3.4 HTML解析与Python实现

3.4.1 HTML与CSS要素

3.4.2 BeautifulSoup库及对象

3.4.3 BeautifulSoup库遍历文档树

3.4.4 BeautifulSoup库搜索文档树

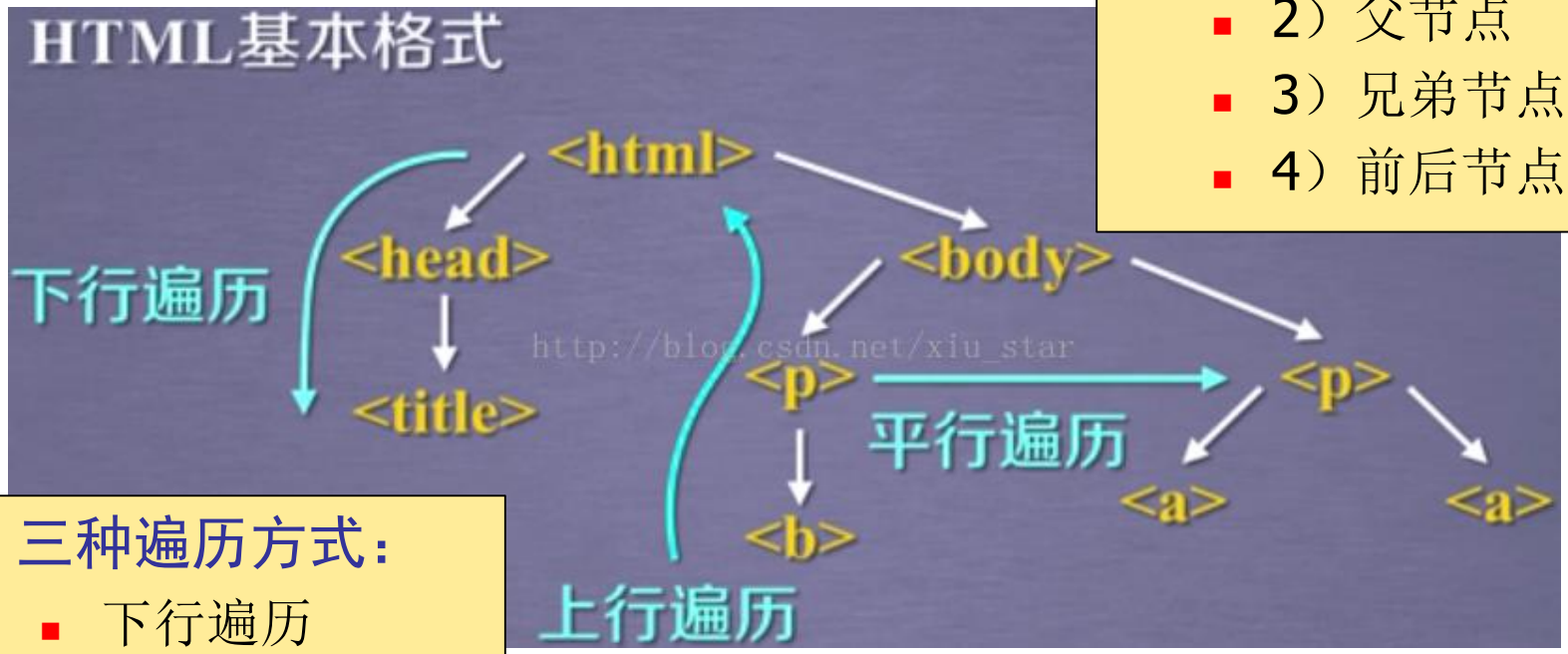
3.4.5 BeautifulSoup库查找CSS过滤器

3.4.6 Python解析网页实例

3.4.3 BeautifulSoup库遍历文档树

- BeautifulSoup会将HTML转化为文档树进行搜索、遍历
  https://blog.csdn.net/jia666666/article/details/82107815
  https://www.jb51.net/article/159136.htm
  https://blog.csdn.net/su749520/article/details/79243950

- 文档树节点类型：
  - 1）子节点
  - 2）父节点
  - 3）兄弟节点
  - 4）前后节点



- 三种遍历方式：
  - 下行遍历
  - 上行遍历
  - 平行遍历

3.4.3 BeautifulSoup库遍历文档树

1）子节点

- Tag包含的多个字符串或者其他的Tag，都是其子节点
- BeautifulSoup提供了许多操作和遍历子节点的属性。
- 标签树的下行遍历时，子节点属性：
  - Tag.contents属性：以列表形式返回所有子节点
  - Tag.children:迭代器/生成器，可用于循环访问,遍历儿子节点
  - Tag.descendants属性：contents 和 .children 属性仅包含tag的直接子节点，.descendants 属性可以对所有tag的子孙节点进行递归循环

- 注意！BeautifulSoup中的字符串节点不支持这些属性，因为字符串没有子节点

3.4.3 BeautifulSoup库遍历文档树

- 容器
  - 由多个对象组成的结构。
  - 比如：列表[0,1,2]，元组(1,2,3)，字典{a:1,b:2} ，集合{1,2,3}都是容器
- 迭代器（Iterator对象）
  - 所有的容器都是可迭代对象，也就是可以遍历元素
  - 可以被转化为不依赖索引取值的容器
  - 一个数据流，或是一个有序序列，不能提前知道序列的长度，只有在需要返回下一个数据时它才会计算；
- 生成器
  - 就是一个迭代器的例子，如果说迭代器是人，那么生成器就人中的一个人。

- 以下文为例子（bs_tree.py)

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc)
```

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

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names w
ere
<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>
"""
```

```
soup.head
# <head><title>The Dormouse's story</title></head>

soup.title
# <title>The Dormouse's story</title>
```

按标签获取对象

tag的.contents属性可以将tag的子节点以**列表的方式输出**:

```
1  head_tag = soup.head
2  head_tag
3  # <head><title>The Dormouse's story</title></head>
4
5  head_tag.contents
6  [<title>The Dormouse's story</title>]
7
8  title_tag = head_tag.contents[0]
9  title_tag
10 # <title>The Dormouse's story</title>
11 title_tag.contents
12 # [u'The Dormouse's story']
```

列表[ ]

字符串（不带[ ]）

字符串没有子节点, 没有.contents属性

```
text = title_tag.contents[0]
text.contents
# AttributeError: 'NavigableString' object has no attribute 'contents'
```

- 用tag的.children生成器，对tag的子节点进行循环 bs_tree.html

```
# 待分析字符串
html_doc = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a time there were three little sis and their names were
        <a href="http://example.com/elsie" class="sister" id="link1">El
        <a href="http://example.com/lacie" class="sister" id="link2">La
        and
        <a href="http://example.com/tillie" class="sister" id="link3">T
        and they lived at the bottom of a well.
    </p>

    <p class="story">...</p>
    </body>
</html>
"""
```

