首先，bs是配合解析器来完成对html或者xml格式文本内容的解析与数据提取的。

目前用lxml作为解析器即可

如何使用

- Beautiful Soup 将复杂的 HTML 文档转换成一个复杂的由 Python 对象构成的树形结构，但处理对象 的过程只包含 4 种类型的对象: Tag, NavigableString, BeautifulSoup, 和 Comment。
- 然后呢，我们知道使用bs得把所需要的html文本路径，解析器这俩东西放到参数表里
  - soup = bs(1,2)
  - 像上边这样
- 然后呢，正如上边所说我们要处理的就是bs传给soup的对象
  - Tag就是html标签，有开闭的整体
  - NavigableString，字符串对应 tag 中的一段文本。Beautiful Soup 用 `NavigableString` 类来包装 tag 中的字符串
  - `BeautifulSoup` 对象表示的是一个文档的全部内容。大部分时候，可以把它当作 `Tag` 对象， 它支持 遍历文档树 和 搜索文档树 中描述的大部分的方法。
    - 因为 `BeautifulSoup` 对象并不是真正的HTML或XML的tag,所以它没有name和attribute属性。 但有时查看它的 `.name` 属性是很方便的，所以 `BeautifulSoup` 对象包含了一个 值为 “document” 的特殊属性 `.name`
  - comment

选择元素的方式

节点选择

这是用tag来选择的，具体格式是 `soup.tag` ， 下面是一些相关知识

- 可以用tag的名字选择，比如 `<h1>` 的h1，那么就是 `soup.h1` 可以返回它检索到的第一个h1标签的信息，可以做赋值操作改变这个tag的名字
- 进一步还可以加上 `.` 来选择这个标签的属性attrs，属性有id，class，等等，包含在开标签内
- 比如 `soup.a.href.string` ,a标签里的href属性的字符串内容或者可以写成 `soup.a['href'].string`
- 如果只找属性的话可以 `soup.[attrs]` 或者直接 `soup.attrs` ，就可以返回第一个检索到的这个属性的对应的值
- 多值属性：顾名思义就是包含多个值的属性，比如 `class="nm mn"` 就表示class里包含两个值

- 对应的，你选择这个属性时返回包含所有值的列表，比如上面的就是写成 `soup.class` 然后返回 `["mn", "mn"]` 这个列表

关联选择

可以选择一个标签的父标签，子标签，兄弟标签

CSS选择器

使用对bs解析后的对象用select（）方法就ok了

关于CSS选择器语法

- id选择器：用 `#id的值` 来定位
- 类/class选择器：用 `.类的值` 来定位
- 标签选择器：直接用tag名字选择
- 混合使用：比如某个特定子tag ： li，就可以先用id/class选择器来选择其父tag，然后再用li就可以了，
  - 注意要加引号，然后每一个元素（id，class，tag）都要用空格隔开，并且父到子关系是从左到右的
- 由于select选择是返回所有满足语法的tag，所以可以根据索引来选择需要的tag
  - 比如 `soup.select("a")[1]` 表示的就是选择的所有a标签中第二个a标签

CSS选择器高级用法

嵌套选择

- 就是选择一个标签后继续选择里边的下一个标签
- 比如,举个例子网址，例子

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <title>白月黑羽测试网页1</title>

        <style>
            .animal {color: red;}
        </style>
    </head>

    <body>
```

```
        <div class="plant"><span>土豆</span></div>
        <div class="plant"><span>洋葱</span></div>
        <div class="plant"><span>白菜</span></div>

        <div class="animal"><span>狮子</span></div>
        <div class="animal"><span>老虎</span></div>
        <div class="animal"><span>山羊</span></div>

        <input  type="text" id='searchtext' />

        <div id='container'>

            <div id='layer1'>
                <div id='inner11'>
                    <span>内层11</span>
                </div>
                <div id='inner12'>
                    <span>内层12</span>
                </div>
            </div>

            <div id='layer2'>
                <div id='inner21'>
                    <span>内层21</span>
                </div>
            </div>

        </div>


        <div id='bottom'>
            <div class='footer1'>
                <span class='copyright'>版权</span>
                <span class='date'>发布日期：2018-03-03</span>
            </div>
            <div class='footer2'>
                <span>备案号
                    <a href="http://www.miitbeian.gov.cn">苏ICP备88885574号
</a>
                </span>
            </div>
        </div>

