第四关 迫在眉睫：解析网页

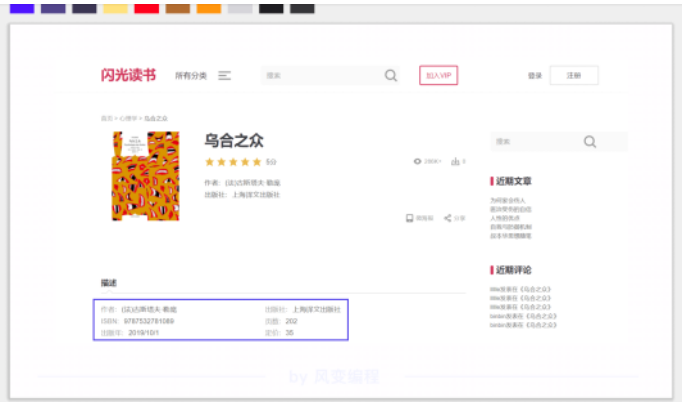
2022年12月2日 9:39

1. 项目代码

根据任务要求，我们需要获得所有书籍的 ISBN、定价等信息。
为了用爬虫完成这个任务，我们先用两节课学习了简单的网络请求知识和 HTML 语言知识。接下来的两节课，我们会边学知识边做项目。
本节课的任务是爬取单本书的信息，这可以当作一个小的爬虫项目。
在实现该项目之前，我们需要明确爬取的具体内容，再来看如何用代码实现。

1.1 明确需求

首先，爬取单本书籍的信息，需要进入书籍的简介页面。
以《乌合之众》为例，《乌合之众》书籍简介页面的 URL 是： https://wp.forchange.cn/psychology/11069/ 。
网页的显示内容如下：



框出的内容（作者、ISBN、出版年、出版社、页数、定价）为我们想要获取的内容，这些信息我们可以都爬下来备用。
今天我们可以把爬取到的数据暂时存到变量中，以备后续使用。下节课我会教你怎么把批量爬取到的数据存到文件中。
只要把目标分析清楚，实现爬虫项目的过程并不难，因为爬虫的流程很固定。

1.2 体验代码

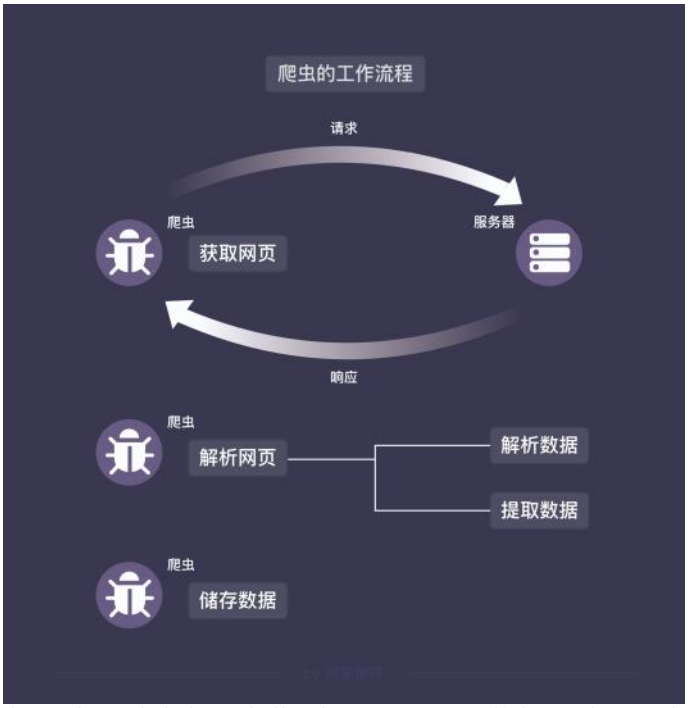
在开始学习前，先给你展示一下今天项目的最终成果。
代码体验

```python
import requests
from bs4 import BeautifulSoup

# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'

# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')

# 搜索书籍信息的父节点<div>
info_tag = bs.find('div', class_='res-attrs')
# 搜索每条信息的节点<dl>
info_list = info_tag.find_all('dl')

# 创建字典，存储书籍信息
info_dict = {}
```

```
25
26      # 遍历搜索结果，提取文本内容，存储到字典中
27      for info in info_list:
28          # 提取信息提示项<dt>的元素内容
29          key = info.find('dt').text[:-2]
30          # 提取书籍信息<dd>的元素内容
31          value = info.find('dd').text
32          # 将信息添加到字典中
33          info_dict[key] = value
34
35      # 打印查看字典中的书籍信息
36      print(info_dict)
```

```
importrequests
frombs4importBeautifulSoup

#获取网页
#《乌合之众》网页的URL
url='https://wp.forchange.cn/psychology/11069/'
#请求网页
res=requests.get(url)
#打印响应的状态码
print(res.status_code)
#将响应内容的编码格式设置为utf-8
res.encoding='utf-8'

#解析网页
#解析请求到的网页，得到BeautifulSoup对象
bs=BeautifulSoup(res.text,'html.parser')

#搜索书籍信息的父节点<div>
info_tag=bs.find('div',class_='res-attrs')
#搜索每条信息的节点<dl>
info_list=info_tag.find_all('dl')

#创建字典，存储书籍信息
info_dict={}

#遍历搜索结果，提取文本内容，存储到字典中
forinfoininfo_list:
#提取信息提示项<dt>的元素内容
key=info.find('dt').text[:-2]
#提取书籍信息<dd>的元素内容
value=info.find('dd').text
#将信息添加到字典中
info_dict[key]=value

#打印查看字典中的书籍信息
print(info_dict)
```

还记得爬虫第一课说过的爬虫流程么？爬虫的流程主要可以分为三步，分别是：获取网页、解析网页以及存储数据。
其中，解析网页这一步实现的就是从 HTML 文档中提取想要的数据，这也是爬虫流程的重头戏。
解析网页可以分为两个子步骤：解析数据，提取数据。



在我们今天的任务中，目标数据在网页的 HTML 文档中，因此，要解析的是 HTML 文档。
上节课我们做了准备工作：认识 HTML 文档的结构，这节课就正式开始解析并提取数据！
但 HTML 文档的结构有一定的规则，需要配合 Python 的第三方库，才能高效地解析并从中提取网页数据。

1.3 功能拆解

接下来，我将对上方程序进行拆解，看看程序是如何实现“获取网页、解析数据、提取数据”这三个功能。

代码拆解

第 1 和 4-12 行实现了获取网页，得到了包含网页的 HTML 文档的 Response 对象。都是爬虫第一课学过的知识，可以读一遍代码复习一下。

第 2 和 14-16 行实现了解析数据（即解析 HTML 文档）。看起来很简单吧，除去模块导入，只有第 16 行一行代码。但是新知识却不少，不能掉以轻心。

第 18-36 行实现了提取数据并存储到字典中的功能。这部分是本节课的重中之重，除了知识学习，更重要的是综合运用学到的知识，解决问题。

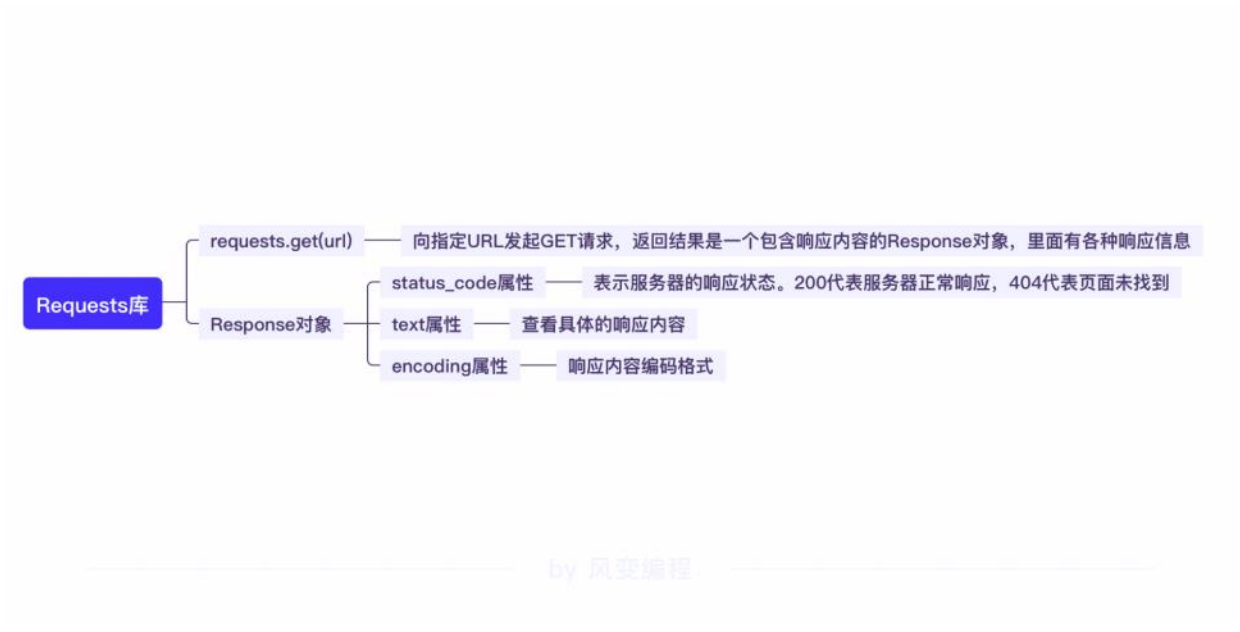
以上就是今天我们要实现的代码。

体验并拆解完代码，我们正式开始学习。
按照爬虫的流程，第一步是获取网页，也就是发起网页请求，获取网页源代码。

2. 获取网页

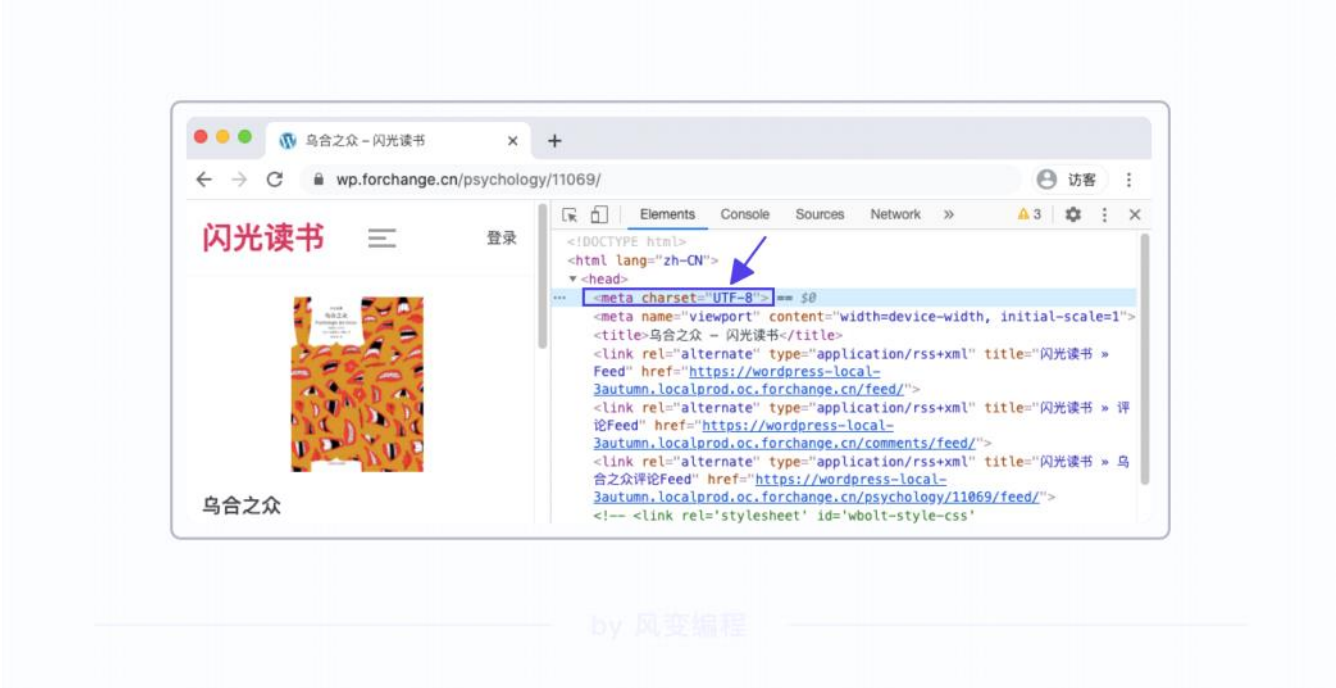
本任务中，要获取的是包含 HTML 文档的 Response 对象。
这是爬虫课程第一节就介绍的内容，用到的知识还记得么？带你复习一下。