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
print soup.body.children
2 #<listiterator object at 0x7f71457f5710>
```

列表生成器，需要循环遍历

```
# 待分析字符串
html_doc = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a
        <a href="http://example.c
        <a href="http://example.c
        and
        <a href="http://example.c
        and they lived at the bot
    </p>

    <p class="story">...</p>
    </body>
</html>
"""
```

```
1 for child in  soup.body.children:
2       print child
```

运行结果

```
<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 class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>,

<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and

<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>


<p class="story">...</p>
```

- Tag.descendants：生成器，对所有tag的子孙节点进行递归循环，遍历获取其中的内容

```
soup.body.
for child in soup.descendants:
    print child
```

运行结果

```
<html><head><title>Th
<body>
<p class="title" name
<p class="story">Once
<a class="sister" hre
<a class="sister" hre
<a class="sister" hre
and they lived at the
<p class="story">...<
</body></html>
<head><title>The Dorm
<title>The Dormouse's story</titl
The Dormouse's story
<body>
<p class="title" name="dromouse"><b>The
<p class="story">Once upon a time there
<a class="sister" href="http://example.
<a class="sister" href="http://example.
<a class="sister" href="http://example.
and they lived at the bottom of a well.
<p class="story">...</p>
</body>
```

```
<p class="title" name="dromouse"><b>The Dormouse's sto
<b>The Dormouse's story</b>
The Dormouse's story


<p class="story">Once upon a time there were three little sisters; and their names were
<a class="
<a class="
<a class="
and they l
Once upon
```

```
<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>
 Elsie
,

<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>
Lacie
 and

<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
Tillie
;
and they lived at the bottom of a well.


<p class="story">...</p>
...
```

3.4.3 BeautifulSoup库遍历文档树

- 获取子节点的内容，涉及三个属性：
  - .string
    - 返回单个文本内容。
    - 如果一个标签里面没有标签了，返回标签里面的内容。
    - 如果标签里面只有唯一的一个标签了，也会返回最里面的内容
    - 如果tag包含了多个子节点, 无法确定应该调用哪个子节点的内容，输出结果是 None
  - .strings
    - 返回多个文本内容，且包含空行和空格
  - stripped_strings
    - 返回多个文本内容，且不包含空行和空格

- Tag.String：Tag只有一个String子节点时，可以这么访问，否则返回None

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
#
html_doc = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a time there were three little sisters; and t
        <a href="http://example.com/elsie" class="sister" id="link1">Elsie</
        <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>
    </body>
</html>
"""
```

```
1 print soup.head.string
2 #The Dormouse's story
3 print soup.title.string
4 #The Dormouse's story
```

两句都只有一个子节点内容

```
1 print soup.html.string
2 # None
```

多个子节点，无法确定

Tag.strings：获取多个内容，需要遍历获取

```
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a time there were
        <a href="http://example.com/elsie" class
        <a href="http://example.com/lacie" class
        and
        <a href="http://example.com/tillie" clas
        and they lived at the bottom of a well.
    </p>

    <p class="story">...</p>
    </body>
</html>
"""
```

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
for string in soup.strings:
    print(repr(string))
    # u"The Dormouse's story"
    # u'\n\n'
    # u"The Dormouse's story"
    # u'\n\n'
    # u'Once upon a time there were three l
    # u'Elsie'
    # u',\n'
    # u'Lacie'
    # u' and\n'
    # u'Tillie'
    # u';\nand they lived at the bottom of
    # u'\n\n'
    # u'...'
    # u'\n'
```

3.4.3 BeautifulSoup库遍历文档树

- repr() 函数
  - Python的内置函数
  - 将对象转化为供解释器读取的形式
- 语法
  - repr(object)
  - 参数：object -- 对象。
  - 返回值：一个对象的 string 格式。

```
>>>s = 'RUNOOB'
>>> repr(s)
"'RUNOOB'"
>>> dict = {'runoob': 'runoob.com', 'google': 'google.com'};
>>> repr(dict)
"{'google': 'google.com', 'runoob': 'runoob.com'}"
>>>
```

- Tag.stripped_strings：去除输出的字符串中多余空白

```
# 待分析字符串
html_doc = """
<html>
    <head>
        <title>The Dormouse's st
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a
        <a href="http://example.
        <a href="http://example.
        and
        <a href="http://example.
        and they lived at the bo
    </p>

    <p class="story">...</p>
    </body>
</html>
"""
```

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
for string in soup.stripped_strings:
    print(repr(string))
    # u"The Dormouse's story"
    # u"The Dormouse's story"
    # u'Once upon a time there were three little si
    # u'Elsie'
    # u','
    # u'Lacie'
    # u'and'
    # u'Tillie'
    # u';\nand they lived at the bottom of a well.'
    # u'...'
```

3.4.3 BeautifulSoup库遍历文档树

2）父节点

- 每个tag或字符串都有父节点，在文档的树形结构中，根节点是BeautifulSoup对象
- 标签树的上行遍历时，节点的属性：
  - .parent属性：节点的父标签
  - .parents属性：节点先辈标签的迭代类型，用于循环遍历先辈节点

- 注意：在遍历一个标签的所有先辈标签时，会遍历到soup本身，而soup的先辈不存在（也就是None），因此也就没有.name信息

- Tag.parent：返回某节点的直接父节点

```
# 待分析字符串
html_doc = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a time there we
        <a href="http://example.com/elsie" cla
        <a href="http://example.com/lacie" cla
        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>
    </body>
</html>
"""
```

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
p = soup.p
print p.parent.name
#body


content = soup.head.title.string
print content.parent.name
#title
```

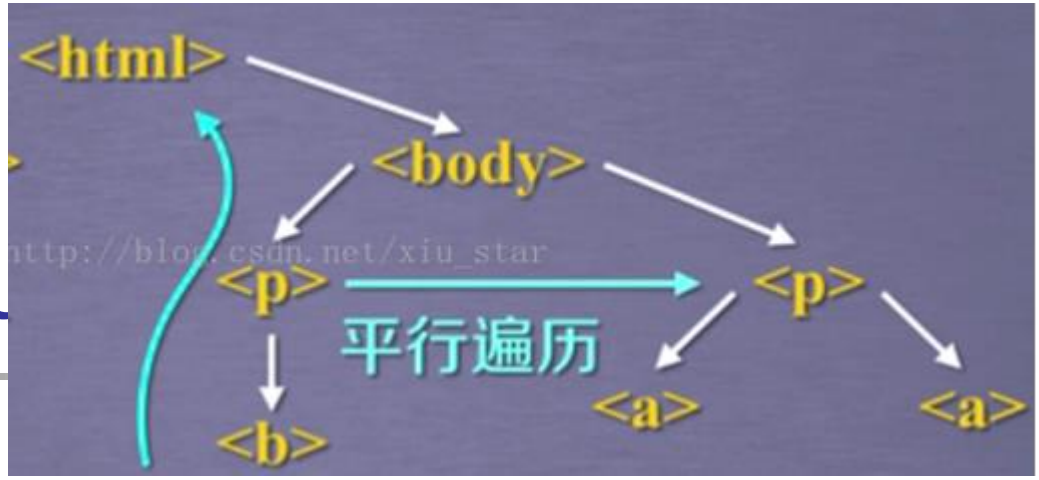
- Tag.parents ： 返回某节点的所有父辈以及上辈的节点

