**爬虫初体验**

上一关，我们学习了HTML基础知识，完成了读懂和修改HTML文档的目标。

最后，还写了这样一段代码：即通过调用requests库，获取到了网页源代码：

import requests #调用requests库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

#获取网页源代码，得到的res是Response对象

html=res.text

#把res的内容以字符串的形式返回

print('响应状态码:',res.status\_code) #检查请求是否正确响应

print(html) #打印网页源代码

而在今天这一关，如果用一个词来描述我们要学习的内容，那就是——**BeautifulSoup**模块。

BeautifulSoup到底在爬虫中发挥着怎样的作用呢？一起来看看。

BeautifulSoup是什么

我们得先来回忆一下爬虫的四个步骤：



第0关的requests库帮我们搞定了爬虫第0步——获取数据；第1关的HTML知识，是进行爬虫必不可少的背景知识，能辅助我们解析和提取数据。

而本关学习目标：学会使用BeautifulSoup解析和提取网页中的数据。



【解析数据】是什么意思呢？

我们平时使用浏览器上网，浏览器会把服务器返回来的HTML源代码翻译为我们能看懂的样子，之后我们才能在网页上做各种操作。

而在爬虫中，也要使用能读懂html的工具，才能提取到想要的数据。

这就是解析数据。

【提取数据】是指把我们需要的数据从众多数据中挑选出来。

老师还想提醒一下：解析与提取数据在爬虫中，既是一个重点，也是难点。因为这一关要讲两步，信息量会比之前两关大，所以希望你在学习的时候，能做好一定的心理准备，投入更多精力。

不过，我对你有信心。O(∩\_∩)O

那么接下来，我们就来正式学习BeautifulSoup是如何解析和提取数据的吧~

BeautifulSoup怎么用

由于BeautifulSoup不是Python标准库，需要单独安装它，我们的学习系统已经安装好了。如果你是在自己的电脑上运行，需要在终端输入一行代码运行：pip install BeautifulSoup4。（Mac电脑需要输入pip3 install BeautifulSoup4）

解析数据

BeautifulSoup解析数据的用法很简单，请看下图：



在括号中，要输入两个参数，第0个参数是要被解析的文本，注意了，它必须必须必须是字符串。

括号中的第1个参数用来标识解析器，我们要用的是一个Python内置库：html.parser。（它不是唯一的解析器，但是比较简单的）

我们看看具体的用法。仍然以网站这个书苑不太冷为例（url：[https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html](https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html" \t "_blank)），假设我们想爬取网页中的书籍类型、书名、链接、和书籍介绍。

根据之前所学的requests.get()，我们可以先获取到一个Response对象，并确认自己获取成功：

import requests #调用requests库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

#获取网页源代码，得到的res是response对象

print(res.status\_code) #检查请求是否正确响应

html = res.text #把res的内容以字符串的形式返回

print(html)#打印html

上面的代码是第0关学过的内容，好，接下来就轮到BeautifulSoup登场解析数据了，请特别留意第2行和第6行新增的代码。

import requests

from bs4 import BeautifulSoup

#引入BS库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

html = res.text

soup = BeautifulSoup(html,'html.parser') #把网页解析为BeautifulSoup对象

第2行是引入BeautifulSoup库。

第6行中的第0个参数，必须是字符串类型；括号中的第1个参数是解析器。

这就是解析数据的用法。

接下来，我们来打印看看soup的数据类型，和soup本身（第5行开始为新增代码）。

import requests

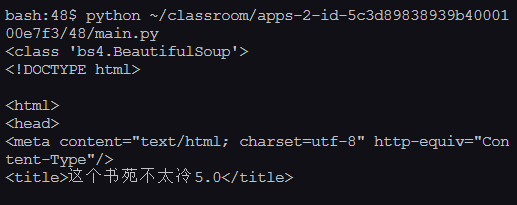
from bs4 import BeautifulSoup

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

soup = BeautifulSoup( res.text,'html.parser')

print(type(soup)) #查看soup的类型

print(soup) # 打印soup



看看运行结果，soup的数据类型是<class 'bs4.BeautifulSoup'>，说明soup是一个BeautifulSoup对象。

下一行开始，就是我们打印的soup，它是我们所请求网页的完整HTML源代码。我们所要提取的书名、链接、书籍内容这些数据都在这里面。

可是疑点来了：如果有非常细心的同学，也许会发现，打印soup出来的源代码和我们之前使用response.text打印出来的源代码是完全一样的。

也就是说，我们好不容易用BeautifulSoup写了一些代码来解析数据，但解析出的结果，竟然和没解析之前一样。

你听我解释，事情是这样的：虽然response.text和soup打印出的内容表面上看长得一模一样，却有着不同的内心，它们属于不同的类：<class 'str'> 与<class 'bs4.BeautifulSoup'>。前者是字符串，后者是已经被解析过的BeautifulSoup对象。之所以打印出来的是一样的文本，是因为BeautifulSoup对象在直接打印它的时候会调用该对象内的str方法，所以直接打印 bs 对象显示字符串是str的返回结果。

