li>1234567
```

可以发现，调用attr方法后，li节点多了一个原本不存在的属性name，其值为link。接着调用text方法，传入文本后，li节点内部的文本全被改为传入字符串文本。最后调用html传入传入文本后，li节点内又变为传入的html文本。

- remove
  顾名思义，remove方法的作用是移除，有时为信息的
  提取带来非常大的便利：

```
html = '''
<div class="wrap">
    Hello, World
    <p>This is a paragraph.</p>
 </div>
'''
from pyquery import PyQuery as pq
doc = pq(html)
wrap = doc('.wrap')
print(wrap.text())12345678910
```

现在想要提取文本中的Hello，World这个字符串，而不要p节点的字符串，该怎么操作呢？　　这里直接尝试提取class为wrap节点的内容，看看是不是我们想要的：

```
Hello, World
This is a paragraph.12
```

这个结果中包含p节点的内容，也就说text方法把所有纯文本内容提取出来。想要去掉p节点的文本，可以再把p节点的文本提取一边，然后移除这个字符串，明显这个做法很繁琐 这时remove方法就派上用场了：

```
from pyquery import PyQuery as pq
doc = pq(html)
wrap = doc('.wrap')
wrap.find("p").remove()
print(wrap.text())12345
```

此时结果为：

```
Hello, World1
```

07　伪类选择器

CSS选择器之所以强大，还有一个很重要的原因就是它支持多种多样的伪类选择器。例如选择第一个节点、最后一个节点、奇偶数节点、包含某一文本的节点等等，我们用一个实例感受一下：

```
html = '''
<div class="wrap">
    <div id="container">
        <ul class="list">
             <li class="item-0">first item</li>
             <li class="item-1"><a href="link2.html">second item</a></li>
             <li class="item-0 active"><a href="link3.html"><span class="bold">third item</s
pan></a></li>
             <li class="item-1 active"><a href="link4.html">fourth item</a></li>
             <li class="item-0"><a href="link5.html">fifth item</a></li>
```

```
        </ul>
    </div>
 </div>

'''
from pyquery import PyQuery as pq
doc = pq(html)
li = doc('li:first-child')#节点、第一个
print(li)
li = doc('li:last-child')#节点、最后一个
print(li)
li = doc('li:nth-child(2)')#节点、第二个
print(li)
li = doc('li:gt(2)')#获取序号比2大的标签
print(li)
li = doc('li:nth-child(2n)')#获取偶数的标签,2n+1则是获得奇数
print(li)
li = doc('li:contains(second)')#包含second文本的
print(li)12345678910111213141516171819202122232425262728
```

结果如下：

```
<li class="item-0">first item</li>

<li class="item-0"><a href="link5.html">fifth item</a></li>

<li class="item-1"><a href="link2.html">second item</a></li>

<li class="item-1 active"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a></li>

<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-1 active"><a href="link4.html">fourth item</a></li>

<li class="item-1"><a href="link2.html">second item</a></li>12345678910111213
```