```
# 待分析字符串
html_doc = """
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
    <p class="title aq">
        <b>
            The Dormouse's story
        </b>
    </p>

    <p class="story">Once upon a time ther
        <a href="http://example.com/elsie"
        <a href="http://example.com/lacie"
        and
        <a href="http://example.com/tillie
        and they lived at the bottom of a
    </p>

    <p class="story">...</p>
    </body>
</html>
"""
```

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

```
content = soup.head.title.string
for parent in  content.parents:
    print parent.name


title
head
html
[document]
```

3.4.3 Beautifu

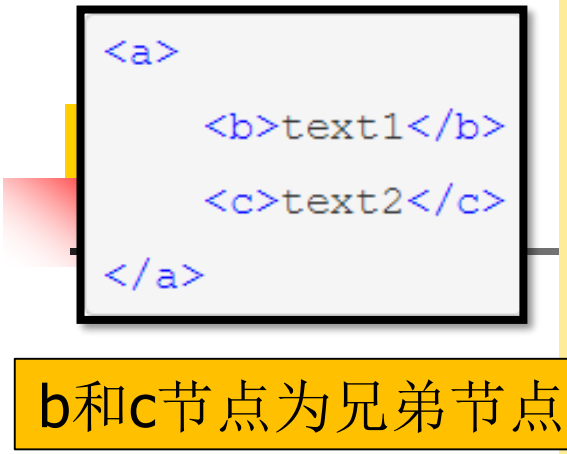


3）兄弟节点

- 和本节点处在同一级的节点称为兄弟节点
- 标签树平行遍历时，节点属性：

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

- 注意：平行遍历必须发生在同一个父节点下的各节点间



b和c节点为兄弟节点

- Tag.next_sibling、Tag.previous_sibling
  - 获取节点的下一个/前一个兄弟节点，如果节点不存在，为None
  - 通常兄弟节点是字符串或空白，因为空白或者换行也可以被视作一个节点，所以得到的结果可能是空白或者换行。

```
sibling_soup = BeautifulSoup("<a><b>text1</b><c>text2</c></b></a>")
sibling_soup.b.next_sibling
# <c>text2</c>
sibling_soup.c.previous_sibling
# <b>text1</b>
```

自己试试，体会！

```
link
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
link.next_sibling
# u',\n'
```

- Tag.next_siblings 、Tag.previous_siblings：
  - 对当前节点的兄弟节点迭代输出（全部兄弟节点）

自己试试，体会！

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

```
for sibling in soup.a.next_siblings:
    print(repr(sibling)) # u',\n'
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>
# u' and\n'
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
# u'; and they lived at the bottom of a well.'
# None
```

- Tag.next_siblings 、Tag.previous_siblings：
  - 对当前节点的兄弟节点迭代输出（全部兄弟节点）

自己试试，体会！

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

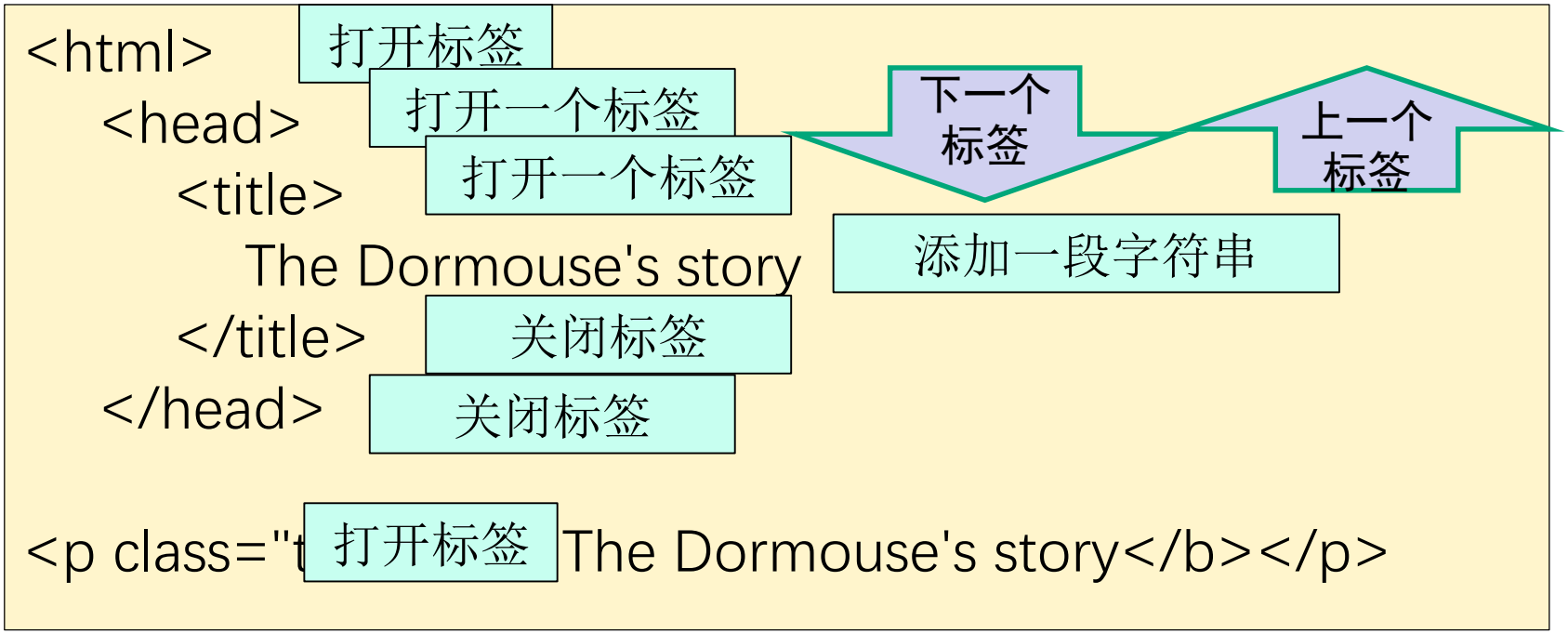
```
for sibling in soup.find(id="link3").previous_siblings:
    print(repr(sibling))
```