2.1 复习 requests 库



用来获取网页的库是 requests 库。

发起网页请求的语法是requests.get(url)，参数 url 是网页的 URL（即网址），返回值是 Response 对象。
是否请求成功看响应状态码，获取响应状态码的语法是Response 对象.status_code，如果请求成功，响应状态码为 200。
为防止获取的内容乱码，要设定响应内容的编码格式，语法是Response 对象.encoding = '编码格式'，编码格式可以用枚举法：① 先试 utf-8；② 不行再试一下 GBK。
除了枚举以外，我们还学习了另一种方法来直接确定编码格式，就是在网页的 HTML 文档中，找到<meta>元素的charset属性，该属性里存放着网页的编码格式，可以将该编码格式做为 Response 对象 encoding 属性的值。
比如，我们的目标网页，<meta>元素为<meta charset="UTF-8">，则网页编码格式是：utf-8。



2.2 功能块练习

复习完 requests 库的知识点以后，请你来获取《乌合之众》书籍信息页的网页。

获取网页

```
#获取网页
#导入requests库

#网页的URL

url=

#请求网页，得到Response对象

res=

#打印响应状态码，查看是否请求成功

#将响应内容的编码格式设置为utf-8
```

答案：

```
# 获取网页
# 导入 requests 库
import requests

# 网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页，得到 Response 对象
res = requests.get(url)
# 打印响应状态码，查看是否请求成功
print(res.status_code)
# 将响应内容的编码格式设置为 utf-8
res.encoding = 'utf-8'
```

```
200
```

到这里，获取网页的知识复习完毕，也成功得到包含目标页面 HTML 文档 的 Response 对象。
获取完网页后，可以进行爬虫的第二步：数据的解析、提取。

3. 解析数据

不过在提取之前，需要先解析数据，而解析数据需要使用 bs4 库。

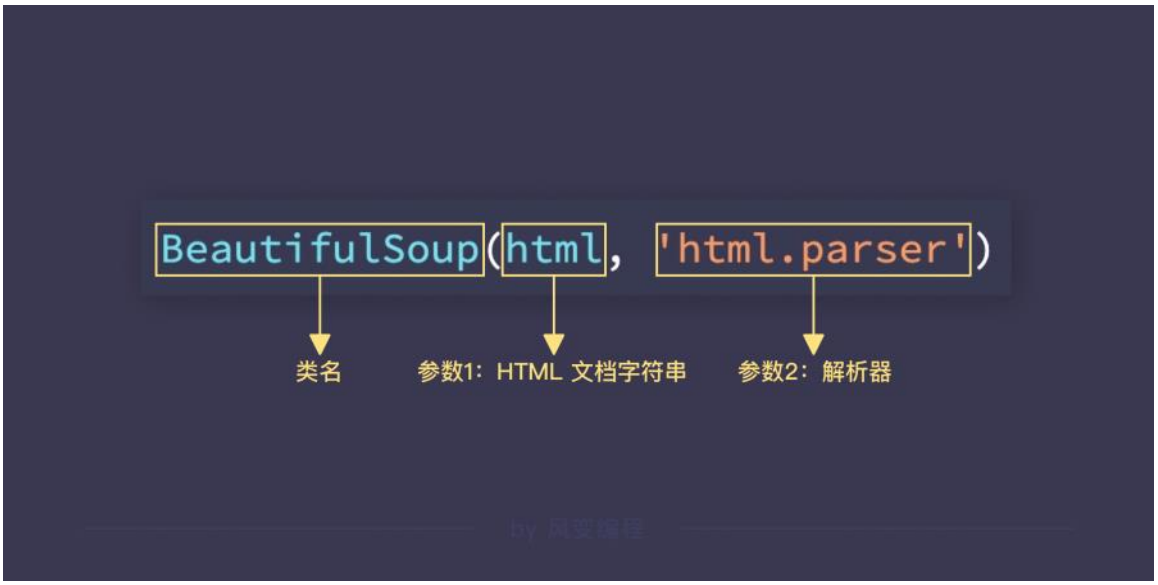
3.1 解析与 bs4 库

bs4 库是一个可以方便地从 HTML 文档中提取数据的 Python 第三方库，通常也称为“Beautiful Soup库”，bs4 的 4 表示版本。我们今天就用它来提取数据。
就像处理 csv、Excel 一样，Python 要处理文件和文档，需要先生成一个 Python 对象，比如 csv 对象，工作簿对象，工作表对象等。
处理获取到的网页也是一样。本节课中，获取的是 HTML 网页，解析数据就是要将 HTML 文档转化为 Python 程序可处理的 Python 对象。
解析 HTML 文档的工具是解析器。
就像建筑图纸有一本使用说明书，如果说 HTML 文档是搭建网页的建筑图纸，那解析器就是 HTML 文档的使用说明书，这份说明书是给程序用的。

程序要依照解析器这个说明书，才能理解 HTML 文档中的元素、标签之间的结构关系，从而将 HTML 文档解析成程序能处理的数据。
解析器有不同种类，相互之间性能也有差异。我们今天要用的解析器是html.parser，html.parser 是 Python 标准库中一种内置的解析器。
有了 HTML 文档和解析器，就够了么？
还不行，有了材料，还得用起来，这就需要 bs4 库来帮忙，它负责依据说明书按图纸制造出模型。
也就是说，bs4 库在解析中的作用是：使用解析器解析 HTML 文档。
bs4 是第三方库，使用前需要先安装。学习系统中已经事先安装好，你直接导入使用即可。
本节课中，我们实际要用的是bs4库中的BeautifulSoup 类。
解析 HTML 文档的过程就是实例化 BeautifulSoup 类，得到 BeautifulSoup 对象的过程。
这个过程很简单：先导入 bs4 库中的 BeautifulSoup 类，然后实例化 BeautifulSoup 类。
首先是导入，用from…import…语句可以导入 bs4 库中的 BeautifulSoup 类。
一行代码就可以搞定，像这样：

```
1  from bs4 import BeautifulSoup
```

导入后，具体怎么用呢？来看一下实例化 BeautifulSoup 类的语法：BeautifulSoup(html, 'html.parser')。



实例化这个类时，通常要传入 2 个参数：第 1 个参数可以是字符串格式的 HTML 文档，指示要解析的内容；第 2 个参数可以是解析器的名字，用来标识怎样解

析文
档。
具体使用时，第一个参数要注意一下，数据类型是字符串。我们**获取网页得到的是包含 HTML 文档的 Response 对象，需要调用 Response 对象的 text 属性（即Response 对象.text），得到 HTML 文档的字符串形式。**
第二个参数，要传入解析器的名字，也是字符串类型，我们用到的解析器是'html.parser'。
除了 html.parser，BeautifulSoup 也支持其他类型的解析器，不同的解析器有不同优缺点，此处不展开讲，如果感兴趣可以简单看一下下表。

解析器	使用方法	优势	劣势
Python标准库	BeautifulSoup(markup, "html.parser")	·Python的内置标准库 ·执行速度适中 ·文档容错能力强	·Python 2.7.3 or 3.2.2前的版本中文档容错能力差
lxml HTML 解析器	BeautifulSoup (markup, "lxml")	·速度快 ·文档容错能力强	·需要安装C语言库
lxml XML 解析器	BeautifulSoup(markup, ["lxml-xml"]) BeautifulSoup(markup, "xml")	·速度快 ·唯一支持XML的解析器	·需要安装C语言库
html5lib	BeautifulSoup(markup, "html5lib")	·最好的容错性 ·以浏览器的方式解析文档 ·生成HTML5格式的文档	·速度慢 ·不依赖外部扩展

3.2 功能块练习

解析数据的知识讲解就到这里，来继续写项目代码吧。

解析数据

```
#导入requests库
importrequests
#从bs4中导入BeautifulSoup类


#《乌合之众》网页的URL
url='https://wp.forchange.cn/psychology/11069/'
#请求网页，并将结果赋值给变量res
res=requests.get(url)
#打印响应状态码，查看是否请求成功
print(res.status_code)
#设置响应内容的编码格式
res.encoding='utf-8'

#用BeautifulSoup和解析器'html.parser'解析请求到的网页


#打印查看解析结果
print(bs)
```

答案

```python
# 导入 requests 库
import requests
# 从 bs4 中导入 BeautifulSoup 类
from bs4 import BeautifulSoup

# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页，并将结果赋值给变量 res
res = requests.get(url)
# 打印响应状态码，查看是否请求成功
print(res.status_code)
# 设置响应内容的编码格式
res.encoding = 'utf-8'

# 用 BeautifulSoup 和解析器'html.parser'解析请求到的网页
bs = BeautifulSoup(res.text,'html.parser')

# 打印查看解析结果
print(bs)
```

