数据提取之BeautifuSoup模块与Css选择器(拓展)

- bs4全称是BeautifulSoup4

> xpath：根据路径找数据
>
> bs4：根据方法找数据(调用对应的方法实现对应的功能)
>
> 学习方法：https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/

```
"""
# BeautifulSoup
是一个高效的网页解析库，可以从HTML或XML文件中提取数据
支持不同的解析器，比如，对HTML解析，对XML解析，对HTML5解析
就是一个非常强大的工具，爬虫利器
一个灵感又方便的网页解析库，处理高效，支持多种解析器
利用它就不用编写正则表达式也能方便的实现网页信息的抓取
"""

# 安装 pip install BeautifulSoup4 -i 换源地址
```

```
# 解析器 pip install lxml/html5lib


# 标签选择器
### 通过标签选择
#### .string() --获取文本节点及内容

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title" name="dromouse"><b>
<span>The Dormouse's story</span></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>
"""
from bs4 import BeautifulSoup   # 导包
```

```
soup = BeautifulSoup(html, 'lxml')  # 参数1：要解析的html  参数2：解析器

# print(soup.prettify())  # 代码补全

print(soup.html.head.title.string)

print(soup.title.string)  #title是个节点，.string是属性 作用是获取字符串文本

# 选取整个head，包含标签本身
print(soup.head) # 包含head标签在内的所有内容

print(soup.p) # 返回匹配的第一个结果


#%% md

### 获取名称
#### .name()  --获取标签本身名称

#%%

html = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title" name="dromouse"><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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

print(soup.title.name)  # 结果为标签本身  -->
title
print(soup.p.name)  # --> 获取标签名

#%% md

### 获取属性值

#### .attrs()  --通过属性拿属性值

#%%

html = """
```

```
<html><head><title>The Dormouse's
story</title></head>
<body>
<p class="title asdas" name="abc" id =
"qwe"><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/123"
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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

print(soup.p.attrs['name'])# 获取p标签name属
性的属性值

print(soup.p.attrs['id']) # 获取p标签id属性的
属性值
print(soup.p['id']) #第二种写法

print(soup.p['class']) # 以列表得形式保存
print(soup.a['href'])  # 也是只返回第一个值
```

```
#%% md

### 嵌套选择

一定要有子父级关系

#%%

html = """
<html><head><title>The Dormouse's
story</title></head>
<body>
<p class="title" name="dromouse"><b>The abc
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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
```

```
print(soup.body.p.b.string)  #层层往下找

#%% md

### 子节点和子孙节点

#%%

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <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">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

# 标签选择器只能拿到部分内容 ，不能拿到所有，那如何解决？？

# .contents属性可以将tag(标签)的子节点以列表的形式输出
# print(soup.p.contents)  # 获取P标签所有子节点内容 返回一个list

for i in soup.p.contents:
    print(i)


#%%



#%%

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
```

```
    </head>
    <body>
        <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">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

# .children是一个list类型的迭代器
print(soup.p.children)  # 获取子节点  返回一个
迭代器

for i in soup.p.children:
```

```
    print(i)

for i, child in enumerate(soup.p.children):

    print(i, child)

#%%

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <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">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.p.descendants)  # 获取子孙节点  返
回一个迭代器
for i, child in
enumerate(soup.p.descendants):
    print(i, child)

#%% md

### 父节点和祖先节点

#%%

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <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">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.a.parent)  # 获取父节点

#%%

html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <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">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(list(enumerate(soup.a.parents)))  #
获取祖先节点

#%% md

### 兄弟节点

#%%
```

```
html = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <p class="story">
            <span>abcqweasd</span>
            Once upon a time there were
three little sisters; and their names were
            <a
href="http://example.com/elsie"
class="sister" id="link1">
                <span>Elsie</span>
            </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>
"""
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
```

```
print(list(enumerate(soup.a.next_siblings))
)  # 后边的所有的兄弟节点
print('---'*15)
print(list(enumerate(soup.a.previous_siblin
gs))) # 前边的

#%% md

## 实用：标准选择器

#%% md

### find_all( name , attrs , recursive ,
text , **kwargs )

#%% md

可根据标签名、属性、内容查找文档

#%% md

#### 使用find_all根据标签名查找

#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
```

```
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo-2</li>
            <li class="element">Bar-2</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

print(soup.find_all('ul'))  # 拿到所有ul标签
及其里面内容
print(soup.find_all('ul')[0])

ul = soup.find_all('ul')
print(ul) # 拿到整个ul标签及其里面内容
print('_____________'*10)

for ul in soup.find_all('ul'):
#     print(ul)  # 遍历ul标签
    for li in ul:
#         print(li)  #遍历li标签
```

```
        print(li.string)  # 拿到所有li标签里的
文本内容

#%% md

#### 获取文本值

#%%

for ul in soup.find_all('ul'):
    for i in ul.find_all("li"):
        print(i.string)

#%% md

#### 根据属性查找

#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1"
name="elements">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
```

```
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

# 第一种写法 通过attrs
# print(soup.find_all(attrs={'id': 'list-
1'})) # 根据id属性
print("-----"*10)
# print(soup.find_all(attrs={'name':
'elements'}))  # 根据name属性

for ul in soup.find_all(attrs={'name':
'elements'}):
    print(ul)
    print(ul.li.string)  #只能给你返回第一个值
# # # #     print('-----')
    for li in ul:
#         print(li)
        print(li.string)

#%% md
```

```
#### 特殊的属性查找

#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''

from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

# 第二种写法
print(soup.find_all(id='list-1'))
```

```
print(soup.find_all(class_='element'))  #
class属于Python关键字，做特殊处理 _

# 推荐的查找方法 li标签下的class属性
print(soup.find_all('li',
{'class','element'}))

#%% md

####  根据文本值选择 text

#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
```

```
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

print(soup.find_all(text='Foo')) # 可以做内容统计用
print(soup.find_all(text='Bar'))
print(len(soup.find_all(text='Foo'))) # 统计数量

#%% md

### find( name , attrs , recursive , text , **kwargs )

#%% md

find返回单个元素，find_all返回所有元素

#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
```

```
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
print(soup.find('ul')) # 只返回匹配到的第一个
print(soup.find('li'))

print(soup.find('page')) # 如果标签不存在返回
None

#%% md

### find_parents()  find_parent()

#%% md

find_parents()返回所有祖先节点，find_parent()
返回直接父节点。

#%% md
```

```
### find_next_siblings()
find_next_sibling()

#%% md

find_next_siblings()返回后面所有兄弟节点，
find_next_sibling()返回后面第一个兄弟节点。

#%% md

### find_previous_siblings()
find_previous_sibling()

#%% md

find_previous_siblings()返回前面所有兄弟节点，
find_previous_sibling()返回前面第一个兄弟节点。

#%% md

### find_all_next()  find_next()

#%% md

find_all_next()返回节点后所有符合条件的节点，
find_next()返回第一个符合条件的节点

#%% md

### find_all_previous() 和 find_previous()
```