```
# ' and\n'
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>
# u',\n'
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
# u'Once upon a time there were three little sisters; and their names were\n'
# None
```

3.4.3 BeautifulSoup库遍历文档树

4）前后节点（回退和前进）

- 分析HTML解析器：把这段字符串转换成一连串的事件



- Beautiful Soup提供了重现解析器初始化过程的方法。

3.4.3 BeautifulSoup库遍历文档树

4）前后节点（回退和前进）

https://www.cnblogs.com/cxchanpin/p/7053776.html

- .next_element和.previous_element：
  - 解析过程中下一个和上一个被解析的对象(字符串或者tag)
  - 不针对于兄弟节点，而是针对于所有节点，不分层次的前一个和后一个节点(结果可能与.next_siblings相同，但通常是不一样的)
- .next_elements和.previous_elements
  - 迭代获取所有前和后节点
  - 通过它们就可以向前或者向后访问文档的解析内容，就好像文档正在被解析一样

3.4.3 BeautifulSoup库遍历文档树

- 标签的.next_element属性结果是在标签被解析之后的解析内容，不是标签后的句子部分。
- .previous_element属性刚好与.next_silbing相反，它指向当前被解析的对象的前一个解析对象。

```
<head><title>The Dormouse's story</title></head>
```

```
print soup.head.next_element

#<title>The Dormouse's story</title>
```

自己试试，体会！

不是针对于兄弟节点，而是所有节点，不分层次

3.4.3 BeautifulSoup库遍历文档树

- .next_elements 和 .previous_elements :
  - 迭代器，可以向前或向后访问文档的解析内容，就好像文档正在被解析一样

```
for element in last_a_tag.next_elements:
    print(repr(element))
# u'Tillie'
# u';\nand they lived at the bottom of a well.'
# u'\n\n'
# <p class="story">...</p>
# u'...'
# u'\n'
# None
```

自己试试，体会！

3.4 HTML解析与Python实现

3.4.1 HTML与CSS要素

3.4.2 BeautifulSoup库及对象

3.4.3 BeautifulSoup库遍历文档树

3.4.4 BeautifulSoup库搜索文档树

3.4.5 BeautifulSoup库查找CSS过滤器

3.4.6 Python解析网页实例

3.4.4 BeautifulSoup库搜索文档树

（https://blog.csdn.net/qq_22592457/article/details/95191430）

- BeautifulSoup定义了很多搜索方法，最常用的是find_all()、find()