我们之后还会用BeautifulSoup库来提取数据，如果这不是一个BeautifulSoup对象，我们是没法调用相关的属性和方法的，所以，我们刚才写的代码是非常有用的，并不是重复劳动。

到这里，你就学会了使用BeautifulSoup去解析数据：

from bs4 import BeautifulSoup

soup = BeautifulSoup(字符串,'html.parser')

完成了爬虫的第1步：解析数据，下面就是爬虫的第2步：提取数据。

提取数据

我们仍然使用BeautifulSoup来提取数据。

这一步，又可以分为两部分知识：find()与find\_all()，以及Tag对象。



先看find()与find\_all()。

find()与find\_all()是BeautifulSoup对象的两个方法，它们可以匹配html的标签和属性，把BeautifulSoup对象里符合要求的数据都提取出来。

它俩的用法基本是一样的，区别在于，find()只提取首个满足要求的数据，而find\_all()提取出的是所有满足要求的数据。



看两个例子你就清楚了。以这个网页为例（URL: <https://localprod.pandateacher.com/python-manuscript/crawler-html/spder-men0.0.html>）：



在网页的HTML代码中，有三个<div>元素，用find()可以提取出首个元素，而find\_all()可以提取出全部。

看代码（第7行为新增代码），然后点击运行，查看结果：

import requests

from bs4 import BeautifulSoup

url = 'https://localprod.pandateacher.com/python-manuscript/crawler-html/spder-men0.0.html'

res = requests.get (url)

print(res.status\_code)

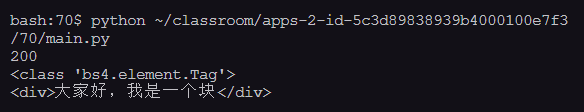
soup = BeautifulSoup(res.text,'html.parser')

item = soup.find('div') #使用find()方法提取首个<div>元素，并放到变量item里。

print(type(item)) #打印item的数据类型

print(item) #打印item

看，运行结果正是首个<div>元素吧！我们还打印了它的数据类型：<class 'bs4.element.Tag'>，说明这是一个Tag类对象。



再来试试find\_all()吧，它可以提取出网页中的全部三个<div>元素，请看代码（第7行为新增代码），然后点击运行。

import requests

from bs4 import BeautifulSoup

url = 'https://localprod.pandateacher.com/python-manuscript/crawler-html/spder-men0.0.html'

res = requests.get (url)

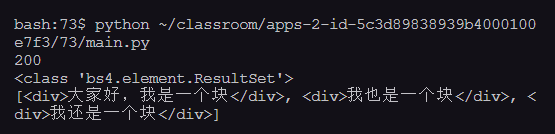
print(res.status\_code)

soup = BeautifulSoup(res.text,'html.parser')

items = soup.find\_all('div') #用find\_all()把所有符合要求的数据提取出来，并放在变量items里

print(type(items)) #打印items的数据类型

print(items) #打印items



运行结果是那三个<div>元素，它们一起组成了一个列表结构。打印items的类型，显示的是<class 'bs4.element.ResultSet'>，是一个ResultSet类的对象。其实是Tag对象以列表结构储存了起来，可以把它当做列表来处理。

下面，我想强调一下它们用法中的两个要点：



首先，请看举例中括号里的class\_，这里有一个下划线，是为了和python语法中的类 class区分，避免程序冲突。当然，除了用class属性去匹配，还可以使用其它属性，比如style属性等。