#%% md

find_all_previous()返回节点后所有符合条件的节点，find_previous()返回第一个符合条件的节点

#%% md

CSS选择器

#%% md

通过select()直接传入CSS选择器即可完成选择

如果对HTML里的CSS选择器很熟悉可以考虑用此方法

#%% md

注意：

1，写CSS时，标签名不加任何修饰，类名前加.，id名前加#

2，用到的方法时soup.select()，返回类型是list

3，多个过滤条件需要用空格隔开,从前往后是逐层筛选

```
#%%

html='''
<div class="pan">q321312321</div>
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')

# 层级 ul li
print(soup.select('ul li'))  # 标签不加任何修
饰
print("----"*10)
```

```
print(soup.select('.panel .panel-heading'))
# 类名前加.
print("----"*10)

print(soup.select('#list-1 .element'))
print("----"*10)


#%%

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(html, 'lxml')
for ul in soup.select('ul'):
    for i in ul.select('li'):
        print(i.string)


### 获取属性


html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
```

```
# []获取id属性  attrs获取class属性
for ul in soup.select('ul'):
    print(ul['id'])
    print(ul.attrs['class'])


### 获取内容
### get_text()

html='''
<div class="panel">
    <div class="panel-heading">
        <h4>Hello</h4>
    </div>
    <div class="panel-body">
        <ul class="list" id="list-1">
            <li class="element">Foo</li>
            <li class="element">Bar</li>
            <li class="element">Jay</li>
        </ul>
        <ul class="list list-small"
id="list-2">
            <li class="element2">Foo</li>
            <li class="element2">Bar</li>
        </ul>
    </div>
</div>
'''
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'lxml')
```

```
for li in soup.select('li'):
    print(li.string)
    print(li.get_text())  # 获取内容
```

* 推荐使用lxml解析库，必要时使用html.parser
* 标签选择筛选功能弱但是速度快
* 建议使用find()、find_all() 查询匹配单个结果或者多个结果
* 如果对CSS选择器熟悉建议使用select()
* 记住常用的获取属性和文本值的方法