```
find_all(tag, attributes, recursive, text, limit, keywords)
find(tag, attributes, recursive, text, keywords)
```

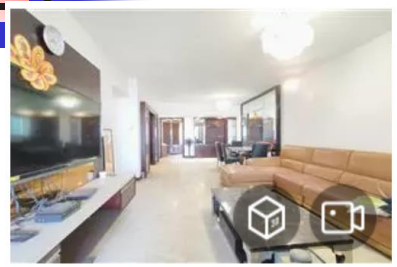
- 功能：搜索当前tag的(所有)tag子节点
- 二者的区别：
  - find_all: 根据范围限制参数limit限定的范围，取满足条件的元素，组成一个list
    - (默认：不设置limit，代表取所有符合要求的元素)
  - find：只取符合要求的第一个元素，等价于 find_all的limit =1 时的情形

find_all(tag, attributes, recursive, text, limit, keywords)
(标签, 属性, 递归, 文本, 限制, 关键词）

- 参数：相当于过滤器一样可以进行筛选处理
  - tag: 标签名称组成的set对象，取值类型为字符串、正则表达式、列表、True和方法；
  - attributes:字典类型，封装一个标签的若干属性和对应的属性值；
  - Recursive(少用):布尔变量。标识是否查找标签的子标签；True（默认值），查找所有标签，以及标签的子标签；False，只查找文档的一级标签；
  - text：用标签的文本内容去匹配，而不是用标签的属性；取值类型为字符串 、正则表达式 、列表、True;
  - limit: 限制返回结果的数量; find_all() 方法返回全部的搜索结构,如果文档树很大,用 limit 参数限制返回结果
  - keyword:可选指定属性的标签；如果一个指定名字的参数不是搜索内置的参数名,搜索时会把该参数当作指定名字tag的属性来搜索,如果包含一个名字为 id 的参数,Beautiful Soup会搜索每个tag的”id”属性;如果是class、id等参数，用keywords 或者attributes用法一样，如果是一些其他参数，则用keywords

1.标签tag用法https://blog.csdn.net/qq_22592457/article/details/95191430

- 例1，以中原地产二手房网页面举例（gz.centanet.com/ershoufang/）
- 编程获取房源信息



广州雅居乐花园十年小雅 四房带车位

广州雅居乐花园十年小雅 | 4室2厅 | 149平

中层/33层 东北 简装 2010年

番禺-广州雅居乐 南村镇兴南大道398号

距7号线员岗站663米 随时可看

630万

42282元/平

小区均价：43299元/平

翠湖南向三房 采光通风好 可用公积金贷 随时看房...

祈福新村E区 | 3室2厅 | 96平

中层/6层 南 简装 2002年

番禺-祈福 钟村市广路与金山大道交界处

满五 唯一住房

338万

35208元/平

小区均价：29123元/平

富豪山庄景峰豪庭 南北对流 豪华装修 可拎包入住...

富豪山庄景峰豪庭 | 4室2厅 | 159.6平

中层/11层 南北 简装 2014年

番禺-金山谷 东环街市广路

全网热榜 满二 降价

530万

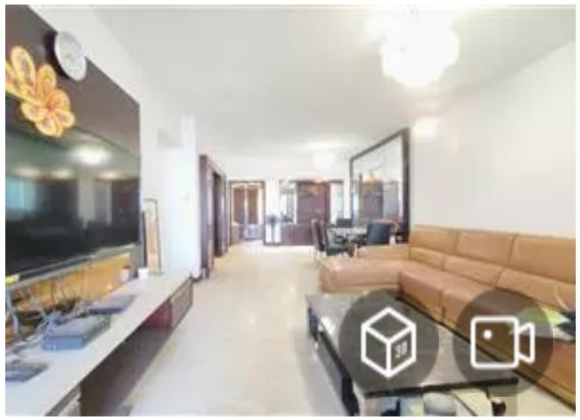
33208元/平

小区均价：35095元/平

1）采用开发者工具，观察html，获取页面上的标题

选中每个房源标题→右键菜单→检查

弹出对应代码，查找规律



广州雅居乐花园十年小雅 四房

广州雅居乐花园十年小雅 | 4室2厅 |

中层/33层 东北 简装 2010年

番禺-广州雅居乐 南村镇兴南大道398

距7号线员岗站663米 随时可看

在新标签页中打开链接(T)

在新窗口中打开链接(W)

在隐身窗口中打开链接(G)

链接另存为(K)...

复制链接地址(E)

检查(N) Ctrl+Shift+I

广州雅居乐花园十年小

广州雅居乐花园十年小雅 四房带车位

中层/33层 东北 简装

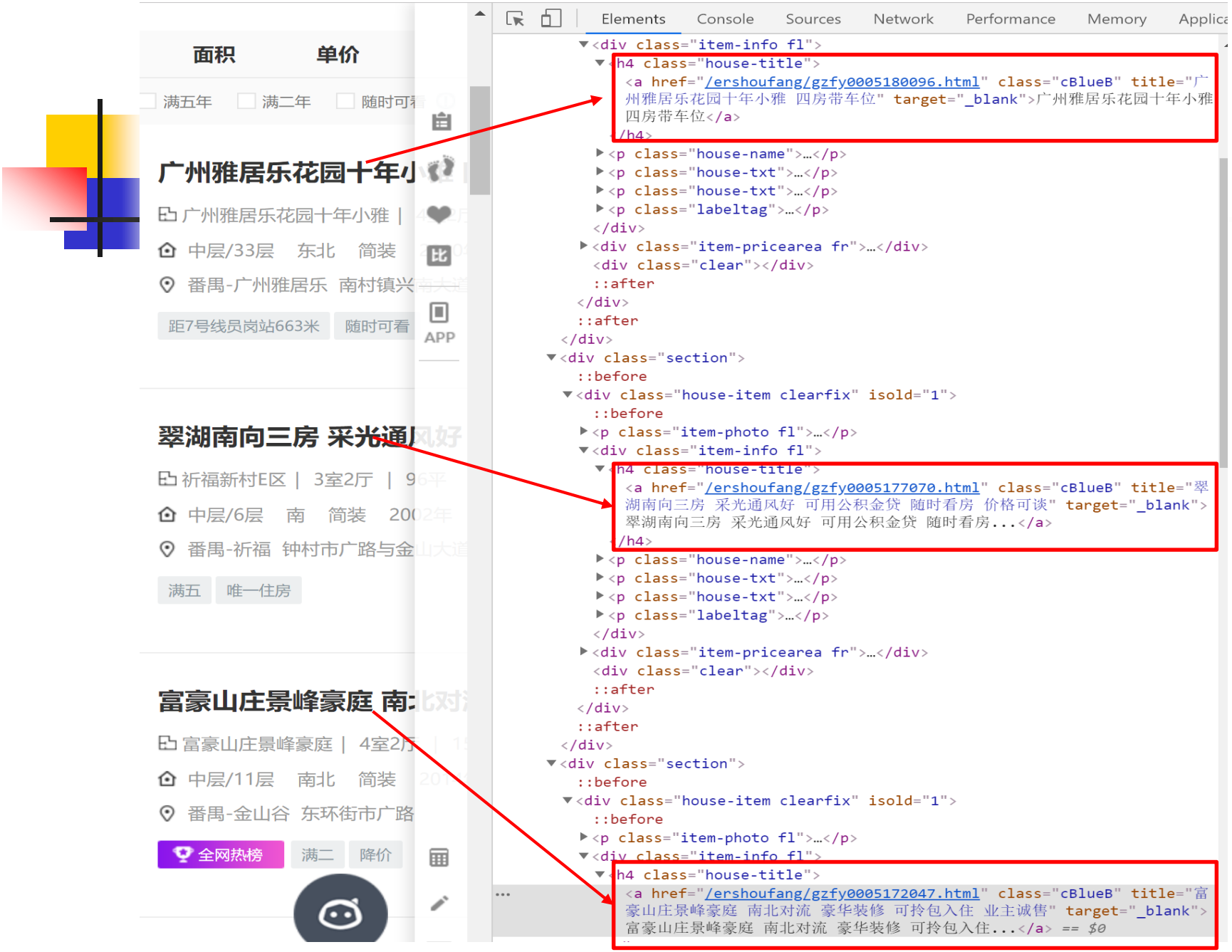
番禺-广州雅居乐 南村镇兴

距7号线员岗站663米 随时可看

APP

翠湖南向三房 采光通风好

```
<script type="text/template" id="suggestListTemplete">…</script>
<script type="text/template" id="hotKeyTemplete">…</script>
<div class="section-wrap section-breadcrumb visible-desktop">…</div>
<div class="section-wrap section-condition">…</div>
<div class="section-wrap section-select">…</div>
<div class="section-wrap section-houselists">
  <div class="section">
    ::before
    <div class="house-item clearfix curr" isold="1">
      ::before
      <p class="item-photo fl">…</p>
      <div class="item-info fl">
        <h4 class="house-title">
          <a href="/ershoufang/gzfy0005180096.html" class="cBlueB" title="广州雅居乐花园十年小雅 四房带车位" target="_blank">广州雅居乐花园十年小雅 四房带车位</a> == $0
```


面积
单价
满五年
满二年
随时可看
广州雅居乐花园十年小
广州雅居乐花园十年小雅 |
中层/33层 东北 简装
番禺-广州雅居乐 南村镇兴
距7号线员岗站663米
随时可看
APP
翠湖南向三房 采光通
祈福新村E区 | 3室2厅 |
中层/6层 南 简装
番禺-祈福 钟村市广路与金
满五
唯一住房
富豪山庄景峰豪庭 南
富豪山庄景峰豪庭 | 4室2厅
中层/11层 南北 简装
番禺-金山谷 东环街市广路
全网热榜
满二
降价
Elements
Console
Sources
Network
Performance
Memory

<h4 class="house-title">
广州雅居乐花园十年小雅 四房带车位
</h4>
…
…
…
…

…

::after

::after

::before

::before
…

<h4 class="house-title">
翠湖南向三房 采光通风好 可用公积金贷 随时看房...
</h4>
…
…
…
…

…

::after

::after

::before

::before
…

<h4 class="house-title">
富豪山庄景峰豪庭 南北对流 豪华装修 可拎包入住... == $0

```
▼<h4 class="house-title">
    <a href="/ershoufang/gzfy0005180096.html" class="cBlueB" title="广州雅居乐花园十年小雅 四房带车位" target="_blank">广州雅居乐花园十年小雅四房带车位</a>
  </h4>
```

```
▼<h4 class="house-title">
    <a href="/ershoufang/gzfy0005177070.html" class="cBlueB" title="翠湖南向三房 采光通风好 可用公积金贷 随时看房 价格可谈" target="_blank">翠湖南向三房 采光通风好 可用公积金贷 随时看房...</a>
  </h4>
```