其次，括号中的参数：标签和属性可以任选其一，也可以两个一起使用，这取决于我们要在网页中提取的内容。

如果只用其中一个参数就可以准确定位的话，就只用一个参数检索。如果需要标签和属性同时满足的情况下才能准确定位到我们想找的内容，那就两个参数一起使用。

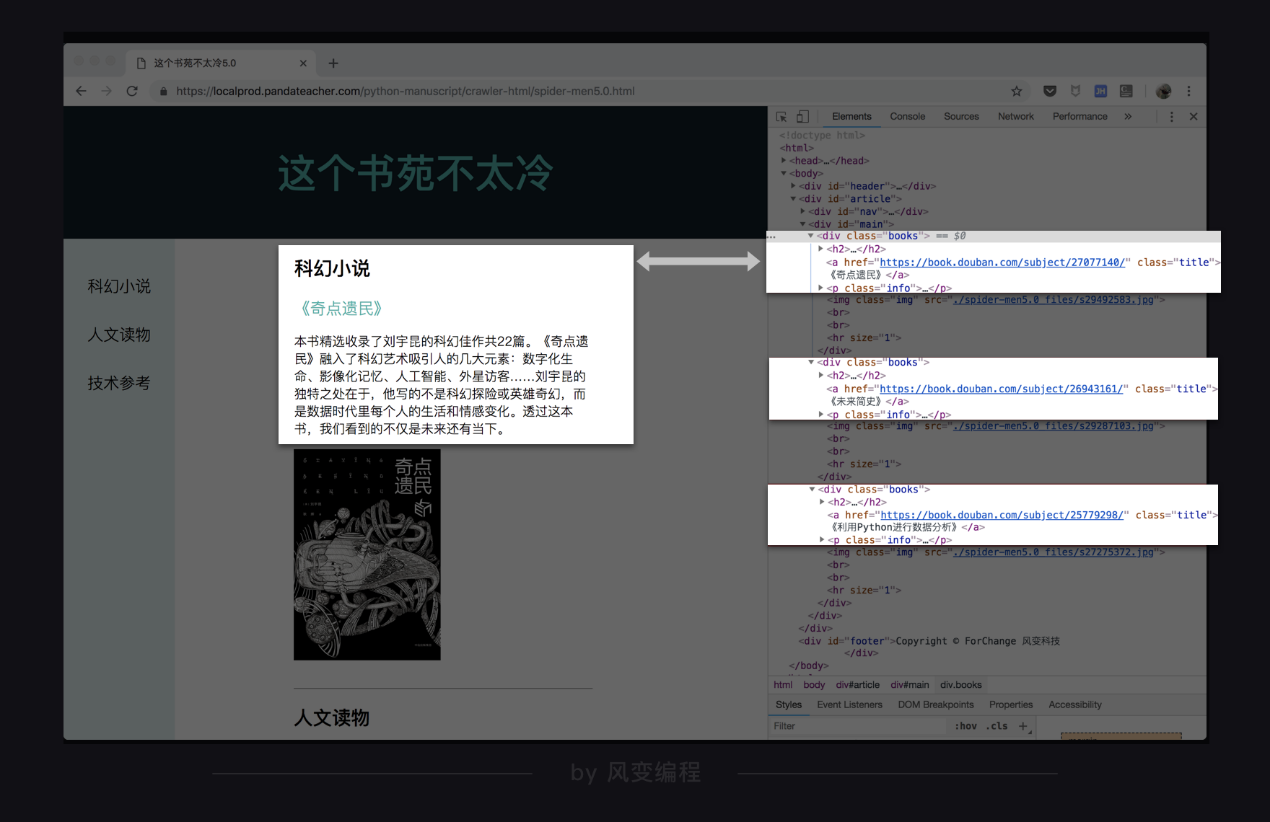
再次总结一下find()与find\_all()的用法：



这么多的内容，不太可能一下就记住，要想熟练使用，还需要大量练习。那么现在我们就来做个小练习吧，仍然以网站这个书苑不太冷为例：

（url：[https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html](https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html" \t "_blank)），目标是爬取网页中的三本书的书名、链接、和书籍介绍。

