上一课时我们学习了正则表达式的基本用法，然而一旦你的正则表达式写法有问题，我们就无法获取需要的信息。

你可能会思考：每个网页，都有一定的特殊结构和层级关系，而且很多节点都有 id 或 class 作为区分，我们可以借助它们的结构和属性来提取信息吗？

这的确可行。这个课时我会为你介绍一个更加强大的 HTML 解析库：pyquery。利用它，我们可以直接解析 DOM 节点的结构，并通过 DOM 节点的一些属性快速进行内容提取。

接下来，我们就来感受一下 pyquery 的强大之处。

准备工作

pyquery 是 Python 的第三方库，我们可以借助于 pip3 来安装，安装命令如下：

```
pip3 install pyquery
```

更详细的安装方法可以参考：https://cuiqingcai.com/5186.html。

初始化

我们在解析 HTML 文本的时候，首先需要将其初始化为一个 pyquery 对象。它的初始化方式有多种，比如直接传入字符串、传入 URL、传入文件名，等等。

下面我们来详细介绍一下。

字符串初始化

我们可以直接把 HTML 的内容当作参数来初始化 pyquery 对象。我们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'))
```

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>
```

这里首先引入 pyquery 这个对象，取别名为 pq，然后声明了一个长 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。

接下来，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，这样就可以选择所有的 li 节点。

URL 初始化

初始化的参数不仅可以以字符串的形式传递，还可以传入网页的 URL，此时只需要指定参数为 url 即可：

```
from pyquery import PyQuery as pq
doc = pq(url='https://cuiqingcai.com')
print(doc('title'))
```

运行结果：

```
<title>静觅丨崔庆才的个人博客</title>
```

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。这就相当于将网页的源代码以字符串的形式传递给 pyquery 类来初始化。

它与下面的功能是相同的：

```
from pyquery import PyQuery as pq
import requests
doc = pq(requests.get('https://cuiqingcai.com').text)
print(doc('title'))
```

文件初始化

当然除了传递一个 URL，我们还可以传递本地的文件名，参数指定为 filename 即可：

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

当然，这里需要有一个本地 HTML 文件 demo.html，其内容是待解析的 HTML 字符串。这样它会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 类来初始化。

以上 3 种方式均可初始化，当然最常用的初始化方式还是以字符串形式传递。

基本 CSS 选择器

我们先用一个实例来感受一下 pyquery 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')))
```

运行结果：

```
<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'>
```

在上面的例子中，我们初始化 pyquery 对象之后，传入 CSS 选择器 #container .list li，它的意思是先选取 id 为 container 的节点，然后再选取其内部 class 为 list 的所有 li 节点，最后打印输出。

可以看到，我们成功获取到了符合条件的节点。我们将它的类型打印输出后发现，它的类型依然是 pyquery 类型。

下面，我们直接遍历这些节点，然后调用 text 方法，就可以获取节点的文本内容，代码示例如下：

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

运行结果如下：

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

怎么样？我们没有再写正则表达式，而是直接通过选择器和 text 方法，就得到了我们想要提取的文本信息，是不是方便多了？

下面我们再来详细了解一下 pyquery 的用法吧，我将为你讲解如何用它查找节点、遍历节点、获取各种信息等操作方法。掌握了这些，我们就能更高效地完成数据提取。

查找节点

下面我们介绍一些常用的查询方法。

子节点

查找子节点需要用到 find 方法，传入的参数是 CSS 选择器，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)
```

运行结果：

```
<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>
```

首先，我们通过 .list 参数选取 class 为 list 的节点，然后调用 find 方法，传入 CSS 选择器，选取其内部的 li 节点，最后打印输出。可以发现，find 方法会将符合条件的所有节点选择出来，结果的类型是 pyquery 类型。

find 的查找范围是节点的所有子孙节点，而如果我们只想查找子节点，那可以用 children 方法：

```
lis = items.children()
print(type(lis))
print(lis)
```

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>
```

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

```
lis = items.children('.active')
print(lis)
```

运行结果：

```
<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>
```

我们看到输出的结果已经做了筛选，留下了 class 为 active 的节点。

父节点

我们可以用 parent 方法来获取某个节点的父节点，下面用一个实例来感受一下：

```
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</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)
```

运行结果如下：

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

```
<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>
```

在上面的例子中我们首先用 .list 选取 class 为 list 的节点，然后调用 parent 方法得到其父节点，其类型依然是 pyquery 类型。

这里的父节点是该节点的直接父节点，也就是说，它不会再去查找父节点的父节点，即祖先节点。

但是如果你想获取某个祖先节点，该怎么办呢？我们可以用 parents 方法：