```
▼<h4 class="house-title">
    <a href="/ershoufang/gzfy0005172047.html" class="cBlueB" title="富豪山庄景峰豪庭 南北对流 豪华装修 可拎包入住 业主诚售" target="_blank">富豪山庄景峰豪庭 南北对流 豪华装修 可拎包入住...</a> == $0
  </h4>
```

- 房源信息对应的tag是h4

3.4.4 BeautifulSoup库搜索文档树

编程：（bs_find3.4.4-h4.py)

```
from bs4 import BeautifulSoup
import requests

url = 'https://gz.centanet.com/ershoufang/'
urlhtml=requests.get(url)
urlhtml.encoding='utf-8'
soup=BeautifulSoup(urlhtml.text, 'html.parser')

alink = soup.find_all('h4')
print(alink)
```

或lxml

运行结果

[<h4 class="house-title">
广州雅居乐花园十年小雅 四房带车位
</h4>, <h4 class="house-title">
犀牛路小区精装修24h轮岗治安管理好装修保养好方...
</h4>, <h4 class="house-title">
翠湖南向三房 采光通风好 可用公积金贷 随时看房...
</h4>, <h4 class="house-title">
祈福新村康怡居 实用两房 南向中层 带车位
</h4>, <h4 class="house-title">
祈福新村翠怡居 南北对流 通风采光好 无遮挡 业...
</h4>, <h4 class="house-title">
富豪山庄景峰豪庭 南北对流 豪华装修 可拎包入住...
</h4>, <h4 class="house-title">
蝶舞轩 大四房 采光好 周边配套齐全 生活方便 居...
</h4>, <h4 class="house-title">
西华路金花街社区，南北对流三房准电梯
</h4>, <h4 class="house-title">
文化大院 南北对流两房 厅出大阳台 安静舒适 方...
</h4>, <h4 class="house-title">
南田路散 繁华地段 交通便捷配套成熟户型方正光...
</h4>, <h4 class="house-title">
花园新村一房一厅 户型方正 采光通透 装修精美 ...
</h4>, <h4 class="house-title">
急售四房 采光通风好，东南向望花园，全屋精装欢迎...
</h4>, <h4 class="house-title">
水荫横路 实用两房 水荫校区 有匙即看
</h4>, <h4 class="house-title">
滨江西路电梯房出租家电齐全，看房提前联系
</h4>, <h4 class="house-title">
广九大马路堤畔街

- 类似，以中原地产优质新房（https://gz.centanet.com/xinfang/）
- 编程获取房源信息
- 分析源代码，发现房源信息对应的tag是h5

3.4.4 BeautifulSoup库搜索文档树

编程：（bs_find3.4.4-h5.py)

```
from bs4 import BeautifulSoup
import requests

url = ‘https://gz.centanet.com/新fang/'
urlhtml=requests.get(url)
urlhtml.encoding='utf-8'
soup=BeautifulSoup(urlhtml.text, 'html.parser')

alink = soup.find_all('h5')
print(alink)
```

或lxml

运行结果

```
[<h5>
<i class="icons-hot"></i>
<a href="/xinfang/lp-11098/" target="_blank" title="天健云山府">天健云山府</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-10495/" target="_blank" title="敏捷·绿湖首府">敏捷·绿湖首府</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<i class="icons-hot"></i>
<a href="/xinfang/lp-12319/" target="_blank" title="南沙保利天汇">南沙保利天汇</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-12364/" target="_blank" title="时代大家(时代江岸花园)">时代大家(时代江岸花园)</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-10577/" target="_blank" title="越秀星汇云城">越秀星汇云城</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-10458/" target="_blank" title="富力悦禧城">富力悦禧城</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-10938/" target="_blank" title="香江天赋">香江天赋</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-10987/" target="_blank" title="西福蓝湾">西福蓝湾</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
<a href="/xinfang/lp-12375/" target="_blank" title="保利悦公馆">保利悦公馆</a>
<label class="yellow_tag" for="">在售</label>
<label class="red_tag" for="">住宅</label>
</h5>, <h5>
```

3.4.4 BeautifulSoup库搜索文档树

find_all(tag, attributes, recursive, text, limit, keywords)

- tag参数可以查找所有指定名字为 name 的tag

```
1  #第一个参数为Tag的名称
2  tag.find_all('title')
3  #得到"<title>&%^&*</title>",结果为一个列表
4
5  第二个参数为匹配的属性
6  tag.find_all("title",class="sister")
7  #得到如"<title class = "sister">%^*&</title>
8
9  # 第二个参数也可以为字符串，得到字符串匹配的结果
10 tag.find_all("title","sister")
11 #得到如"<title class = "sister">%^*&</title>
```

3.4.4 BeautifulSoup库搜索文档树

- A.Tag值为字符串
  - 在搜索方法中传入一个字符串参数
  - Beautiful Soup会查找与字符串完整匹配的内容

```
1 soup.find_all('b')

2 # [<b>The Dormouse's story</b>]

3

4 print soup.find_all('a')

5 #[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>, <a class="sister"
href="http://example.com/lacie" id="link2">Lacie</a>, <a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>]
```

查找文档中所有的<b>标签

查找文档中所有的<a>标签

- B.Tag值为正则表达式
  - Beautiful Soup会通过正则表达式的 match() 来匹配内容.

```
import re
for tag in soup.find_all(re.compile("^b")):
    print(tag.name)
# body
# b
```

查找文档中所有的<b>标签

- C. Tag值为列表
  - Beautiful Soup会将与列表中任一元素匹配的内容返回.

```
soup.find_all(["a", "b"])
# [<b>The Dormouse's story</b>,
#  <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

查找文档中所有<a>标签和<b>标签

- D.Tag值为True
  - True 可以匹配任何值

```
for tag in soup.find_all(True):
    print(tag.name)
# html
# head
# title
# body
# p
# b
# p
# a
# a
```

查找到所有的tag,但是不会返回字符串节点

- E. Tag值为方法
  - 方法只接受一个元素参数
  - 如果这个方法返回 True 表示当前元素匹配并且被找到,否则返回 False

判断当前对象：如果包含 class 属性却不包含 id 属性,那么将返回 True:

```
def has_class_but_no_id(tag):
    return tag.has_attr('class') and not tag.has_attr('id')
```

方法作为参数

```
soup.find_all(has_class_but_no_id)
# [<p class="title"><b>The Dormouse's story</b></p>,
#  <p class="story">Once upon a time there were...</p>,
#  <p class="story">...</p>]
```