打开网址，在网页上点击右键-检查，查看源代码，先看一看目标数据所对应的位置。



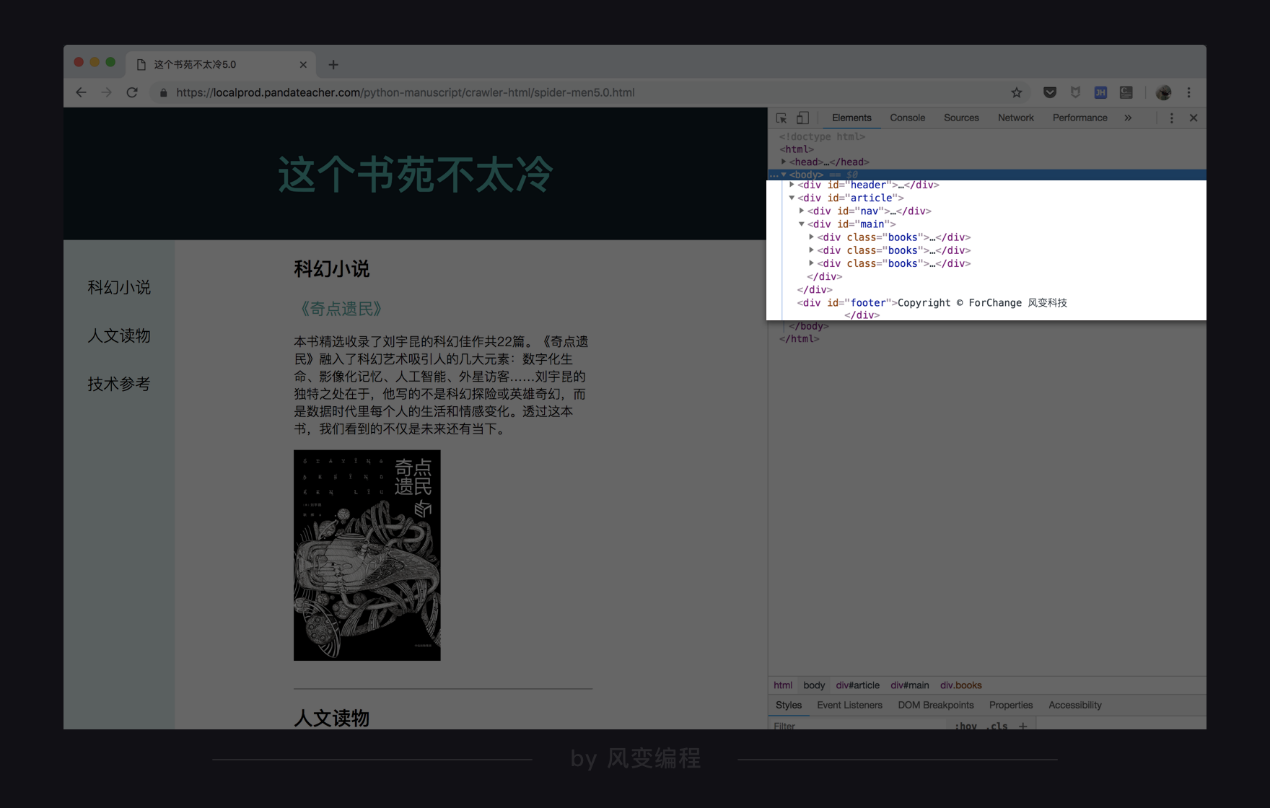
滑动一下网页，看见我们想要的每一本书的数据，分别存在了三个<div>元素中，并且有相同的属性：class="books"，这个共同点就是我们去提取数据的关键。

我们可以先把这三个<div>元素提取出来，然后再进一步提取那些具体的书的类型、书名等等。

由于我们要找的不是一本书的数据，而是所有书的数据都要找，所以这时应该用find\_all()。

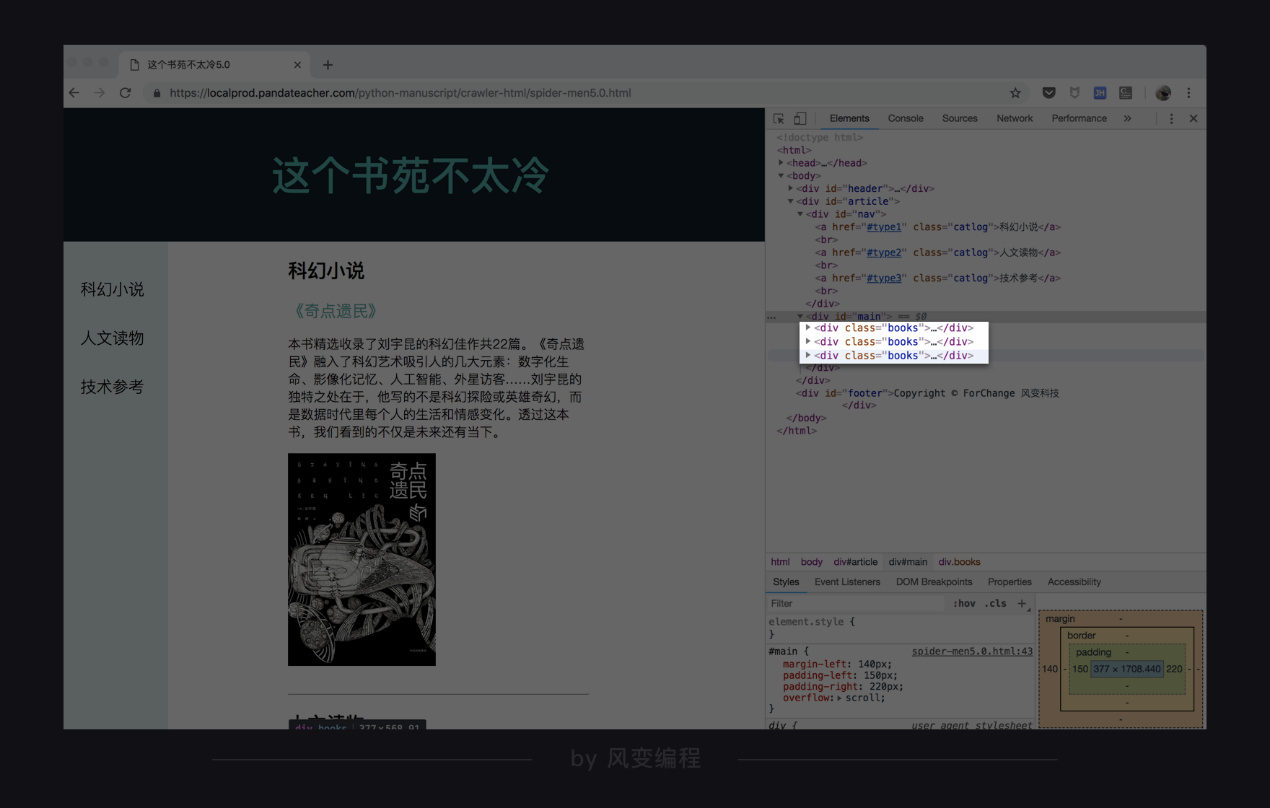
接下来要考虑的就是，要用什么参数去查找和定位，标签，还是属性。此时，可以用到开发者工具的搜索功能，点击Ctrl+F，Mac电脑用command+F。

