

beautifulsoup

子标签：父标签的下一级

后代标签：父标签的下一级，下下一级，。。。

—class BeautifulSoup(网页的二进制编码, "html.parser")

也可以是这种格式BeautifulSoup(open('a.html', 'r'), "html.parser")直接打开文件中的html源码

第二个参数是解析器

中国大学MOOC

| 解析器 | 使用方法 | 条件 |
|--------------|---------------------------------|----------------------|
| bs4的HTML解析器 | BeautifulSoup(mk,'html.parser') | 安装bs4库 |
| lxml的HTML解析器 | BeautifulSoup(mk,'lxml') | pip install lxml |
| lxml的XML解析器 | BeautifulSoup(mk,'xml') | pip install lxml |
| html5lib的解析器 | BeautifulSoup(mk,'html5lib') | pip install html5lib |

FE:

```
>>> url = 'http://blog.fishc.com/'
>>> from bs4 import BeautifulSoup
>>> import urllib.request
>>> html = urllib.request.urlopen(url)
>>> bs = BeautifulSoup(html.read(), 'html.parser')
>>> bs
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"><h...
...
<div id="message">加载中.....</div>
<div class="mumu" id="mumu"></div>
</div>
</div></div>
```

=====基本属性=====

—BeautifulSoup类的基本元素

Beautiful Soup类的基本元素

| 基本元素 | 说明 |
|-----------------|---------------------------------------|
| Tag | 标签，最基本的信息组织单元，分别用<和</>标明开头和结尾 |
| Name | 标签的名字，<p>...</p>的名字是'p'，格式：<tag>.name |
| Attributes | 标签的属性，字典形式组织，格式：<tag>.attrs |
| NavigableString | 标签内非属性字符串，<...>中字符串，格式：<tag>.string |
| Comment | 标签内字符串的注释部分，一种特殊的Comment类型 |

—BeautifulSoup. 标签a(. 标签b....)

访问bs对象的后代标签中第一个标签a

可以使用BeautifulSoup. 标签a. name获得标签名字

可以使用BeautifulSoup. 标签a. parent. name多个parent获得上级的名字

FE:

```
>>> bs.title
```

```
<title>鱼C工作室 | 鱼C视频发布|编程教学|编程交流|视频教学|鱼C教学</title>
```

```
>>> bs.h1
```

```
<h1>
```

```
<a href="http://blog.fishc.com" title="返回 blog.FishC.com 首页">blog.FishC.com</a>
```

```
</h1>
```

```
>>> bs.h1.a
```

```
<a href="http://blog.fishc.com" title="返回 blog.FishC.com 首页">blog.FishC.com</a>
```

—BeautifulSoup. 标签a(. 标签b....)[属性]

查看标签属性

FE:

```
>>> bs.h1.a['href']
```

```
'http://blog.fishc.com'
```

—BeautifulSoup. attrs

获得标签组成的字典

因为是字典，所以提取的时候可以使用字典方法提取

FE:

```
>>> bs.h1.a.attrs
```

```
{'title': '返回 blog.FishC.com 首页', 'href': 'http://blog.fishc.com'}
```

—BeautifulSoup. get_text()/ .string

一般使用'.string' 返回标签里面的字符串，而且.string跨越多个标签属性，要是文本里面有
这一类的标签，都可以跨越

get_text() 不可以跨越

去除标签，返回文本对应字符串

FE:

```
>>> bs.h1.a.get_text()
```

```
'blog.FishC.com'
```

```
>>> bs.h1.get_text()
```

```
'\nblog.FishC.com\n'
>>> bs.h1.get_text()
'\nblog.FishC.com\n'
```

一注释处理

依靠type()判断数据类型

FE:

```
>>> newsoup = BeautifulSoup('<b><!--This is a comment--></b><p>This is not a comment.</p>', "html.parser")
>>> newsoup.b.string
'This is a comment'
>>> newsoup.b.get_text()
'',
>>> type(newsoup.b.string)
<class 'bs4.element.Comment'>
>>> type(newsoup.p.string)
<class 'bs4.element.NavigableString'>
```

遍历

一下行遍历



标签树的下行遍历

| 属性 | 说明 |
|--------------|----------------------------------|
| .contents | 子节点的列表，将<tag>所有儿子节点存入列表 |
| .children | 子节点的迭代类型，与.contents类似，用于循环遍历儿子节点 |
| .descendants | 子孙节点的迭代类型，包含所有子孙节点，用于循环遍历 |

