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

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

--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</br>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'

--注释处理

依靠type()判断数据类型

FE：

>>> newsoup = BeautifulSoup('<!--This is a comment--> This is not a comment. ',"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==========

--下行遍历

-- BeautifulSoup.contents

返回列表

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

>>> soup.head.contents

[<title>fef</title>]

--BeautifulSoup.children

返回迭代器

FE：

>>> bs.find('div',{'id':'container'}).div

<h1>

blog.FishC.com

</h1>

>>> bs.find('div',{'id':'container'}).div.children

<list_iterator object at 0x000000E68967FC88>

>>> for i in bs.find('div',{'id':'container'}).div.children:

 print(i)

<h1>

blog.FishC.com

</h1>

--上行遍历

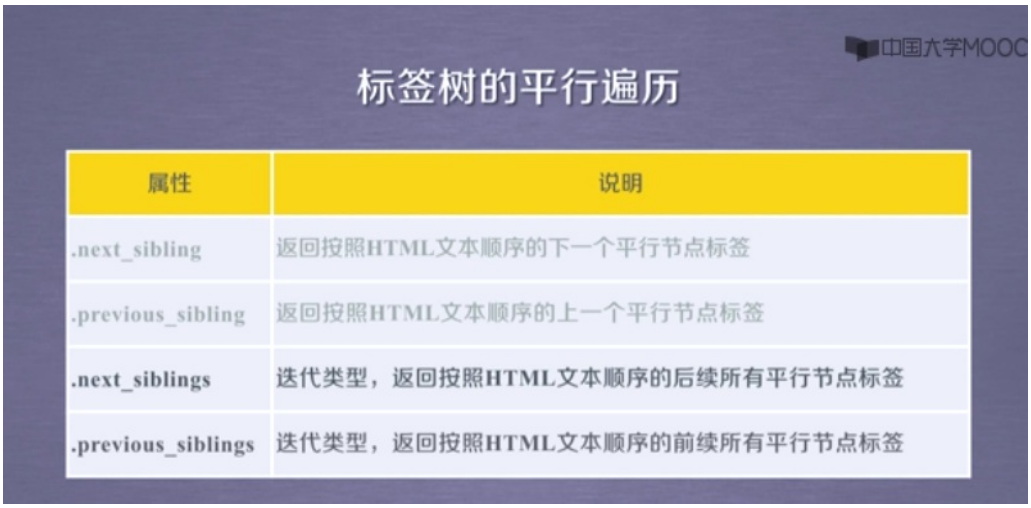
遍历parents时候，会遍历到 BeautifulSoup本身，而他本身不存在parent信息，即返回None，所以加个if，else。语句判断就好了

--BeautifulSoup.parent

>>> soup.title.parent

<head><title>fef</title></head>

--平行遍历



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

--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 class="current">发现精彩]

>>>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