数据提取之CSS选择器

css 语法概要

熟悉前端的同学对 css 选择器一定不会陌生，比如 jquery 中通过各种 css 选择器语法进行 DOM 操作等

> 学习网站：http://www.w3cmap.com/cssref/css-selectors.html

选择器	例子	例子描述	CSS
.class	.intro	选择 class="intro" 的所有元素。	1
#id	#firstname	选择 id="firstname" 的所有元素。	1
*	*	选择所有元素。	2
element	p	选择所有 <p> 元素。	1
element,element	div,p	选择所有 元素和所有 <p> 元素。	1
element element	div p	选择 元素内部的所有 <p> 元素。	1
element>element	div>p	选择父元素为 元素的所有 <p> 元素。	2
element+element	div+p	选择紧接在 元素之后的所有 <p> 元素。	2
[attribute]	[target]	选择带有 target 属性所有元素。	2
[attribute=value]	[target=_blank]	选择 target="_blank" 的所有元素。	2
[attribute~=value]	[title~=flower]	选择 title 属性包含单词 "flower" 的所有元素。	2
[attribute|=value]	[lang|=en]	选择 lang 属性值以 "en" 开头的所有元素。	2
:link	a:link	选择所有未被访问的链接。	1
:visited	a:visited	选择所有已被访问的链接。	1
:active	a:active	选择活动链接。	1
:hover	a:hover	选择鼠标指针位于其上的链接。	1
:focus	input:focus	选择获得焦点的 input 元素。	2
:first-letter	p:first-letter	选择每个 <p> 元素的首字母。	1
:first-line	p:first-line	选择每个 <p> 元素的首行。	1
:first-child	p:first-child	选择属于父元素的第一个子元素的每个 <p> 元素。	2
:before	p:before	在每个 <p> 元素的内容之前插入内容。	2
:after	p:after	在每个 <p> 元素的内容之后插入内容。	2
:lang(language)	p:lang(it)	选择带有以 "it" 开头的 lang 属性值的每个 <p> 元素。	2
element1~element2	p~ul	选择前面有 <p> 元素的每个 元素。	3
[attribute^=value]	a[src^="https"]	选择其 src 属性值以 "https" 开头的每个 <a> 元素。	3
[attribute$=value]	a[src$=".pdf"]	选择其 src 属性以 ".pdf" 结尾的所有 <a> 元素。	3
[attribute*=value]	a[src*="abc"]	选择其 src 属性中包含 "abc" 子串的每个 <a> 元素。	3
:first-of-type	p:first-of-type	选择属于其父元素的首个 <p> 元素的每个 <p> 元素。	3
:last-of-type	p:last-of-type	选择属于其父元素的最后 <p> 元素的每个 <p> 元素。	3
:only-of-type	p:only-of-type	选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。	3
:only-child	p:only-child	选择属于其父元素的唯一子元素的每个 <p> 元素。	3
:nth-child(n)	p:nth-child(2)	选择属于其父元素的第二个子元素的每个 <p> 元素。	3
:nth-last-child(n)	p:nth-last-child(2)	同上，从最后一个子元素开始计数。	3
:nth-of-type(n)	p:nth-of-type(2)	选择属于其父元素第二个 <p> 元素的每个 <p> 元素。	3
:nth-last-of-type(n)	p:nth-last-of-type(2)	同上，但是从最后一个子元素开始计数。	3
:last-child	p:last-child	选择属于其父元素最后一个子元素每个 <p> 元素。	3
:root	:root	选择文档的根元素。	3
:empty	p:empty	选择没有子元素的每个 <p> 元素（包括文本节点）。	3
:target	#news:target	选择当前活动的 #news 元素。	3
:enabled	input:enabled	选择每个启用的 <input> 元素。	3
:disabled	input:disabled	选择每个禁用的 <input> 元素	3
:checked	input:checked	选择每个被选中的 <input> 元素。	3
:not(selector)	:not(p)	选择非 <p> 元素的每个元素。	3
::selection	::selection	选择被用户选取的元素部分。	3

数据提取性能比较

选择器	性能	使用难度	安装难度
正则表达式	快	困难	简单（内置模块）
BeautifulSoup	慢	简单	简单（纯Python）
lxml	快	简单	相对困难

在爬虫中使用css选择器，代码教程

```
>>> from requests_html import session

# 返回一个Response对象
>>> r = session.get('https://python.org/')

# 获取所有链接
>>> r.html.links
{'/users/membership/',
'/about/gettingstarted/'}

# 使用css选择器的方式获取某个元素
>>> about = r.html.find('#about')[0]

>>> print(about.text)
About
Applications
Quotes
```

```
Getting Started
Help
Python Brochure
```

案例

目标网站：http://www.weather.com.cn/textFC/hb.shtml

需求：爬取全国的天气（获取城市以及最低气温）

> 1. 不同的url会涉及不同的数据
> 2. 先来获取第一个页面的数据
>
>   目标url: http://www.weather.com.cn/textFC/hz.shtml
>
>   思路分析：
>
>   1. 通过find方法，定位的div class=conMidtab2
>   2. 通过find_all方法，找所有的tr标签

```
函数功能
1. 得到网页源码
2. 解析数据
3. 保存数据
4. 主函数
```