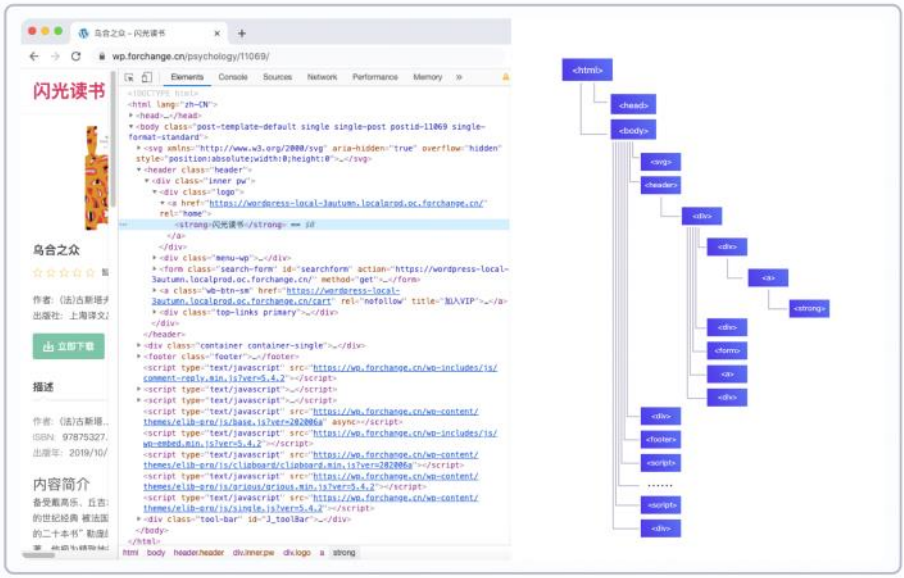
到这里，我们就将获取的数据解析成 BeautifulSoup 对象了。
但是，从打印的结果来看，BeautifulSoup 对象也是一个庞然大物。要怎么从 BeautifulSoup 对象中提取想要的数据呢？
也不是一件难事，虽然打印出的 BeautifulSoup 对象看起来内容很多，但从 BeautifulSoup 对象中提取网页数据却很方便。
为什么说方便呢？这归功于 BeautifulSoup 对象数据的结构，数据的结构决定了提取数据的方式。

4. 提取数据

在此之前，我们了解的对象中：
1）列表的结构是序列形式，列表元素按顺序排列，用索引取值；
2）字典的结构是映射形式，键和值一一对应，用键取值；
3）Excel 工作表对象的内部结构则是单元格按行、列排列，可以按行、单元格取值。
BeautifulSoup 对象的数据结构和你之前接触过的都不太一样，BeautifulSoup 对象是一种树状结构。

4.1 BeautifulSoup和Tag对象

以《乌合之众》书籍信息页为例，具体看一下。



BeautifulSoup 对象中的每个节点都是另一种 Python 对象：Tag 对象。例如，<html>节点，<head>节点，<body>节点，<header>节点等都是 Tag 对象。

BeautifulSoup 对象代表整个 HTML 文档。Tag 对象则与 HTML 文档中的元素一一对应。如果我们想找到某个 HTML 元素，让程序去找对应的 Tag 对象就可以了。

就像 HTML 文档中元素嵌套着元素一样，BeautifulSoup 对象中，节点和节点之间也是层层嵌套的关系。比如，<html>节点中嵌套着<head>节点和<body>节点；<body>节点中又嵌套着<svg>节点、<header>节点等。

在 BeautifulSoup 对象树状图中，对于相互连接的两个节点：上层节点称为下层节点的 parent 节点，也称父节点；下层节点称为上层节点的 child 节点，也称子节点。

父节点中嵌套着子节点，也嵌套着子节点的子节点。

这就像一张族谱，找到了其中一个人，就能顺藤摸瓜，找到他的子子孙孙。

此外，在树状图中，与某节点有关联的节点中：位于该节点上层的所有节点都是它的“父辈节点”，位于该节点下层的所有节点，都是它的“子孙节点”。

再啰嗦一句，就像族谱中的关系一样，父辈中包括“父”，子孙中包括“子”。

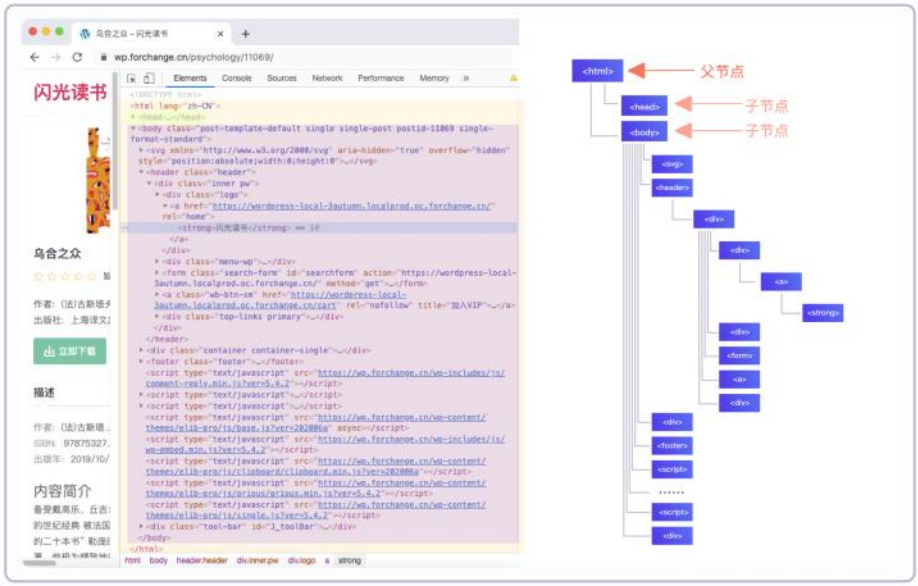
在使用的时候，我们不用把树状图画出来，可以直接看 HTML 文档中元素的嵌套与缩进关系，判断元素对应节点之间的父子关系。

什么意思呢？看两个例子：

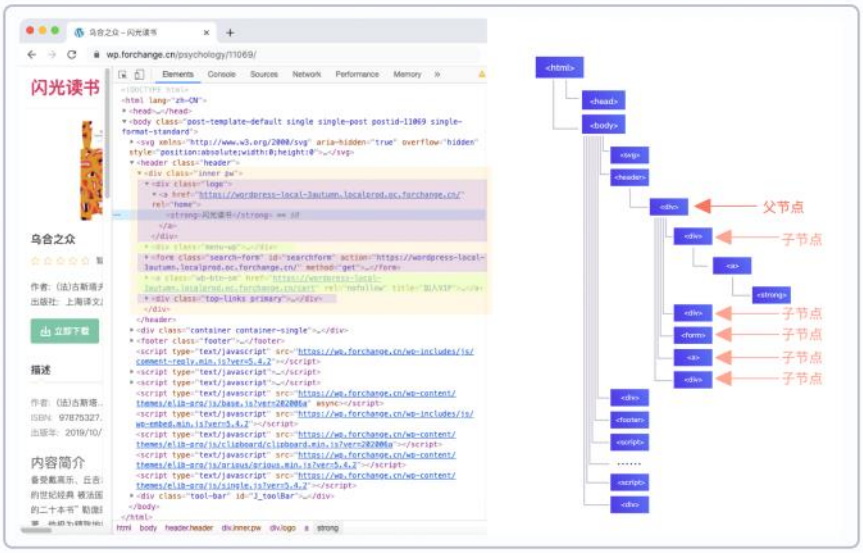
1）下图中，父节点（<html>节点）中嵌套着 2 个子节点：

a.<head>节点、<body>节点；

b.<html>节点是图中其他所有节点的父辈节点。



2)下图中，父节点（ 节点）中嵌套着 5 个子节点：
a. 节点、 节点、<form>节点、<a>节点、 节点；
b.<form>节点属于<body>节点的子孙节点。



特别要说明的是，BeautifulSoup 对象是所有 Tag 对象的父辈节点；反过来讲，所有 Tag 对象都是 BeautifulSoup 对象的子孙节点。
来做个练习巩固一下吧。
根据下方局部 HTML 文档和对应的 BeautifulSoup 对象结构图，回答问题：

单选题

下列关于 Tag 对象的说法，正确的一项是：

A.
2 号<svg>节点中嵌套着 3 号 节点。

B.
BeautifulSoup 对象中的 3 号 节点对应着 HTML 文档中 元素的开始标签 。

C.
号<body>节点是图中所有其他节点的父辈节点，也就是说，<body>节点中嵌套着图中其他所有节点。

D.
4 号<a>节点是 5 号节点的父节点，反之节点也是 4 号<a>节点的父节点。

分析

A: 2号 svg 节点和3号 div 节点间不存在嵌套关系；B：BeautifulSoup 对象中的节点与 HTML 文档中的元素（而不是元素的标签）一一对应；D：5号 strong 节点是4号 a 节点的子节点。
从 HTML 文档中看，<body>元素中嵌套着截图中其他元素；从 BeautifulSoup 对象树状图中也能看出，其他节点都是<body>节点的子孙节点。

了解了 BeautifulSoup 对象中节点间的嵌套关系，就可以依据这种关系提取节点了。具体怎么做呢？

接下来介绍两种方案：一是用.元素名操作；二是用 find_all() 和 find() 方法。
先了解第一种方案：.元素名操作。

4.2 .元素名

.元素名是 BeautifulSoup 对象和 Tag 对象通用的操作，可以获取嵌套在当前节点内的子孙节点。
元素名是指 HTML 文档中元素的名称，.元素名的执行结果是得到一个 Tag 对象。

以BeautifulSoup 对象为例，对应的语法是**BeautifulSoup 对象.元素名**。
对于 BeautifulSoup 对象来说，所有 Tag 对象都嵌套在 BeautifulSoup 对象中。
因此，可以试着用BeautifulSoup 对象.元素名取任意节点。
但是，如果一个 HTML 文档中有同名元素，只能返回第一个元素名匹配的 Tag 对象。
来看个例子。
BeautifulSoup 对象.元素名


```
<html>
<head>
<meta charset="utf-8">
<title>大川神的爬虫世界</title>
</head>
<body>
<div id="header">
<h1>川神教你HTML</h1>
</div>
<div class="poems" id="section1">
<h2>静夜思</h2>
<h3>李白（唐）</h3>
<p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
</div>
<div class="poems" id="section2">
<h2>早发白帝城</h2>
<h3>李白（唐）</h3>
<p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
</div>
</body>
</html>
```

```
from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

```
27  # 解析 HTML 文档
28  bs = BeautifulSoup(html, 'html.parser')
29  # 用.div获取<div>节点
30  div_tag = bs.div
31  # 打印查看结果  只取到第一个div元素
32  print(div_tag)
[2]  ✓ 0.2s
...  <div id="header">
     <h1>川神教你HTML</h1>
     </div>
```

解析后用BeautifulSoup 对象.div提取的结果是什么呢?
阅读代码，重点看第30 行，然后运行程序，查看终端结果。
成功获取一个 div 节点!

从显示可以看到，**只获取到了第一个 节点!**

BeautifulSoup 对象.元素名的用法就先看到这里。
操作很方便吧? 只要知道 HTML 元素的元素名，就可以了获取到 Tag 对象。但是只能获取 BeautifulSoup 对象中第一个元素名匹配的 Tag 对象。
前面说过，.元素名是 BeautifulSoup 对象和 Tag 对象的通用操作。接下来，看一下 Tag 对象如何使用.元素名操作。

语法是: Tag 对象.元素名。
功能是获取嵌套在 Tag 对象中第一个元素名匹配的 Tag 对象。
查看下面的代码，重点注意代码里的第 31 行，然后运行看看:

```
1  from bs4 import BeautifulSoup
2
3  html = '''
4      <html>
5      <head>
6      <meta charset="utf-8">
7      <title>大川神的爬虫世界</title>
8      </head>
9      <body>
10     <div id="header">
11     <h1>川神教你HTML</h1>
12     </div>
13     <div class="poems" id="section1">
14     <h2>静夜思</h2>
15     <h3>李白（唐）</h3>
16     <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
17     </div>
18     <div class="poems" id="section2">
19     <h2>早发白帝城</h2>
20     <h3>李白（唐）</h3>
21     <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
22     </div>
23     </body>
24     </html>
25     '''
```

```
26  # 解析 HTML 文档
27  bs = BeautifulSoup(html, 'html.parser')
28  # 用.div获取<div>节点
29  div_tag = bs.div
30  # 从<div>节点中获取<h1>节点
31  h1_tag = div_tag.h1
32
33  # 打印查看结果
34  print(h1_tag)
```

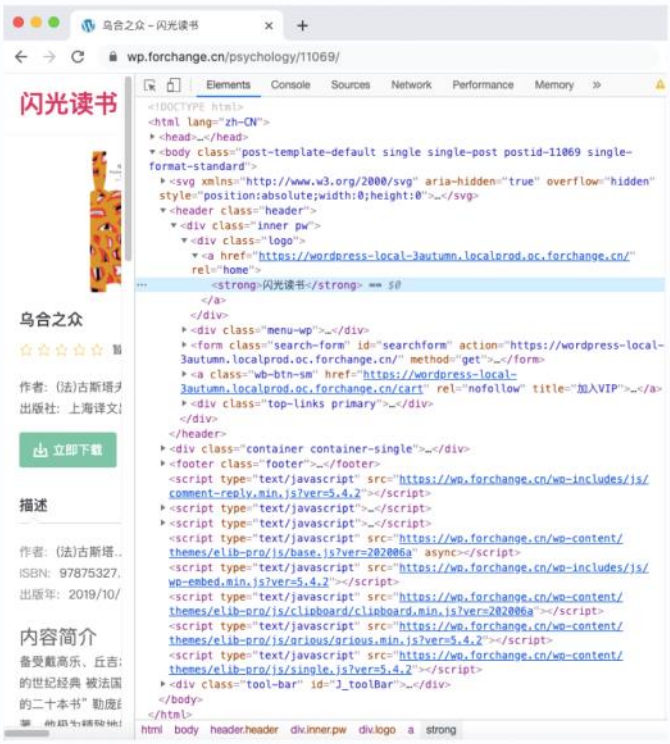
```
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用.div获取<div>节点
div_tag = bs.div
# 从<div>节点中获取<h1>节点
h1_tag = div_tag.h1
# 打印查看结果
print(h1_tag)
```

.元素名操作还有一个拓展用法：.元素名.元素名.……，BeautifulSoup 对象和 Tag 对象都可以这样操作。

能这样“顺藤摸瓜”的原因就是 BeautifulSoup 对象和 Tag 对象，以及 Tag 对象和 Tag 对象之间的嵌套关系。
例如：用bs.div.h1也可以获取<h1>节点，这个操作你简单了解一下就可以。

.元素名操作的讲解就到这里，主要用法归纳起来是这样：

可以看出，.元素名操作很方便，语法也很简单。但使用的局限性同样显而易见，如果网页结构复杂、或者需要提取多个同名元素，.元素名就不太适用了。比如，《乌合之众》书籍简介页的 HTML 文档，就复杂得多：



所以接下来要介绍另外的方法：find() 和 find_all()。BeautifulSoup 对象和 Tag 对象都可以使用这两种方法，通过传入参数，精准定位到我们需要的 Tag 对象。

4.3 find_all() 和 find() 方法

先看find_all() 方法。

这个函数的功能是：搜索所有满足条件的 Tag 对象。

find_all()返回的结果是一个类似列表的可迭代对象，里面包含了所有满足参数条件的 Tag 对象。

要使用这个方法，涉及到两个问题：一、从什么范围找？二、怎么用？

我们来一个个看。

首先，从什么范围找呢?

可以从 BeautifulSoup 对象中找，也可以从 Tag 对象中找。

如果从 BeautifulSoup 对象中搜索，语法是：BeautifulSoup 对象.find_all()；

从 Tag 对象中搜索，语法是：Tag 对象.find_all()。

接下来，怎么用呢？以 BeautifulSoup 对象为例，来看看具体语法。

find_all() 方法的参数就是需要满足的条件。我们主要用到两种参数：HTML 元素名和 HTML 元素属性。
来分别看一下这两种参数：
第一种参数是：HTML 元素名。
传入HTML 元素名作为 find_all() 方法的参数，即可搜索所有元素名匹配的 Tag 对象。
用元素名查找对应的 Tag 对象时，每次只能传入一个元素名，而且要以字符串的形式传入。
例如：BeautifulSoup 对象.find_all('div')，可以获取 BeautifulSoup 对象中所有的 元素。
依然以刚才看到的 HTML 文档为例。阅读下方的代码，重点看 29 行，然后运行代码并查看终端结果。

```python
# find_all 体验

from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

```
28  # 解析 HTML 文档
29  bs = BeautifulSoup(html, 'html.parser')
30  # 用find_all()获取所有<div>节点
31  div_all = bs.find_all('div')
32
33  # 打印查看结果
34  print(div_all)
```

✓ 0.2s

```
[<div id="header">
<h1>川神教你HTML</h1>
</div>, <div class="poems" id="section1">
<h2>静夜思</h2>
<h3>李白（唐）</h3>
<p>床前明月光，疑是地上霜。<br/>举头望明月，低头思故乡。</p>
</div>, <div class="poems" id="section2">
<h2>早发白帝城</h2>
<h3>李白（唐）</h3>
<p>朝辞白帝彩云间，千里江陵一日还。<br/>两岸猿声啼不住，轻舟已过万重山。</p>
</div>]
```

```
# find_all 体验
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find_all()获取所有<div>节点
div_all = bs.find_all('div')
# 打印查看结果
print(div_all)
```

从显示的结果可以看到，3 个 节点都拿到了。

第二种参数是：HTML 元素属性。
传入HTML 元素属性作为 find_all() 方法的参数，就可以依据 HTML 元素的属性（如 id，class，href）来搜索对应的 Tag 对象。
传入 HTML 元素属性时，要用参数名 = 参数值的形式，一次可以传入 0 到多个属性。参数名通常是元素的属性名，参数值就是对应的属性值。
这里需要注意的是：HTML 的 class 属性与 Python 的保留关键字 class 重复。因此，作为参数使用 class 属性时，要加一个_，写作class_，避免混淆。
例如：BeautifulSoup 对象.find_all(class_='poems')用来搜索 BeautifulSoup 对象中，所有拥有属性class='poems'的元素对应的 Tag 对象。
阅读下面的代码，重点看下方代码的第 29 行，然后运行代码并查看终端结果。

```python
from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

```python
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find_all()获取所有含属性class="poems"的HTML元素对应的节点
poems_all = bs.find_all(class_='poems')

# 打印查看结果
print(poems_all)
```

✓ 0.2s

```
[<div class="poems" id="section1">
<h2>静夜思</h2>
<h3>李白（唐）</h3>
<p>床前明月光，疑是地上霜。<br/>举头望明月，低头思故乡。</p>
</div>, <div class="poems" id="section2">
<h2>早发白帝城</h2>
<h3>李白（唐）</h3>
<p>朝辞白帝彩云间，千里江陵一日还。<br/>两岸猿声啼不住，轻舟已过万重山。</p>
</div>]
```

```python
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
```

```
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find_all()获取所有含属性class="poems"的HTML元素对应的节点
poems_all = bs.find_all(class_='poems')
# 打印查看结果
print(poems_all)
```

从终端显示的结果可以看到，我们拿到了两个 class 属性为class="poems"的节点。

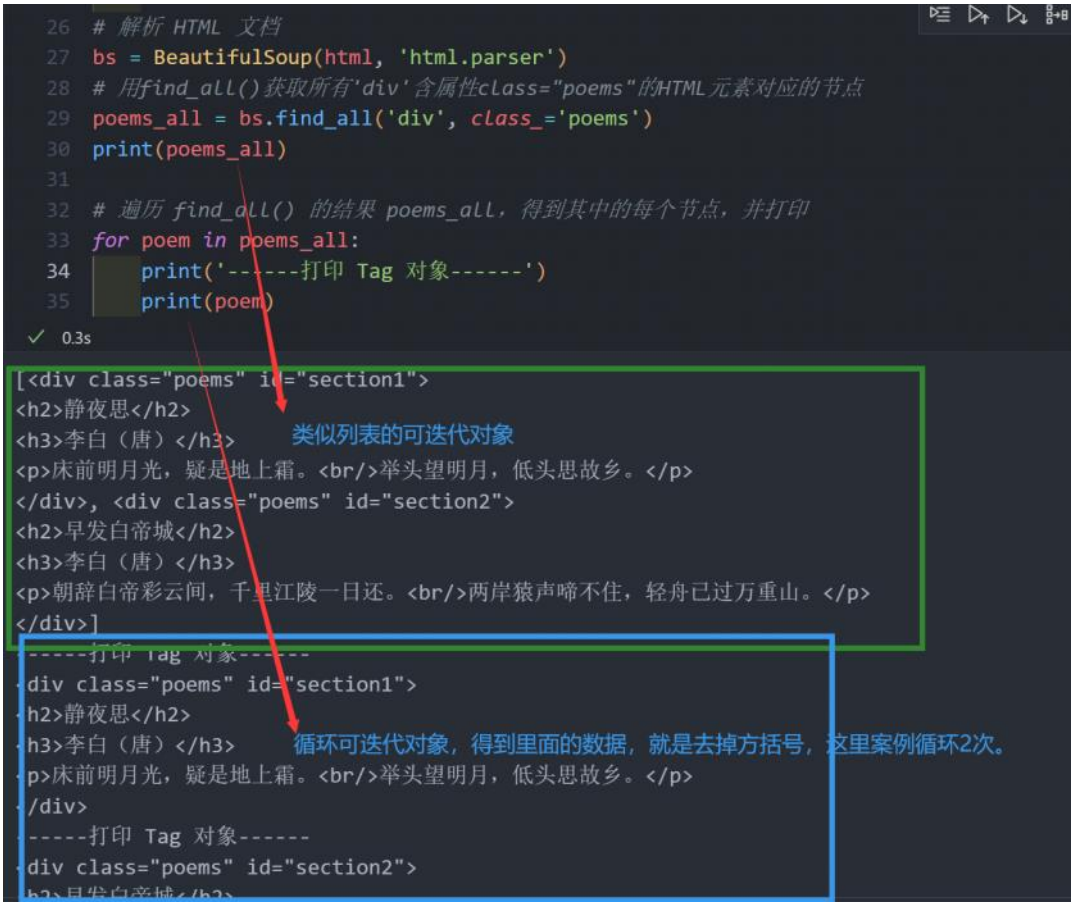
由于 find_all() 返回的都是满足所有参数条件的 Tag 对象，因此，可以结合使用上述两种参数，更准确定位到 Tag 对象。同时使用 HTML 元素名和 HTML 元素属性作为搜索条件时，要把 HTML 元素名作为第 1 个参数，后面接 0 到多个 HTML 元素属性。

例如，想在 BeautifulSoup 对象中搜索所有元素名为div，并且拥有属性class="poems"的元素对应的 Tag 对象，语法应该是：BeautifulSoup 对象.find_all('div', class_='poems')。

需要注意的是，**find_all() 返回的结果并不是 Tag 对象，而是 Tag 对象组成的一个类似列表的可迭代对象。要拿到其中的 Tag 对象，通常需要for 循环来帮忙。**

来看下方的代码理解一下吧，重点看 31-34 行，然后运行一下代码：