结果含class，不含id

2.属性attributes用法

- 用字典封装一个标签的若干属性和对应的属性值。

- 例. 要获取选中的信息，通过观察可知，信息属性"house-name", "house-txt"，标签为p

```
▶<h4 class="house-title">…</h4>
▼<p class="house-name">
   <i class="icons-type"></i>
   <a href="/xiaoqu/xq-0200014645/" title="广州雅居乐花园十年小雅二手房">广州雅居乐花园十年小雅</a>
   <span class="line">|</span> == $0
   <span>4室2厅</span>
   <span class="line">|</span>
  ▼<span>
     "149平"
     <span class="f666"></span>
   </span>
 </p>
▼<p class="house-txt">
   <i class="icons-house"></i>
   <span>中层/33层</span>
   <span>东北</span>
   <span>简装</span>
   <span>2010年</span>
```

编程：（bs_find3.4.4-attr.py）

```
from bs4 import BeautifulSoup
import requests
url = 'https://gz.centanet.com/ershoufang/'
urlhtml=requests.get(url)
urlhtml.encoding='utf-8'
soup=BeautifulSoup(urlhtml.text, 'html.parser')
alink = soup.find_all("p", {"class":{"house-name", "house-txt"}})
print(alink)
```

运行结果

```
[<p class="house-name">
<i class="icons-type"></i><a href="/xiaoqu/xq-0200014645/" title="广州雅居乐花园十年小雅二手房">广州雅居乐花园十年小雅</a><span class="line">|</span>
<span>4室2厅</span>
<span class="line">|</span>
<span>149平<span class="f666"></span></span>
</p>, <p class="house-txt">
<i class="icons-house"></i> <span>中层/33层</span> <span>东北</span> <span>简装</span> <span>2010年</span>
</p>, <p class="house-txt">
<i class="icons-address"></i>
<a href="/ershoufang/panyuqu/">番禺</a>-<a href="/ershoufang/guangzhouyajule/">广州雅居乐</a>
                南村镇兴南大道398号
            </p>, <p class="house-name">
<i class="icons-type"></i><a href="/xiaoqu/xq-0200025013/" title="犀牛路散盘二手房">犀牛路散盘</a><span class="line">|</span>
<span>1室1厅</span>
<span class="line">|</span>
<span>23平<span class="f666"></span></span>
</p>, <p class="house-txt">
<i class="icons-house"></i> <span>高层/7层</span> <span>三面单边</span> <span>简装</span> <span>1990年</span>
</p>, <p class="house-txt">
<i class="icons-address"></i>
<a href="/ershoufang/yuexiuqu/">越秀</a>-<a href="/ershoufang/huanshidong/">环市东</a>
                犀牛路
            </p>, <p class="house-name">
<i class="icons-type"></i><a href="/xiaoqu/xq-0200010376/" title="祈福新村E区二手房">祈福新村E区</a><span class="line">|</span>
<span>3室2厅</span>
```

3.文本text用法

- 用标签的文本内容去匹配，而不是用标签的属性。
- 例，在中原房产网页中，查找户型为“2室1厅”的有多少个

3.4.4 BeautifulSoup库搜索文档树

编程：（bs_3.4.4-text.py）

```
from bs4 import BeautifulSoup
import requests
url = 'https://gz.centanet.com/ershoufang/'
urlhtml=requests.get(url)
urlhtml.encoding='utf-8'
soup=BeautifulSoup(urlhtml.text, 'html.p
alink = soup.find_all(text='2室1厅')
print(alink)
```

- 这里查找是用完全匹配原则
- 如果这里用了find_all(text='2室')，得到的结果会是0个

```
['2室1厅', '2室1厅', '2室1厅', '2室1厅']
```

3.4.4 BeautifulSoup库搜索文档树

find_all(tag, attributes, recursive, text, limit, keywords)

- text 参数接受 字符串 , 正则表达式 , 列表, True

```
soup.find_all(text="Elsie")
# [u'Elsie']


soup.find_all(text=["Tillie", "Elsie", "Lacie"])
# [u'Elsie', u'Lacie', u'Tillie']


soup.find_all(text=re.compile("Dormouse"))
[u"The Dormouse's story", u"The Dormouse's story"]
```

3.4.4 BeautifulSoup库搜索文档树

4.limit 参数

- 限制find_all() 方法返回的搜索结果的数量.
- 当搜索到的结果数量达到 limit 的限制时,就停止搜索返回结果.

```
soup.find_all("a", limit=2)
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

文档树中有3个tag符合搜索条件,但结果只返回了2个,因为我们限制了返回数量

3.4.4 BeautifulSoup库搜索文档树

5. recursive 参数（用的很少）

- find_all() 方法中,该参数默认为True，检索当前tag的所有子孙节点；设置为recursive=False，搜索tag的直接子节点

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

```
soup.html.find_all("title")
# [<title>The Dormouse's story</title>]

soup.html.find_all("title", recursive=False)
# []
```

3.4.4 BeautifulSoup库搜索文档树

6.关键词参数 keyword

- 自选那些具有指定属性的标签

- 例，在中原房产网页中，查找户型为id='one2'的内容

```
from bs4 import BeautifulSoup
import requests
url = 'https://gz.centanet.com/ershoufang/'
urlhtml=requests.get(url)
urlhtml.encoding='utf-8'
soup=BeautifulSoup(urlhtml.text, 'html.parser')
alink = soup.find_all(id="one2")
print(alink)
```

```
>>> alink = soup.find_all(id="one2")
>>> print(alink)
[]
```

运行结果

```
[<li id="one2" onmousedown="setTab('one',2,10)"><span>预售证</span></li>]
```

- 如果一个指定名字的参数不是搜索内置的参数名,搜索时会把该参数当作指定名字tag的属性来搜索

```
soup = BeautifulSoup(html_doc, 'html.parser', from_encoding='utf-8')
```

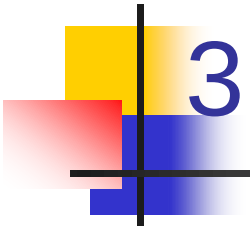
```
<p class="story">Once upon a time there were three little sisters; and their names were
    <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
    <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a>
    and
    <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
    and they lived at the bottom of a well.
</p>
```

使用多个指定名字的参数可以同时过滤tag的多个属性

```
1 soup.find_all(href=re.compile("elsie"), id='link1')
2 # [<a class="sister" href="http://example.com/elsie" id="link1">three</a>]
```

想用 class 过滤，不过 class 是 python 的关键词，这怎么办？加个下划线就可以。

```
soup.find_all("a", class_="sister")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```