一 BeautifulSoup.contents

返回列表

```
>>> soup = BeautifulSoup("<head><title>fef</title></head>", "html.parser")
>>> soup.head.contents
[<title>fef</title>]
```

一 BeautifulSoup.children

返回迭代器

FE:

```
>>> bs.find('div', {'id': 'container'}).div
<div id="header">
<div id="logo">
<h1>
<a href="http://blog.fishc.com" title="返回 blog.FishC.com 首页">blog.FishC.com</a>
</h1>
</div>
</div>
>>> bs.find('div', {'id': 'container'}).div.children
<list_iterator object at 0x000000E68967FC88>
>>> for i in bs.find('div', {'id': 'container'}).div.children:
print(i)
```

```

<div id="logo">
<h1>
<a href="http://blog.fishc.com" title="返回 blog.FishC.com 首页">blog.FishC.com</a>
</h1>
</div>

```

一上行遍历

| 标签树的上行遍历 | |
|----------|------------------------|
| 属性 | 说明 |
| .parent | 节点的父亲标签 |
| .parents | 节点先辈标签的迭代类型，用于循环遍历先辈节点 |

遍历parents时候，会遍历到 BeautifulSoup本身，而他本身不存在parent信息，即返回None，所以加个if，else。语句判断就好了

```

--BeautifulSoup.parent
>>> soup.title.parent
<head><title>fef</title></head>

```

一平行遍历

| 标签树的平行遍历 | |
|--------------------|------------------------------|
| 属性 | 说明 |
| .next_sibling | 返回按照HTML文本顺序的下一个平行节点标签 |
| .previous_sibling | 返回按照HTML文本顺序的上一个平行节点标签 |
| .next_siblings | 迭代类型，返回按照HTML文本顺序的后续所有平行节点标签 |
| .previous_siblings | 迭代类型，返回按照HTML文本顺序的前续所有平行节点标签 |

平行遍历，必须位于同一个父亲标签之下

```

--BeautifulSoup.next_siblings/previous_siblings
返回下一个标签/前一个标签，是generator类型，需要用for进行读取

```

=====漂亮输出=====

```

--BeautifulSoup.prettify()
自动添加'\n'，便于print打印输出!!!
FE:
>>>print(soup.a.prettify())

```

=====

—BeautifulSoup.find(A[,B])

A:可传入标签/标签组成的列表, ['h1',h2]

B:一个标签中对应的若干属性和属性值

FE:

```
>>> bs.find('li',{'class':'current'})
```

```
<li class="current">支持我们</li>
```

—BeautifulSoup.findAll/find_all([A][,B][,recursive=True][string=None])

A:可传入标签/标签组成的列表, ['h1',h2]

B:一个标签中对应的若干属性和属性值, 属性值可以是`re.compile()`构造的字符串

recursive:设置为False, 代表只查询下一级标签

string:可以只设置string的值, 检索标签之间的内容, 必须完全匹配。这里可以运用pyhton的正则表达式进行进一步构建!!! `re.compile(...)`

简写形式: BeautifulSoup(... 参数一样...)

FE:

```
>>> bs.findAll/find_all('li',{'class':'current'})
```

```
[<li class="current">支持我们</li>, <li class="current">发现精彩</li>]
```

```
>>>bs.findAll('','id':'text')也行
```

FE:

```
>>>images = bs.findAll('img',{'src':re.compile('...')})
```

```
>>>for image in images:
```

```
    print(image)
```

```
>>> for i in soup(re.compile('b')):
```

```
    print(i.name)
```

body

br

br

—拓展方法

扩展方法

| 方法 | 说明 |
|--|---|
| <code><.find()</code> | 搜索且只返回一个结果, 字符串类型, 同 <code>.find_all()</code> 参数 |
| <code><.find_parents()</code> | 在先辈节点中搜索, 返回列表类型, 同 <code>.find_all()</code> 参数 |
| <code><.find_parent()</code> | 在先辈节点中返回一个结果, 字符串类型, 同 <code>.find()</code> 参数 |
| <code><.find_next_siblings()</code> | 在后续平行节点中搜索, 返回列表类型, 同 <code>.find_all()</code> 参数 |
| <code><.find_next_sibling()</code> | 在后续平行节点中返回一个结果, 字符串类型, 同 <code>.find()</code> 参数 |
| <code><.find_previous_siblings()</code> | 在前序平行节点中搜索, 返回列表类型, 同 <code>.find_all()</code> 参数 |
| <code><.find_previous_sibling()</code> | 在前序平行节点中返回一个结果, 字符串类型, 同 <code>.find()</code> 参数 |