```
1  from bs4 import BeautifulSoup
2
3  html = '''
4      <html>
5      <head>
6      <meta charset="utf-8">
7      <title>大川神的爬虫世界</title>
8      </head>
9      <body>
10     <div id="header">
11     <h1>川神教你HTML</h1>
12     </div>
13     <div class="poems" id="section1">
14     <h2>静夜思</h2>
15     <h3>李白（唐）</h3>
16     <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
17     </div>
18     <div class="poems" id="section2">
19     <h2>早发白帝城</h2>
20     <h3>李白（唐）</h3>
21     <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
22     </div>
23     </body>
24     </html>
25     '''
```




```
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find_all()获取所有'div'含属性class="poems"的HTML元素对应的节点
poems_all = bs.find_all('div', class_='poems')
print(poems_all)
# 遍历 find_all() 的结果 poems_all，得到其中的每个节点，并打印
for poem in poems_all:
    print('------打印 Tag 对象------')
    print(poem)
```

到这里，find_all() 有什么功能、怎么使用、如何处理它的结果都学完了。

学完 find_all()，再理解 find() 相当容易。

find() 方法使用范围和 find_all() 一样，也是 BeautifulSoup 对象和 Tag 对象都能用的方法；find() 方法的参数要求也和 find_all() 相同。

唯一不同的是，find() 方法返回的结果是一个 Tag 对象，更准确地说是：搜索范围内，满足参数条件的第一个 Tag 对象。这一点和.元素名操作有点儿像。

依然以 BeautifulSoup 对象为例，语法是：

? 你能判断 find() 的搜索结果么？根据下面的 HTML 文档，回答问题。

```
<html>
<head>
<meta charset="utf-8">
<title>大川神的爬虫世界</title>
</head>
<body>
<div id="header">
<h1>川神教你HTML</h1>
</div>
<div class="poems" id="section1">
<h2>静夜思</h2>
<h3>李白（唐）</h3>
<p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
</div>
<div class="poems" id="section2">
<h2>早发白帝城</h2>
<h3>李白（唐）</h3>
<p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
</div>
</body>
</html>
```

单选题

用BeautifulSoup 对象.find('div', class_='poems')搜索，能得到什么呢？

A.

包含“静夜思”的 div 节点和包含“早发白帝城”的div节点。

B.

包含“静夜思”的 div 节点。

解析

A 中的两个节点都满足参数条件，但是 find() 方法只返回其中的第一个 Tag 对象，因此选 B。

眼见为实，重点看 29 行，运行代码，看看结果。

```python
from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

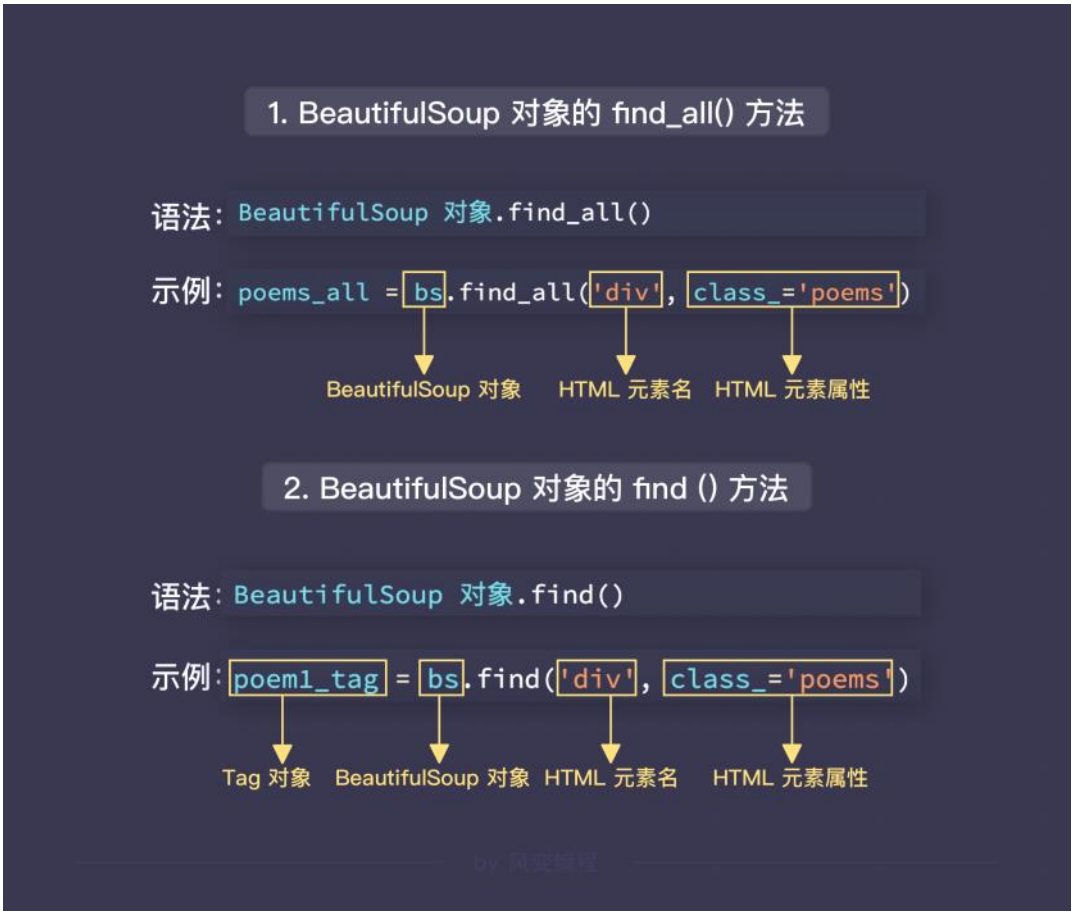
```python
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find()获取第一个满足参数条件的节点
poem1_tag = bs.find('div', class_='poems')

# 打印查看结果
print(poem1_tag)
```

✓ 0.3s

```
<div class="poems" id="section1">
<h2>静夜思</h2>
<h3>李白（唐）</h3>
<p>床前明月光，疑是地上霜。<br/>举头望明月，低头思故乡。</p>
</div>
```

从终端显示的结果可以看到，我们只取到了《静夜思》所在的 Tag 对象。

以上就是 BeautifulSoup 对象的 find_all() 和 find() 方法，我们来归纳一下吧。



接下来，再来看看 Tag 对象的 find_all() 和 find() 方法。
Tag 对象使用 find_all() 和 find() 方法时，用法和 BeautifulSoup 对象完全相同，语法是这样的：

看起来很简单吧？这就用起来吧，这次你自己写代码。
补充下方代码的第 31 行，用 find() 方法从 Tag 对象poem1_tag中提取诗名《静夜思》所在的<h2>节点。

```
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
```

```
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find()获取第一个满足参数条件的节点
poem1_tag = bs.find('div', class_='poems')
# 用find()从poem1_tag中提取<h2>节点
h2_tag =
# 打印查看结果
print(h2_tag)
```

答案:

```
from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

```
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')

# 用find()获取第一个满足参数条件的节点
poem1_tag = bs.find('div', id='section1')

# 用find()从poem1_tag中提取<h2>节点
h2_tag = poem1_tag.find('h2')

# 打印查看结果
print(h2_tag)
```

✓ 0.3s

```
<h2>静夜思</h2>
```

顺便说一下，这里用Tag 对象.元素名也能得到一样的结果。

```
h2_tag = poem1_tag.h2
```

我来总结一下.元素名操作，find() 方法，find_all() 方法的区别与联系：

1).元素名可以通过元素名找到一个 Tag 对象；

2)find_all()可以根据元素名和属性搜索所有满足条件的 Tag 对象；

3) find()和 find_all() 的差别是，find() 只能返回第一个 Tag 对象。

「.元素名」操作，find_all()方法和find()方法

功能	使用方法	参数	结果与功能
.元素名	BeautifulSoup 对象.素名 Tag 对象.元素名	—	结果是一个 Tag 对象，获取第一个元素名匹配的 Tag 对象。
find_all()	BeautifulSoup 对象.find_all() Tag 对象.find_all()	HTML 元素名 HTML 元素属性	结果是一个类似列表的可迭代对象，包含所有满足参数条件的 Tag 对象。
find()	BeautifulSoup 对象.find() Tag 对象.find()	HTML 元素名 HTML 元素属性	结果是一个 Tag 对象，只返回第一个满足参数条件的Tag 对象。

得到 Tag 对象后，我们相当于拿到了 HTML 元素。要从元素中拿到具体的内容，还要从 Tag 对象中进一步提取。

4.4 从 Tag 对象中提取内容

常见的两种内容有 HTML 元素的文本内容和属性值。

那么，怎样才能从 Tag 对象中取出对应元素的文本内容和属性值呢?

首先看提取元素的文本文案:

提取 Tag 对象对应元素文本内容的方法，和获取 Response 对象的文本内容很像：用 text 属性。

语法是Tag 对象.text，功能是获取 Tag 对象中的所有文本内容。

刚才获取到的诗名《静夜思》还是带着标签的 Tag 对象，我修改了下第 31 行代码，用.text获取到了脱掉标签的诗名。

运行下面的代码，查看获取文本内容的结果。

```python
from bs4 import BeautifulSoup

html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
```

获取h2标签的文本内容

```python
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find()获取第一个满足参数条件的节点
poem1_tag = bs.find('div', class_='poems')
# 用find()从poem1_tag中提取<h2>节点，用.text获取文本内容
h2_text = poem1_tag.find('h2').text
# 打印查看结果
print(h2_text)
```

使用text属性获取标签里的内容

```
✓ 0.2s
静夜思
```


```python
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
```

```
# 用find()获取第一个满足参数条件的节点
poem1_tag = bs.find('div', class_='poems')
# 用find()从poem1_tag中提取<h2>节点，用.text获取文本内容
h2_text = poem1_tag.find('h2').text
# 打印查看结果
print(h2_text)
```

接下来看，**提取对应元素的元素属性：**

提取 Tag 对象对应元素的属性，和用字典的键获取值的方式很像，用方括号。

语法是：Tag 对象['属性名']，可用来提取对应元素的属性值。

下节课你会用到它，先简单认识一下。阅读下面的代码，重点看 13 和 29 行，然后运行看看结果。

```
1  from bs4 import BeautifulSoup
2
3  html = '''
4      <html>
5      <head>
6      <meta charset="utf-8">
7      <title>大川神的爬虫世界</title>
8      </head>
9      <body>
10     <div id="header">
11     <h1>川神教你HTML</h1>
12     </div>
13     <div class="poems" id="section1">
14     <h2>静夜思</h2>
15     <h3>李白（唐）</h3>
16     <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
17     </div>
18     <div class="poems" id="section2">
19     <h2>早发白帝城</h2>
20     <h3>李白（唐）</h3>
21     <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
22     </div>
23     </body>
24     </html>
25     '''
```