在搜索栏中输入div试试，搜索结果是：



一共找到了8个<div>元素，但我们只想找到3个，如果只用<div>来检索，就会把其它不需要的信息也提取出来。

再看看属性class="books"，搜索一下会发现，整个HTML源代码中，是只有我们要找的三个元素的属性满足，因此，我们这次就可以只使用这个属性来提取。



（注：点击右键-显示网页源代码，在这个页面里去搜索会更加准确，在这里我们是点击右键-检查，在这个页面里去搜索的）

现在，来看看代码怎么写，再点击运行：

import requests # 调用requests库

from bs4 import BeautifulSoup # 调用BeautifulSoup库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')# 返回一个Response对象，赋值给res

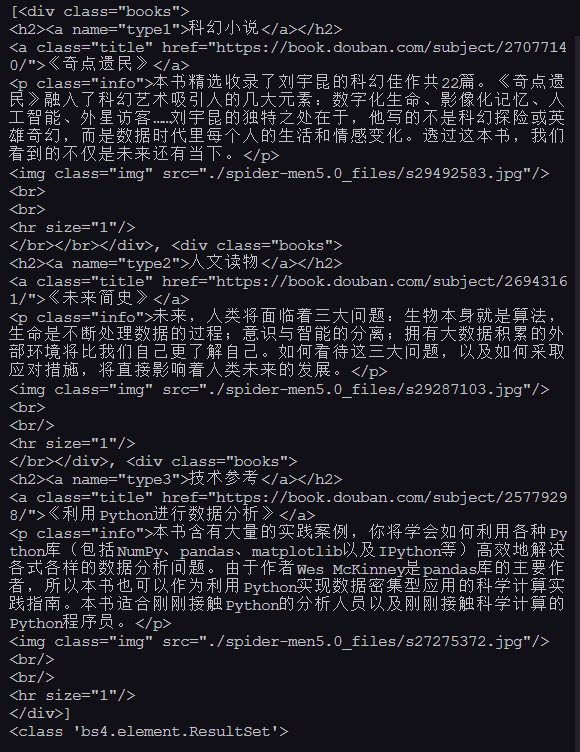
html = res.text# 把Response对象的内容以字符串的形式返回

soup = BeautifulSoup( html,'html.parser') # 把网页解析为BeautifulSoup对象

items = soup.find\_all(class\_='books') # 通过匹配标签和属性提取我们想要的数据

print(items) # 打印items

print(type(items)) #打印items的数据类型



现在，三本书的全部信息都被我们提取出来了。它的数据类型是<class 'bs4.element.ResultSet>， 前面说过可以把它当做列表list来看待。

不过，列表并不是我们最终想要的东西，我们想要的是列表中的值，所以要想办法提取出列表中的每一个值。

用for循环遍历列表，就可以把这三个<div>元素取出来了。

请仔细阅读代码，然后点运行看看：

import requests # 调用requests库

from bs4 import BeautifulSoup # 调用BeautifulSoup库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')# 返回一个Response对象，赋值给res

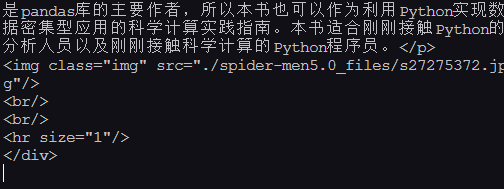
html= res.text# 把Response对象的内容以字符串的形式返回

soup = BeautifulSoup( html,'html.parser') # 把网页解析为BeautifulSoup对象

items = soup.find\_all(class\_='books') # 通过定位标签和属性提取我们想要的数据

for item in items:

print('想找的数据都包含在这里了：\n',item) # 打印item



程序运行很顺利，结果正是那三个<div>元素。

其实到这里，find()和find\_all()的用法讲了，练习也做了，但是，我们现在打印出来的东西还不是目标数据，里面含着HTML标签，所以下面，我们要进入到提取数据中的另一个知识点——Tag对象。