```
from pyquery import PyQuery as pq
doc = pq(html)
items = doc('.list')
parents = items.parents()
print(type(parents))
print(parents)
```

运行结果如下：

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

可以看到，这个例子的输出结果有两个：一个是 class 为 wrap 的节点，一个是 id 为 container 的节点。也就是说，使用 parents 方法会返回所有的祖先节点。

如果你想要筛选某个祖先节点的话，可以向 parents 方法传入 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)
```

运行结果如下：

```
<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>
```

可以看到，输出结果少了一个节点，只保留了 class 为 wrap 的节点。

兄弟节点

前面我们说明了子节点和父节点的用法，还有一种节点叫作兄弟节点。如果要获取兄弟节点，可以使用 siblings 方法。这里还是以上面的 HTML 代码为例：

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

在这个例子中我们首先选择 class 为 list 的节点，内部 class 为 item-0 和 active 的节点，也就是第 3 个 li 节点。很明显，它的兄弟节点有 4 个，那就是第 1、2、4、5 个 li 节点。

我们来运行一下：

```
<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>
```

可以看到，结果显示的正是我们刚才所说的 4 个兄弟节点。

如果要筛选某个兄弟节点，我们依然可以用 siblings 方法传入 CSS 选择器，这样就会从所有兄弟节点中挑选出符合条件的节点了：

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

在这个例子中我们筛选 class 为 active 的节点，从刚才的结果中可以观察到，class 为 active 兄弟节点的是第 4 个 li 节点，所以结果应该是1个。

我们再看一下运行结果：

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

遍历

通过刚才的例子我们可以观察到，pyquery 的选择结果既可能是多个节点，也可能是单个节点，类型都是 pyquery 类型，并没有返回列表。

对于单个节点来说，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)
li = doc('.item-0.active')
print(li)
print(str(li))
```

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

对于有多个节点的结果，我们就需要用遍历来获取了。例如，如果要把每一个 li 节点进行遍历，需要调用 items 方法：

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

运行结果如下：

```
<class 'generator'>
<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'>
```

可以发现，调用 items 方法后，会得到一个生成器，遍历一下，就可以逐个得到 li 节点对象了，它的类型也是 pyquery 类型。每个 li 节点还可以调用前面所说的方法进行选择，比如继续查询子节点，寻找某个祖先节点等，非常灵活。

获取信息

提取到节点之后，我们的最终目的当然是提取节点所包含的信息了。比较重要的信息有两类，一是获取属性，二是获取文本，下面分别进行说明。

获取属性

提取到某个 pyquery 类型的节点后，就可以调用 attr 方法来获取属性：

```
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</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, type(a))
print(a.attr('href'))
```

运行结果如下：

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

在这个例子中我们首先选中 class 为 item-0 和 active 的 li 节点内的 a 节点，它的类型是 pyquery 类型。

然后调用 attr 方法。在这个方法中传入属性的名称，就可以得到属性值了。

此外，也可以通过调用 attr 属性来获取属性值，用法如下：

```
print(a.attr.href)
```

结果：

```
link3.html
```

这两种方法的结果完全一样。

如果选中的是多个元素，然后调用 attr 方法，会出现怎样的结果呢？我们用实例来测试一下：

```
a = doc('a')
print(a, type(a))
print(a.attr('href'))
print(a.attr.href)
```

运行结果如下：

```
<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
link2.html
link2.html
```

照理来说，我们选中的 a 节点应该有 4 个，打印结果也应该是 4 个，但是当我们调用 attr 方法时，返回结果却只有第 1 个。这是因为，当返回结果包含多个节点时，调用 attr 方法，只会得到第 1 个节点的属性。

那么，遇到这种情况时，如果想获取所有的 a 节点的属性，就要用到前面所说的遍历了：

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

运行结果：

```
link2.html
link3.html
link4.html
link5.html
```

因此，在进行属性获取时，先要观察返回节点是一个还是多个，如果是多个，则需要遍历才能依次获取每个节点的属性。

获取文本

获取节点之后的另一个主要操作就是获取其内部文本了，此时可以调用 text 方法来实现：

```
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())
```

运行结果：

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

这里我们首先选中一个 a 节点，然后调用 text 方法，就可以获取其内部的文本信息了。text 会忽略节点内部包含的所有 HTML，只返回纯文字内容。

但如果你想要获取这个节点内部的 HTML 文本，就要用 html 方法了：

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

这里我们选中第 3 个 li 节点，然后调用 html 方法，它返回的结果应该是 li 节点内的所有 HTML 文本。

运行结果：

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

这里同样有一个问题，如果我们选中的结果是多个节点，text 或 html 方法会返回什么内容？我们用实例来看一下：