```
26 # 解析 HTML 文档
27 bs = BeautifulSoup(html, 'html.parser')
28 # 用find()获取第一个满足条件节点的id属性值
29 poem1_id = bs.find('div', class_='poems')['id']
30
31 # 打印查看结果
32 print(poem1_id)
```

✓ 0.2s

```
section1
```

```
from bs4 import BeautifulSoup
html = '''
    <html>
    <head>
    <meta charset="utf-8">
    <title>大川神的爬虫世界</title>
    </head>
    <body>
    <div id="header">
    <h1>川神教你HTML</h1>
    </div>
    <div class="poems" id="section1">
    <h2>静夜思</h2>
    <h3>李白（唐）</h3>
    <p>床前明月光，疑是地上霜。<br>举头望明月，低头思故乡。</p>
```

```
    </div>
    <div class="poems" id="section2">
    <h2>早发白帝城</h2>
    <h3>李白（唐）</h3>
    <p>朝辞白帝彩云间，千里江陵一日还。<br>两岸猿声啼不住，轻舟已过万重山。</p>
    </div>
    </body>
    </html>
    '''
# 解析 HTML 文档
bs = BeautifulSoup(html, 'html.parser')
# 用find()获取第一个满足条件节点的id属性值
poem1_id = bs.find('div', class_='poems')['id']
# 打印查看结果
print(poem1_id)
```

知识学完了，具体怎么做才能又快又准确地找到装着目标数据的 Tag 对象呢？

这部分要综合使用我们学过的知识：

1）HTML 文档元素、标签和属性的理解；
2）浏览器开发者工具的使用；
3）BeautifulSoup 对象和Tag 对象嵌套关系的理解；
4）find()与find_all()方法的使用；
5）Tag 对象.text和Tag 对象['属性名']的灵活使用。

随着你爬的具体网页越多，会越熟练，也会有自己的习惯。

我在这里通过本节课的项目给你介绍些基础做法。

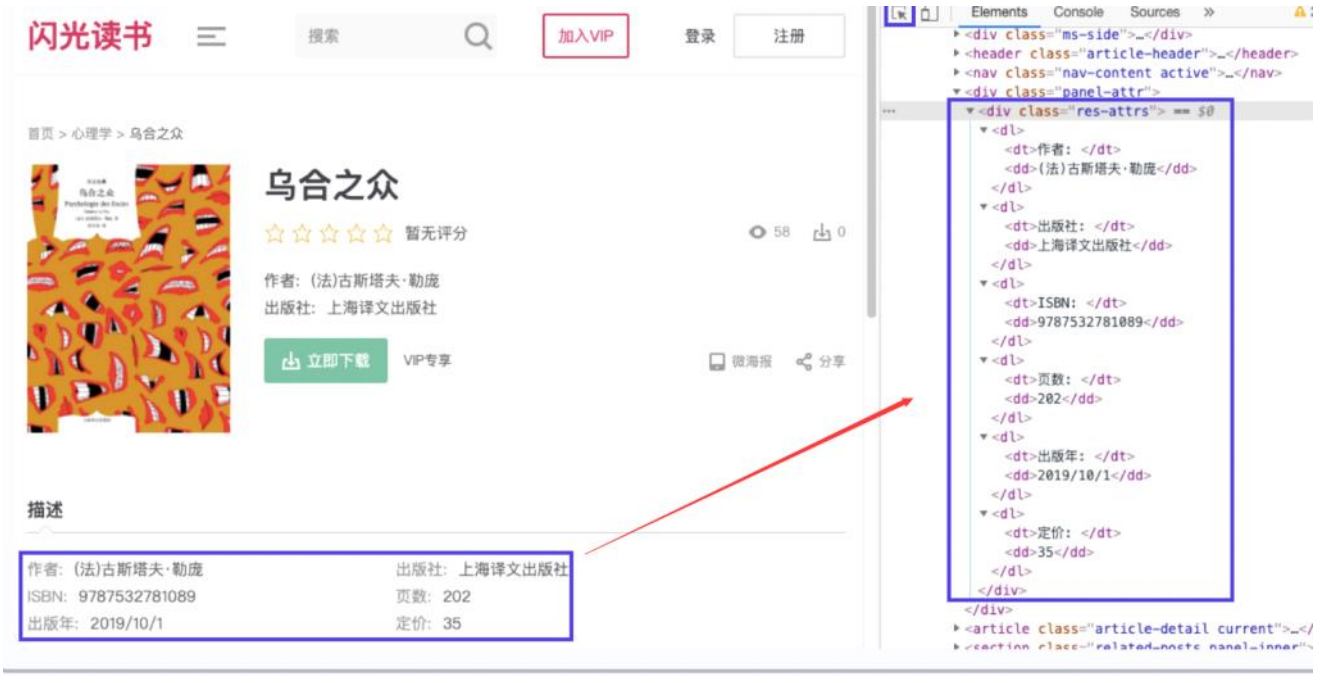
4.5 功能块练习

接下来，需要你对照着要爬取的目标网页，边动手操作边理解。准备好了么？

第 1 步，用 Chrome 浏览器打开一个新的标签页，在地址栏粘贴以下URL：

https://wp.forchange.cn/psychology/11069/。

第 2 步，打开浏览器的开发者工具，用指针工具在网页源码中找到目标数据。
再提醒一下，打开开发者工具的快捷键：Windows 快捷键：Ctrl + Shift + I；Mac 快捷键：Cmd + Opt + I。

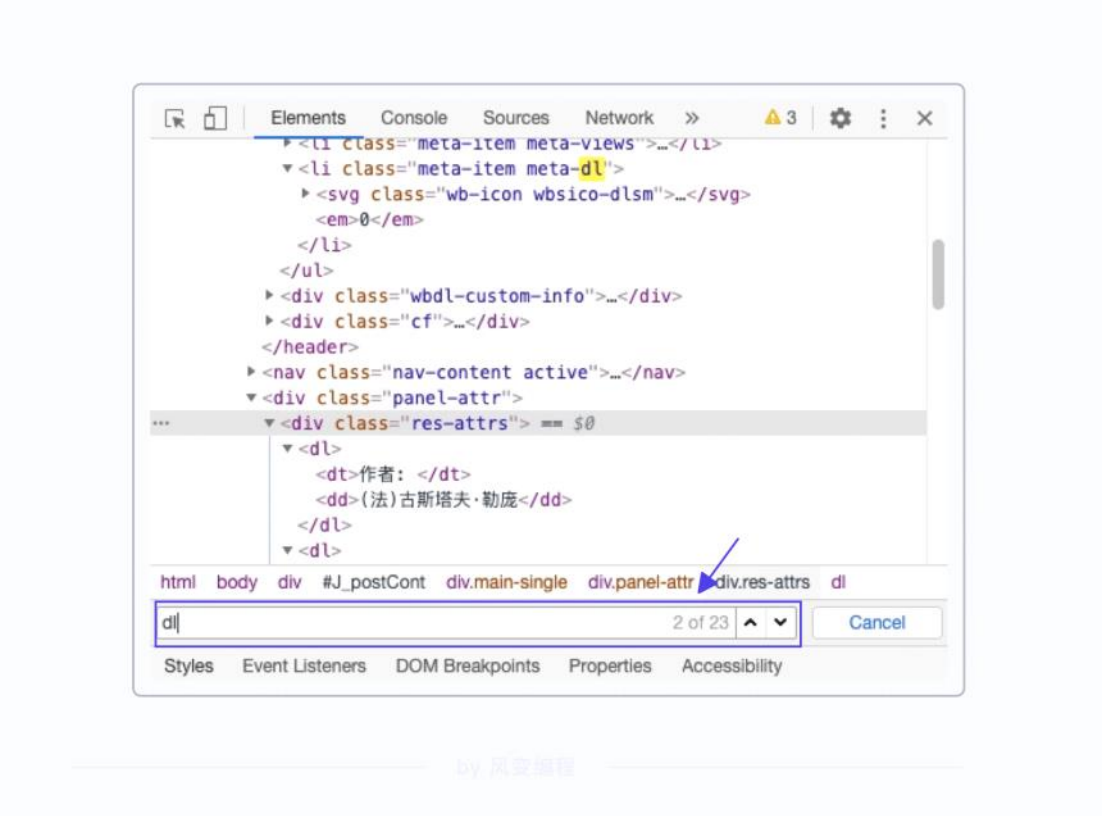


在 Elements 中展开折叠的 HTML 元素，查看目标数据。
从网页源码中可见，我们想要的每一项信息都在<dl>元素中。

第 3 步，重点来了：在 BeautifulSoup 对象中定位目标数据。
我们知道，解析后，整个 HTML 文档转换成了 BeautifulSoup 对象，那么，要用 BeautifulSoup 对象.find_all() 直接找这些元素对应的<dl>节点么？

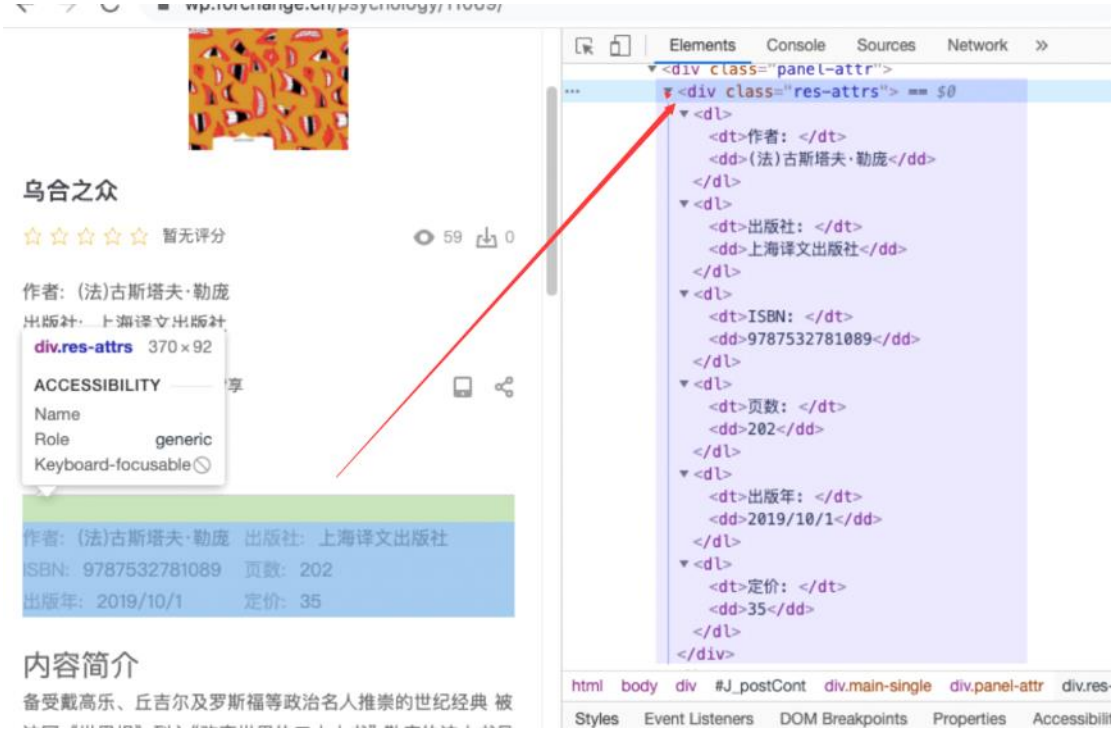
先别急，我们用开发者工具的搜索功能检查一下 HTML 文档，看是不是只有这 6 个<dl>元素。
提示：
点击Elements栏，然后打开搜索框，输入dl并回车。
打开搜索框方式：Ctrl+F (windows) /Cmd+F(mac) 。

搜索框右侧显示了搜索到的结果数量。



搜索框搜到的结果不止 6 个。通过上下键查找发现，有些是元素名，还有找到多余的结果。
这种情况，直接用 find_all() 找<dl>节点会把多余的节点一起拿回来，这显然不合适。
然而，<dl>元素也没有可以用来进一步精准定位的属性。怎么办呢?
我们再把视角往外层标签看一下。

可以发现，这些<dl>元素被包围在一个 class 属性为class="res-attrs"的 元素中。



这个 节点是我们想要的<dl>节点的父节点，如果能找到这个父节点，就可以在父节点中用Tag 对象.find_all()搜索内部的所有<dl>节点，进而拿到我们想要的信息

能不能实现呢？
再来用搜索功能检查一下，这个 元素在 HTML 文档中是不是只有一个。
元素名 div 就不用搜了，能看到的范围就不止一个。
但是 元素还有一个 class 属性：class="res-attrs"。
我们来搜索一下res-attrs看看。

看起来不错，只搜到了一个。

那么，上面的设想就可以执行了：先找到<dl>节点的父节点 ，再找所有<dl>节点。
思路可行，接下来就看如何用代码实现啦！
首先，在 BeautifulSoup 对象中获取父节点 。代码怎么写呢？我给出了几个选项，你来选择正确答案吧，请听题：

单选题

根据元素名 div 和属性 class="res-attrs"，用 find() 方法 在 BeautifulSoup 对象 bs 中搜索 节点，并赋值给变量 info_tag 的代码应该是：

A.info_tag = bs.find('div', class_='res-attrs')

B.info_tag = bs.find('div', class='res-attrs')

C.info_tag = bs.find(class_='res-attrs', 'div')

回答正确

B 选项：HTML元素的 class 属性作为参数名要写成 class_。
C 选项：同时使用HTML元素名和HTML元素属性作为搜索条件时，要把HTML元素名作为第 1 个参数。

先找目标数据的父节点是一个常用做法，这可以缩小搜索范围，把提取数据这个任务从大海捞针变成小河捞针，甚至小碗捞针。
拿到父节点之后，可以用 `find_all()` 方法在父节点中搜索所有<dl>节点。
依然由你来选择正确的代码。请听题

单选题

根据元素名 dl，在 Tag 对象 info_tag 中搜索所有 <dl>节点 （共 6 个），并赋值给变量 info_list 的正确代码是：

A.info_list = info_tag.find_all('dl')

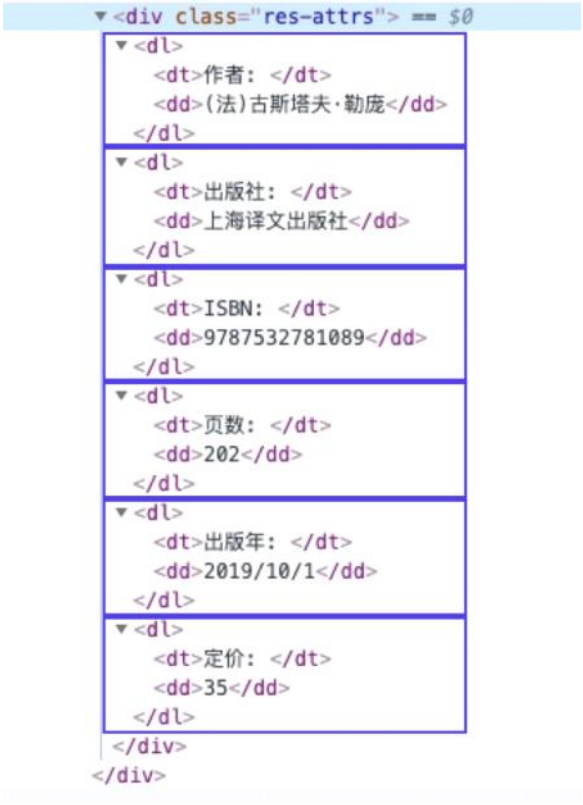
B.info_list = info_tag.find('dl')

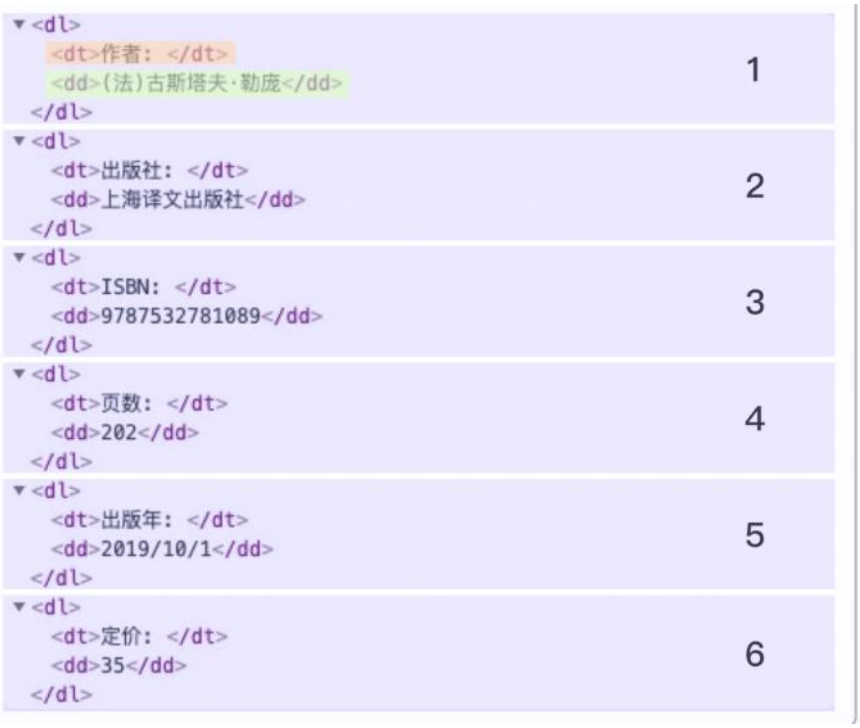
C.info_list = info_tag.find_all(dl)

回答正确

B 选项：用 find() 只能获取当前节点中的第一个dl节点。C 选项：用HTML元素名作为 find_all() 的参数时，元素名dl应写成字符串形式'dl'。

用 `find_all()` 得到的是一个包含所有<dl>节点的类似列表的可迭代对象。
接下来，需要用 `for` 循环遍历 `find_all()` 的结果，得到每个<dl>节点，再从<dl>节点中继续提取。





每一个<dl>节点中有两个节点：<dt>节点和<dd>节点，可以用 find() 方法分别提取。

再梳理一下这个过程：

一、用 find() 获取父节点 ；

二、用 find_all() 获取所有<dl>节点；

三、用 for 循环遍历 find_all() 的结果得到每个<dl>节点；

四、在每个<dl>节点中，用 find() 获取<dt>节点和<dd>节点

思路清晰么？你来写代码实现这个过程吧！
阅读注释和代码，补全第 18 到 29 行缺失的代码，提取<dt>和<dd>节点。