3.4 HTML解析与Python实现

3.4.1 HTML与CSS要素

3.4.2 BeautifulSoup库及对象

3.4.3 BeautifulSoup库遍历文档树

3.4.4 BeautifulSoup库搜索文档树

3.4.5 BeautifulSoup库查找CSS过滤器

3.4.6 Python解析网页实例

3.4.5 BeautifulSoup库查找CSS过滤器

- BeautifulSoup对象select方法
- 功能：筛选出CSS标记的内容/查找html中需要的内容
- 获取内容方法：
  - ①通过标签名查找
  - ②通过类名查找
  - ③通过id名查找
  - ④组合查找：标签名与类名、id名等进行
    - 例.查找p标签中，id等于link1的内容，二者不要用空格分开
  - ⑤属性查找：查找时加入属性元素，属性用中括号括起来
    - 注意属性和标签属于同一节点，所以中间不能加空格，否则会无法匹配到。不在同一节点的使用空格隔开，同一节点的不加空格

3.4.5 BeautifulSoup库查找CSS过滤器

- soup.select()函数语法
  https://blog.csdn.net/wei_lin/article/details/102334956

```
select(self, selector, namespaces=None, limit=None, **kwargs)
```

- 参数：
  - selector，包含CSS选择器的字符串
  - 回顾CSS：
    - 标签名不加任何修饰
    - 类名前加点
    - id名前加 #

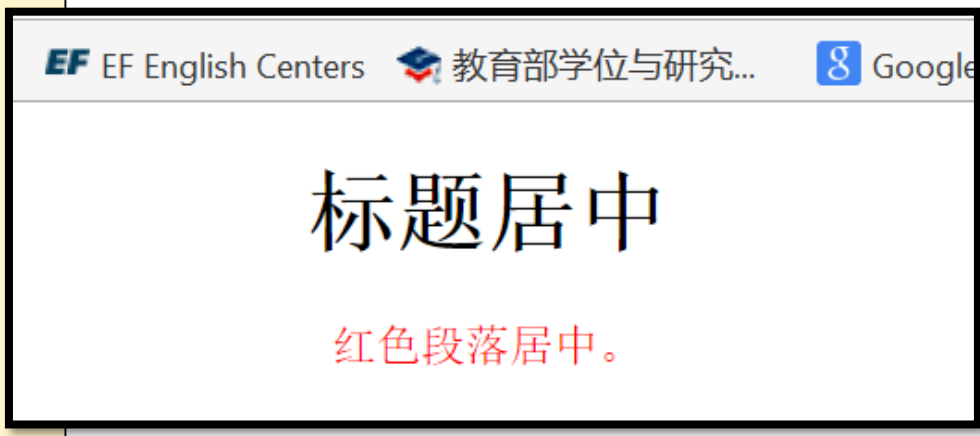
回顾CSS用法.HTML+CSS程序：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
.center
{
    text-align:center;
}
#para1
{
    text-align:center;
    color:red;
}
</style>
</head>
```

类名前加点

id名前加 #

```
<body>
<h1 class="center">标题居中</h1>
<p id="para1">红色段落居中。</p>
</body>
</html>
```



3.4.5 BeautifulSoup库查找CSS过滤器

- 以下面的HTML代码为例，筛选指定内容：

```
html = """
<html><head><title>BeautifulSoup select方法</title></head>
<body>
<p class="c" name="dromouse"><b>通过类筛选元素</b></p>
<p id="sid">根据ID筛选元素</p>
<a href="http://python.org"  id="link1">python编程</a>
<a href="http://python.com"  id="link2">python编程2</a>
<title>BeautifulSoup select方法</title>
<p><a href="http://www.baidu.com">baidu.com</a></p>
"""
```

- 分析代码

```
from bs4 import BeautifulSoup as bs
con=bs(html,'html.parser')
print(con.select('title'))   #①通过标签名查找
print(con.select('.c'))      #②通过类名查找
print(con.select('#sid'))    #③通过id名查找
print(con.select('a#link1')) #④组合查找
print(con.select('head > title'))   #head 下的title标签
print(con.select('a[href=http://python.com]'))  #加入属性href
print(con.select('p a[href=http://www.baidu.com]')) #④组合查找
```


- 运行结果

```
[<title>BeautifulSoup select方法</title>, <title>BeautifulSoup select方法</
title>]
[<p class="c" name="dromouse"><b>通过类筛选元素</b></p>]
[<p id="sid">根据ID筛选元素</p>]
[<a href="http://python.org" id="link1">python编程</a>]
[<title>BeautifulSoup select方法</title>]
[<a href="http://python.com" id="link2">python编程2</a>]
[<a href="http://www.baidu.com">baidu.com</a>]
```

- Select示例使用的程序头：
  - 处理的html
  - Soup赋值

```
from bs4 import BeautifulSoup

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 nam
es were
   <a class="sister" href="http://example.com/elsie" id="link1"> <!-- Elsie --> </a
>,
   <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>and
   <a class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>;
   and they lived at the bottom of a well.
  </p>
  <p class="story"> ... </p>
 </body>
</html>
"""
```

```
soup = BeautifulSoup(html, 'html.parser')
```

3.4.5 BeautifulSoup库查找CSS过滤器

1.通过（HTML）标签名查找

```
1 print(soup.select('title'))
2 print(soup.select('a'))#输出的列表含有多个包含标签<a>的元素
```

输出

```
[<title>The Dormouse's story</title>]
[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

```
[<title> The Dormouse's story </title>]
[<a class="sister" href="http://example.com/elsie" id="link1"> <!-- Elsie --> </
a>, <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>, <a
class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>]
```

3.4.5 BeautifulSoup库查找CSS过滤器

2.通过CCS类选择器查找

```
1 print(soup.select('.story'))
```

输出

```
[<p class="story">Once upon a time there were three little sisters; and their names
 were
<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>,
<p class="story">...</p>]
```

3.通过CSS id 选择器查找

```
1 print(soup.select('#link1'))
```

输出

```
[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]
```

4.组合查找

组合查找和通过标签名与类选择器、id选择器进行查找的组合原理是一样的，例如查找 p 标签中，id 等于 link1的内容，二者需要用空格分开。

```
1 print(soup.select('p #link1'))
```

输出

```
[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]
```

3.4.5 BeautifulSoup库查找CSS过滤器

5.子标签查找

父标签与子标签之间通过" > "表示递进关系

```
1 print(soup.select('p > b'))
```

输出

输出为:

```
1 [<b>The Dormouse's story</b>]
```

通过子标签查找时的注意事项：soup.select()尽量不使用完整selector

3.4.5 BeautifulSoup库查找CSS过滤器

6.通过属性查找

还可以加入属性元素，属性需要用中括号括起来，注意属性和标签属于同一节点，所以中间不能加空格，否则会无法匹配到。

```
# 通过href属性及其值进行查找
print(soup.select('a[href="http://example.com/elsie"]'))
```

输出

```
[<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>]
```