```
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</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)
li = doc('li')
print(li.html())
print(li.text())
print(type(li.text()))
```

运行结果如下：

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

结果比较出乎意料，html 方法返回的是第 1 个 li 节点的内部 HTML 文本，而 text 则返回了所有的 li 节点内部的纯文本，中间用一个空格分割开，即返回结果是一个字符串。

这个地方值得注意，如果你想要得到的结果是多个节点，并且需要获取每个节点的内部 HTML 文本，则需要遍历每个节点。而 text 方法不需要遍历就可以获取，它将所有节点取文本之后合并成一个字符串。

节点操作

pyquery 提供了一系列方法来对节点进行动态修改，比如为某个节点添加一个 class，移除某个节点等，这些操作有时会为提取信息带来极大的便利。

由于节点操作的方法太多，下面举几个典型的例子来说明它的用法。

addClass 和 removeClass

我们先用一个实例来感受一下：

```
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</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)
li = doc('.item-0.active')
print(li)
li.removeClass('active')
print(li)
li.addClass('active')
print(li)
```

首先选中第 3 个 li 节点，然后调用 removeClass 方法，将 li 节点的 active 这个 class 移除，第 2 步调用 addClass 方法，将 class 添加回来。每执行一次操作，就打印输出当前 li 节点的内容。

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-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>
```

可以看到，一共输出了 3 次。第 2 次输出时，li 节点的 active 这个 class 被移除了，第 3 次 class 又添加回来了。

所以说，addClass 和 removeClass 方法可以动态改变节点的 class 属性。

attr、text、html

当然，除了操作 class 这个属性外，也可以用 attr 方法对属性进行操作。此外，我们还可以用 text 和 html 方法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)
```

这里我们首先选中 li 节点，然后调用 attr 方法来修改属性。该方法的第 1 个参数为属性名，第 2 个参数为属性值。最后调用 text 和 html 方法来改变节点内部的内容。3 次操作后，分别打印输出当前的 li 节点。

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

我们发现，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，li 节点内部的文本全被改为传入的字符串文本。最后，调用 html 方法传入 HTML 文本，li 节点内部又变为传入的 HTML 文本了。

所以说，使用 attr 方法时如果只传入第 1 个参数的属性名，则是获取这个属性值；如果传入第 2 个参数，可以用来修改属性值。使用 text 和 html 方法时如果不传参数，则是获取节点内纯文本和 HTML 文本，如果传入参数，则进行赋值。

remove

顾名思义，remove 方法就是移除，它有时会为信息的提取带来非常大的便利。下面有一段 HTML 文本：

```
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())
```

现在我们想提取“Hello, World”这个字符串，该怎样操作呢？

这里先直接尝试提取 class 为 wrap 的节点的内容，看看是不是我们想要的。

运行结果如下：

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

这个结果还包含了内部的 p 节点的内容，也就是说 text 把所有的纯文本全提取出来了。

如果我们想去掉 p 节点内部的文本，可以选择再把 p 节点内的文本提取一遍，然后从整个结果中移除这个子串，但这个做法明显比较烦琐。

这时 remove 方法就可以派上用场了，我们可以接着这么做：

```
wrap.find('p').remove()
print(wrap.text())
```

首先选中 p 节点，然后调用 remove 方法将其移除，这时 wrap 内部就只剩下“Hello, World”这句话了，最后利用 text 方法提取即可。

其实还有很多其他节点操作的方法，比如 append、empty 和 prepend 等方法，详细的用法可以参考官方文档：http://pyquery.readthedocs.io/en/latest/api.html。

伪类选择器

CSS 选择器之所以强大，还有一个很重要的原因，那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。示例如下：

```
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</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)
li = doc('li:first-child')
print(li)
li = doc('li:last-child')
print(li)
li = doc('li:nth-child(2)')
print(li)
li = doc('li:gt(2)')
print(li)
li = doc('li:nth-child(2n)')
print(li)
li = doc('li:contains(second)')
print(li)
```

在这个例子中我们使用了 CSS3 的伪类选择器，依次选择了第 1 个 li 节点、最后一个 li 节点、第 2 个 li 节点、第 3 个 li 之后的 li 节点、偶数位置的 li 节点、包含 second 文本的 li 节点。

关于 CSS 选择器的更多用法，可以参考 http://www.w3school.com.cn/css/index.asp。

到此为止，pyquery 的常用用法就介绍完了。如果想查看更多的内容，可以参考 pyquery 的官方文档：http://pyquery.readthedocs.io。相信一旦你拥有了它，解析网页将不再是难事。