```
import requests
from bs4 import BeautifulSoup
# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'
# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')
# 用 find() 搜索书籍信息的父节点<div>
info_tag =
# 用 find_all() 搜索每条信息的节点<dl>
info_list =
# 用 for 循环遍历搜索结果，在每个<dl>节点中继续提取
    # 用 find() 提取信息提示所在的<dt>节点
    dt =
    print(dt)
    # 用 find() 提取书籍信息所在的<dd>节点
    dd =
    print(dd)
```

答案:

```
import requests
from bs4 import BeautifulSoup

# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res =  requests.get(url)

# 打印响应的状态码
print(res.status_code)

# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'

# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')


# 用 find() 搜索书籍信息的父节点<div>
info_tag = bs.find('div', class_='res-attrs')

# 用 find_all() 搜索每条信息的节点<dl>
info_list = info_tag.find_all('dl')

# 用 for 循环遍历搜索结果，在每个<dl>节点中继续提取
for info in info_list:
```

```
    # 用 find() 提取信息提示所在的<dt>节点
    dt = info.find('dt').text

    print(dt)
    # 用 find() 提取书籍信息所在的<dd>节点
    dd = info.find('dd').text

    print(dd)
```

✓ 0.3s

```
200
作者:
(法)古斯塔夫·勒庞
出版社:
上海译文出版社
ISBN:
9787532781089
页数:
202
出版年:
2019/10/1
定价:
35
```

```
import requests
from bs4 import BeautifulSoup
# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res =  requests.get(url)
# 打印响应的状态码
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'
# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')

# 用 find() 搜索书籍信息的父节点<div>
info_tag = bs.find('div', class_='res-attrs')
# 用 find_all() 搜索每条信息的节点<dl>
info_list = info_tag.find_all('dl')
# 用 for 循环遍历搜索结果，在每个<dl>节点中继续提取
for info in info_list:
    # 用 find() 提取信息提示所在的<dt>节点
    dt = info.find('dt').text
    print(dt)
    # 用 find() 提取书籍信息所在的<dd>节点
    dd = info.find('dd').text
    print(dd)
```

这样我们就一层层提取到了想要的数据。但是，如果程序只写成这样，我们的数据取出来，打印查看完，就找不到了。
下节课要把数据都存到文件中，今天这一步，我们最好先把拿到的数据暂存起来备用。

由于<dt>元素和<dd>元素的内容是一一对应的映射关系，符合 Python 中字典类型的结构特征。
我们可以用字典来暂时存放目标数据。
这次看我写的代码吧。不用修改，仔细读一遍第 23-36 行，运行并查看结果。

```python
import requests
from bs4 import BeautifulSoup

# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'

# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')

# 搜索书籍信息的父节点<div>
info_tag = bs.find('div', class_='res-attrs')
# 搜索每条信息的节点<dl>
info_list = info_tag.find_all('dl')

# 创建字典，存储书籍信息
info_dict = {}

# 遍历搜索结果，提取文本内容，存储到字典中
for info in info_list:
    # 提取信息提示项<dt>的元素内容
    key = info.find('dt').text[:-2]
    # 提取书籍信息<dd>的元素内容
    value = info.find('dd').text
    # 将信息添加到字典中
    info_dict[key] = value

# 打印查看字典中的书籍信息
print(info_dict)
```

✓ 0.3s Python

```
200
{'作者': '(法)古斯塔夫·勒庞', '出版社': '上海译文出版社', 'ISBN': '9787532781089', '页数': '202', '出版年': '2019/10/1', '定价': '35'}
```