    </body>
</html>
```

- 我去有点长，唉反正也差不多
- 比如

```
soup = bs(上边的html,"lxml")
head = soup.select('head')
print(head)
```

- 就会返回
- 然后我们再找它的子标签比如title

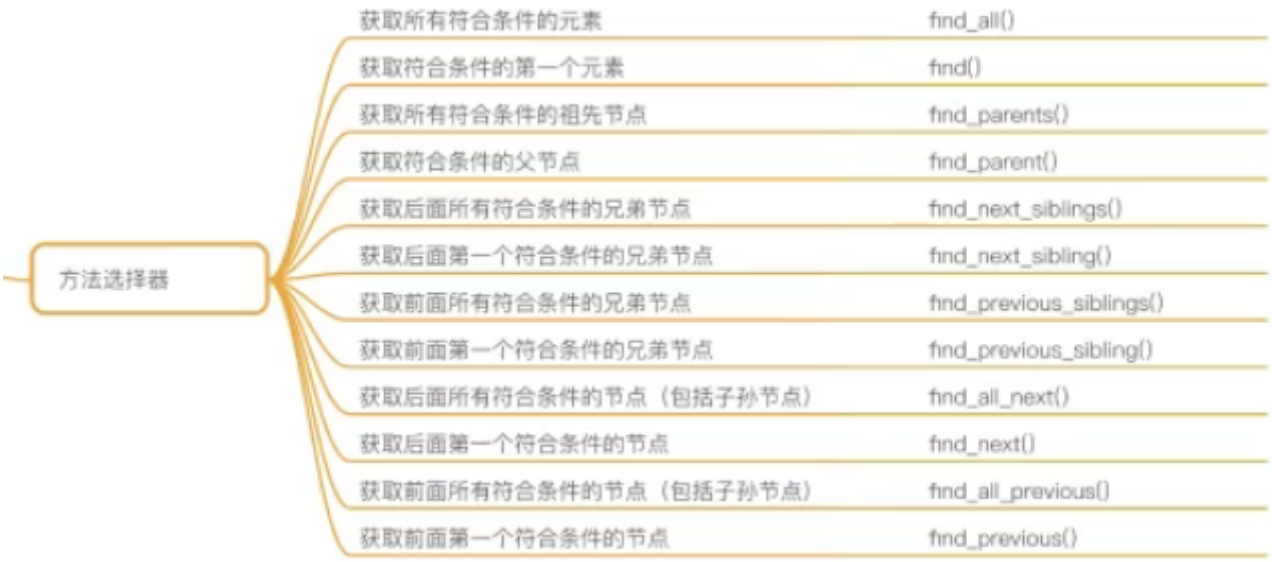
```
title1 = head.select('title')
```

- 这样的就可以了，但加入里边有很多title标签呢，那么此时head就是一个列表，你可以用for循环来打印

获取属性
和上面的节点选择一样，就不多说了

获取文本
和上边节点部分一样，都用string和strings属性获取

方法选择器



先讲find_all和find方法

find_all

- 用于搜索当前节点下的所有符合条件的节点（节点就是tag）
- 但是它会有选择的范围，如果你直接把刚解析好得到的soup拿来用findall那就是全文搜索
    - 但是如果单独提取出来的话比如提取了所有的a标签，赋值给了a1变量，对a1变量用findall那就是对于所有a标签找到符合特定条件的a标签
- 有5个参数，name, attrs, recursive, text, `**kwargs`
- name：查找所有名字为这个指定的name的节点（tag）
    - 形式：
        - 字符串：那就是直接是标签名，比如a就是a标签
            - 例子`soup.find_all(name="a")`就是返回所有a标签
        - 正则表达式：通过match函数匹配内容（可以用re库）
        - 列表：这个列表里包含多个标签，那就会按要求以列表形式返回这些标签，那么别忘了for循环使用
            - 例：`soup.find_all(name=['a','b'])`返回包含a，b标签的列表
        - True：匹配任意标签
- attrs：查询有相应属性及值的标签
    - 形式：字典
    - 比如`soup.find_all({id:"nima"})`就是返回id值为nima的所有标签
        - 注意冒号别写成=
- kwargs：接受常用的属性参数（比如id，class）
    - 形式：参数赋值
    - 例：`soup.find_all(id="114514")`和`soup.find_all(class_="woqu")`（class是python关键字所以加一个下划线）
- text：查询含有接受的文本的标签
    - 形式：字符串（比如text=”jjda“）
    - 1.直接搜索文档中含有这个text的标签然后返回
    - 2.与其他参数混合使用
        - 比如`soup.find_all(name='title', text='卧槽')`
- 不是5个，还有其他的参数
- limit：返回的数量，顾名思义
- recursive：是否获取子孙节点（tag），默认为True

find

- 和find_all不同点是只返回检索到的第一个节点，且不包含子孙节点（也就是说recursive默认为False）