咱们还以上面的代码为例，我们现在拿到的是一个个包含html标签的数据，还没达成目标。

这个时候，我们一般会选择用type()函数查看一下数据类型，因为Python是一门面向对象编程的语言，只有知道是什么对象，才能调用相关的对象属性和方法。

好，来打印一下：

import requests # 调用requests库

from bs4 import BeautifulSoup # 调用BeautifulSoup库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html') # 返回一个response对象，赋值给res

html = res.text# 把res的内容以字符串的形式返回

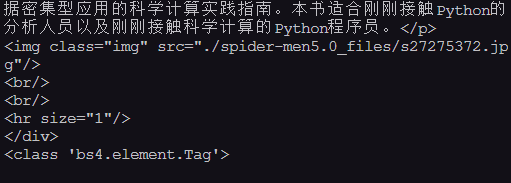
soup = BeautifulSoup( html,'html.parser') # 把网页解析为BeautifulSoup对象

items = soup.find\_all(class\_='books') # 通过定位标签和属性提取我们想要的数据

for item in items:

print('想找的数据都包含在这里了：\n',item) # 打印item

print(type(item))



我们看到它们的数据类型是<class 'bs4.element.Tag'>，是Tag对象，不知道你是否还记得，这与find()提取出的数据类型是一样的。

好，既然知道了是Tag对象，下一步，就是看看Tag类对象的常用属性和方法了。



上图是Tag对象的3种用法，咱们一个一个来讲。

首先，Tag对象可以使用find()与find\_all()来继续检索。

回到我们刚刚写的代码：即爬取这个书苑不太冷网站中每本书的类型、链接、标题和简介，我们刚刚拿到的分别是三本书的内容，即三个Tag对象。现在，先把首个Tag对象展示在下面，方便我们阅读：

<div class="books">

<h2><a name="type1">科幻小说</a></h2>

<a href="https://book.douban.com/subject/27077140/" class="title">《奇点遗民》</a>

<p class="info">本书精选收录了刘宇昆的科幻佳作共22篇。《奇点遗民》融入了科幻艺术吸引人的几大元素：数字化生命、影像化记忆、人工智能、外星访客……刘宇昆的独特之处在于，他写的不是科幻探险或英雄奇幻，而是数据时代里每个人的生活和情感变化。透过这本书，我们看到的不仅是未来还有当下。

</p>

<img class="img" src="./spider-men5.0\_files/s29492583.jpg">

<br>

<br>

<hr size="1">

</div>

1. BeautifulSoup是什么

2. BeautifulSoup怎么用

解析数据

提取数据

3. 对象的变化过程

4. 本关总结

看第2行：书籍的类型在这里面；第3行：我们要取的链接和书名在里面；第4行：书籍的简介在里面。因为是只取首个数据，这次用find()就好。

先阅读下面的代码（从11行开始为新增代码）：

import requests # 调用requests库

from bs4 import BeautifulSoup # 调用BeautifulSoup库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

# 返回一个response对象，赋值给res

html = res.text

# 把res的内容以字符串的形式返回

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

# 把网页解析为BeautifulSoup对象

items = soup.find\_all(class\_='books') # 通过定位标签和属性提取我们想要的数据

for item in items:

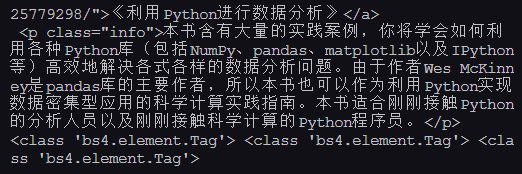
kind = item.find('h2') # 在列表中的每个元素里，匹配标签<h2>提取出数据

title = item.find(class\_='title') #在列表中的每个元素里，匹配属性class\_='title'提取出数据

brief = item.find(class\_='info') #在列表中的每个元素里，匹配属性class\_='info'提取出数据

print(kind,'\n',title,'\n',brief) # 打印提取出的数据

print(type(kind),type(title),type(brief)) # 打印提取出的数据类型



除了我们拿到的数据之外；运行结果的数据类型，又是三个<class 'bs4.element.Tag'>，用find()提取出来的数据类型和刚才一样，还是Tag对象。接下来要做的，就是把Tag对象中的文本内容提出来。

这时，可以用到Tag对象的另外两种属性——Tag.text，和Tag['属性名']。



我们用Tag.text提出Tag对象中的文字，用Tag['href']提取出URL。

只需要修改最后一行代码，我们想要的数据就都能成功提取出来了：

import requests # 调用requests库

from bs4 import BeautifulSoup # 调用BeautifulSoup库

res =requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.html')

# 返回一个response对象，赋值给res

html=res.text

# 把res解析为字符串

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

# 把网页解析为BeautifulSoup对象

items = soup.find\_all(class\_='books') # 通过匹配属性class='books'提取出我们想要的元素

for item in items: # 遍历列表items

kind = item.find('h2') # 在列表中的每个元素里，匹配标签<h2>提取出数据

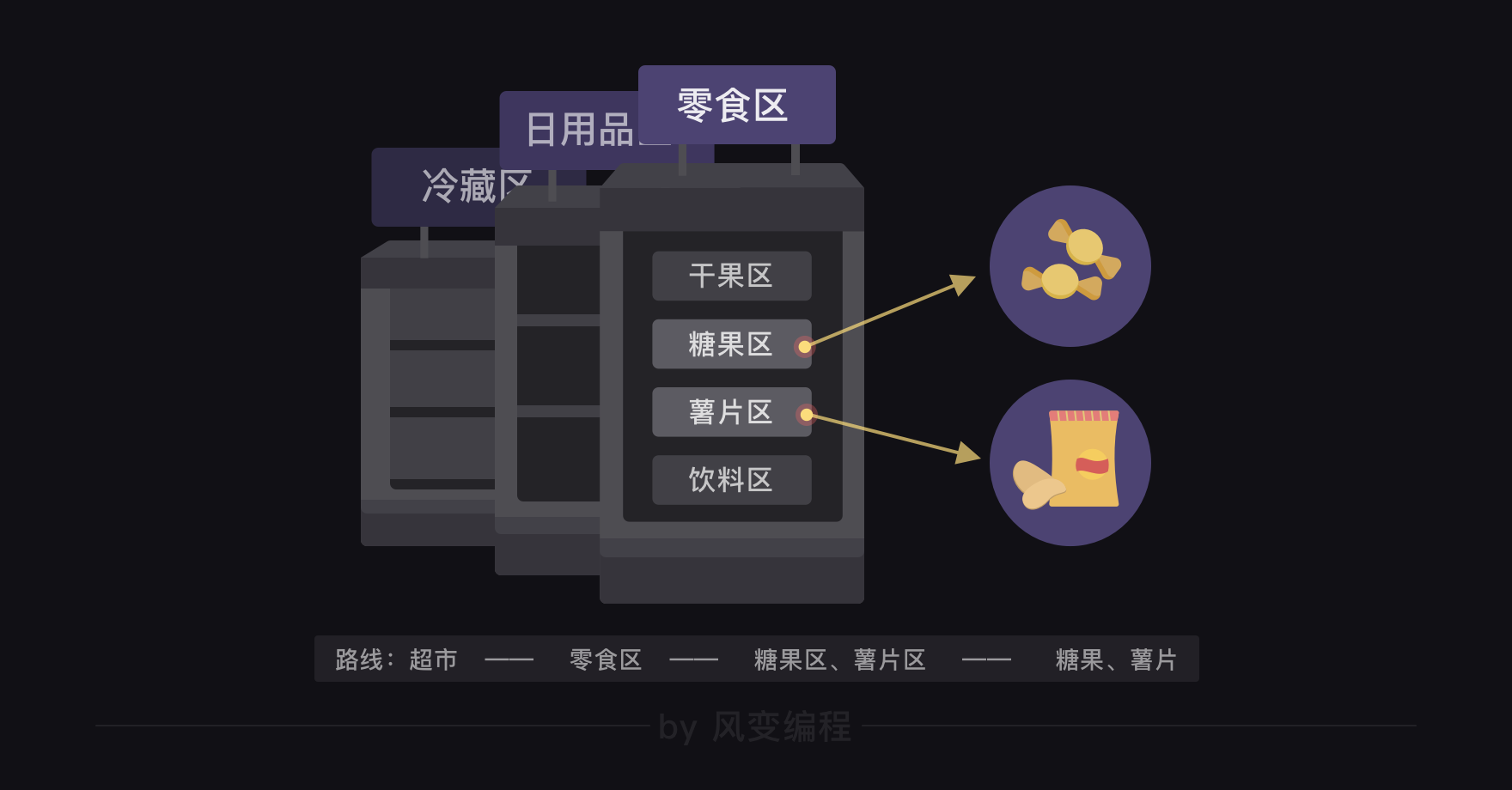
title = item.find(class\_='title') # 在列表中的每个元素里，匹配属性class\_='title'提取出数据

brief = item.find(class\_='info') # 在列表中的每个元素里，匹配属性class\_='info'提取出数据

print(kind.text,'\n',title.text,'\n',title['href'],'\n',brief.text) # 打印书籍的类型、名字、链接和简介的文字

看看终端，拿出来啦~(≧▽≦)/~此处应该有掌声，到这里，我们终于成功解析、提取到了所有的数据。

这个层层检索的过程有点像是在超市买你想要的零食，比如一包糖果和一包薯片，首先要定位到超市的零食区，然后去糖果区找糖果，再去薯片区找薯片。



不过呢，每个网页都有自己的结构，我们写爬虫程序，还是得坚持从实际出发，具体问题具体分析哈。

我为你准备了一些习题，记得要去完成它们，你与爬虫大神的距离，还要靠一个一个练习去缩短。

走到这里，你已经学完了如何用BeautifulSoup库的相关知识来解析和提取数据。面对这扑面而来的新的知识，我们有必要来梳理一下：

对象的变化过程

其实说白了，从最开始用requests库获取数据，到用BeautifulSoup库来解析数据，再继续用BeautifulSoup库提取数据，不断经历的是我们操作对象的类型转换。

请看下图：



上一关，我们的操作对象从URL链接到了Response对象。而这一关，我们的操作对象是这样的：Response对象——字符串——BS对象。到这里，又产生了两条分岔：一条是BS对象——Tag对象；另一条是BS对象——列表——Tag对象。

而操作对象的转变，则是借由一些步骤完成的，在图中是由阿拉伯数字标注的内容：从Response对象到字符串，是通过response.text完成的，我就不赘述了，图上标示得很清楚。

在此刻，我尤其想要强调的是，学到现在的你炒鸡棒的，b(￣▽￣)d，而你记不全这些内容太太太正常了，因为编程从来都是一门强调实操实练的学科。

好，现在想请屏幕前的你深吸一口气，在椅子上一定坐稳了，千万不要晕倒，因为老师对你隐瞒了一件事。

其实刚刚那个图还不完整，完整版的图示是这样的：



Python是一门面向对象编程的过程，图中用英文字母的序号来展示的是每一种对象的方法和属性。比如bs对象的方法有find()和find\_all()。我也不赘述了。

这个流程其实对应的是爬虫四步的前3步：

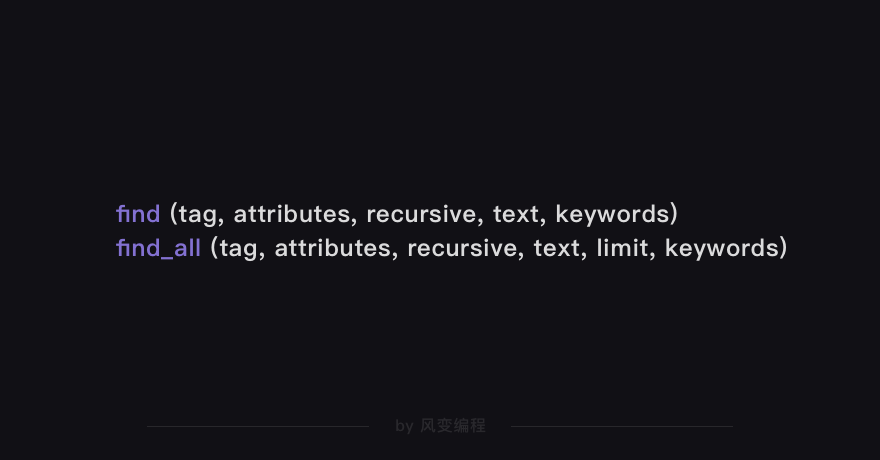


这张图还挺重要的，可以作为你做练习、复习的一个关键参考信息。❀

不过梳理完之后，我们还是得来敲敲代码，光看不练可不行。所以，学完知识之后不只要去写作业，最好把本关的的代码全部再写一次，因为学习可是需要我们付出努力的噢。

最后，我还想多说几句。在BeautifulSoup中，不止find()和find\_all()，还有select()也可以达到相同目的。

其实，在bs的官方文档中，find()与find\_all()的方法，其实不止标签和属性两种，还有这些：



不过，我们所教的这两种方法，足以胜任你的95%以上的html解析提取问题，以及风变编程中所有html解析与提取的问题。

为什么我不把所有知识都讲给你，因为，学BeautifulSoup就可能需要好多个关卡了，我们的目的只是带领你走一条快速入门的路。

学无止境，当这些知识都被你掌握，而且练习了更多的项目之后，就有余力去自学和拓展更多的知识了。

即便我只是挑了重点之中的重点为你讲解解析数据的方法，这一关的知识量还是有些大，在学完这个关卡之后，你必须再回过头复习一遍，这对你去理解和记忆这些知识，是相当有帮助的。

通过这一关的学习，你就搞明白了用bs库解析数据和提取数据的方法，只要数据在html里，你都可以拿到了。

下一关，我们会进行BeautifulSoup库的相关实操，务求咱们把所学的都用好~

期待再次与你相见~