```python
import requests
from bs4 import BeautifulSoup
# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'
# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')
# 搜索书籍信息的父节点<div>
info_tag = bs.find('div', class_='res-attrs')
# 搜索每条信息的节点<dl>
info_list = info_tag.find_all('dl')
# 创建字典，存储书籍信息
info_dict = {}
```

```
# 遍历搜索结果，提取文本内容，存储到字典中
for info in info_list:
    # 提取信息提示项<dt>的元素内容
    key = info.find('dt').text[:-2]
    # 提取书籍信息<dd>的元素内容
    value = info.find('dd').text
    # 将信息添加到字典中
    info_dict[key] = value
# 打印查看字典中的书籍信息
print(info_dict)
```

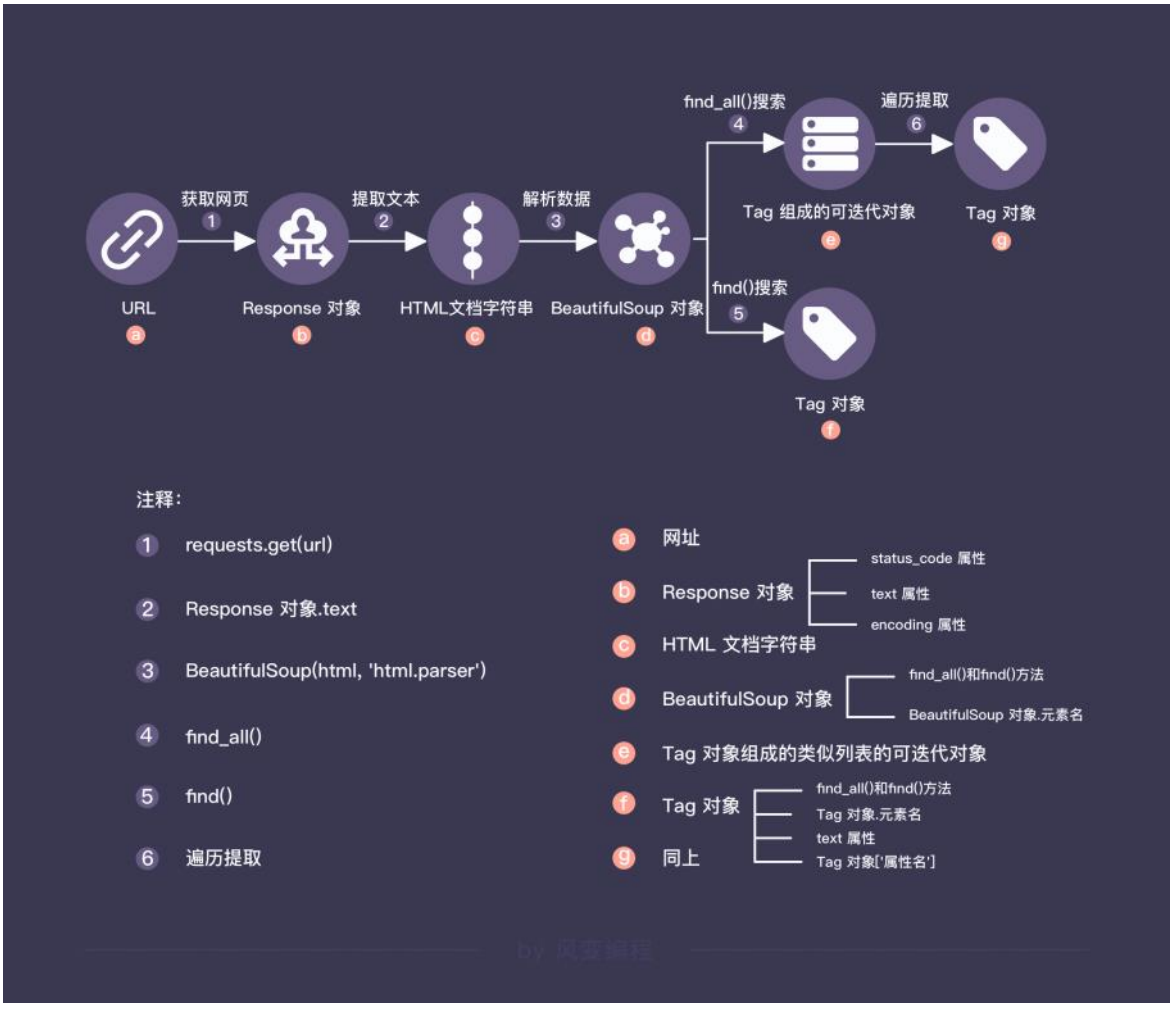
再认真理解一下用字典存储数据的用法吧，给你 1 分钟的时间!

如果你观察得仔细，会发现第 29 行的代码最后用到了字符串的切片功能：`info.find('dt').text[:-2]`;
这是因为<dt>元素内容结尾有一个冒号和一个空格，为了数据美观，用字符串.[:-2]切掉了字符串后两个字符。

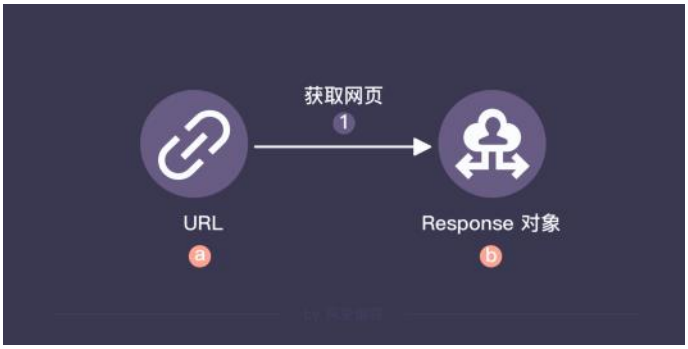
这是一个小技巧，如果爬到的数据前后有你不想要的内容，可以用字符串的各种方法帮忙哟。这里不展开，你自己实践的时候可以多尝试。

5. 程序实现与总结

接下来，我们梳理一下从获取网页到提取数据的流程：



首先，要获取网页：



获取网页的细节有：

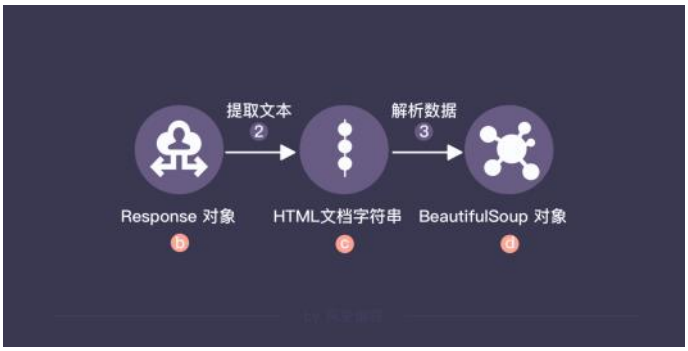
1）用 requests 库的 get() 函数请求网页，参数是网页的 URL，即网址，获取网页的结果是得到 Response 对象。

2）通过 Response 对象的响应状态码可以判断请求是否成功。

3）设置 Response 对象的 encoding 属性可以指定编码格式。

4）通过 Response 对象的 text 属性可以获取字符串形式的 HTML 文档。

然后，要解析数据：



解析数据的细节有：

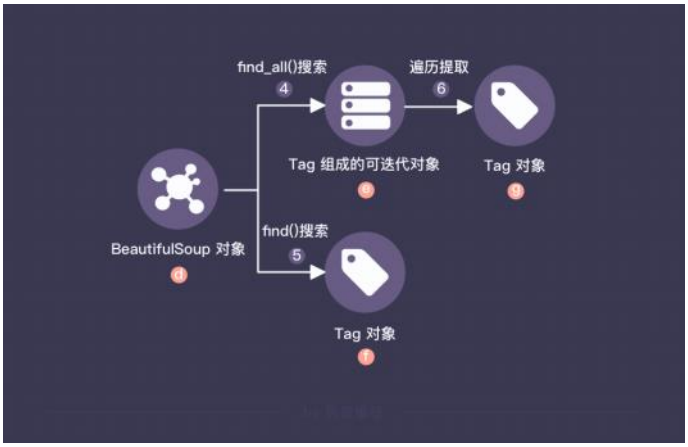
1）通过实例化 bs4 库的 BeautifulSoup 类实现，结果是一个 BeautifulSoup 对象。

2）实例化 BeautifulSoup 类时，需要传入两个参数：一个是 HTML 文档字符串；一个是解析器。

3）我们用的解析器是 Python 标准库内置的'html.parser'。

4）BeautifulSoup 对象代表整个 HTML 文档，是一种树状结构，每一个节点都是 Tag 对象。

解析后，要提取数据：



提取数据的细节有：

1）目的是从 BeautifulSoup 对象中找到目标 Tag 对象，并提取对应 HTML 元素中的数据。

2）Tag 对象与 HTML 文档中的元素一一对应。通过搜索 Tag 对象就能找到相应的 HTML 元素。

3）可以用 find_all() 和 find() 方法搜索 Tag 对象。

4）通过Tag 对象.text和Tag 对象['属性名']，可以提取对应 HTML 元素的文本内容或属性值。

回顾完爬虫流程，你来重头写一遍代码吧。

```
# 导入 requests 库和 bs4 库中的 BeautifulSoup 类
```

```
# 获取网页
# 《乌合之众》网页的 URL
# 请求网页
# 打印响应的状态码，查看是否请求成功
# 将响应内容的编码格式设置为utf-8

# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象

# 用 find() 搜索书籍的信息的父节点
# 用 find_all() 搜索每条信息的节点

# 创建空字典 info_dict ，存储书籍信息

# 用for循环遍历搜索结果，得到每个Tag对象，并提取文本内容，存储到字典中
    # 提取提示项，获取元素文本内容，用[:-2]删除后两个字符
    # 提取书籍信息，获取元素文本内容
    # 将提示项和信息成对添加到字典中

# 打印查看字典中的书籍信息
print(info_dict)
```

答案

```
# 导入 requests 库和 bs4 库中的 BeautifulSoup 类
import requests
from bs4 import BeautifulSoup

# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码，查看是否请求成功
print(res.status_code)
# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'
# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')

# 用 find() 搜索书籍的信息的父节点
parent_tag = bs.find('div', class_='res-attrs')
# 用 find_all() 搜索每条信息的节点
dl_tag = parent_tag.find_all('dl')

# 创建空字典 info_dict ，存储书籍信息
infodict = {}

# 用for循环遍历搜索结果，得到每个Tag对象，并提取文本内容，存储到字典中
for item_tag in dl_tag:
    # 提取提示项，获取元素文本内容，用[:-2]删除后两个字符
    dt_tag = item_tag.find('dt').text[:-2]
    # 提取提示项，获取元素文本内容，用[:-2]删除后两个字符
    dd_tag = item_tag.find('dd').text
    # 将提示项和信息成对添加到字典中
    key = dt_tag
    infodict[key] = dd_tag
# 打印查看字典中的书籍信息
print(infodict)
```

```python
# 导入 requests 库和 bs4 库中的 BeautifulSoup 类
import requests
from bs4 import BeautifulSoup


# 获取网页
# 《乌合之众》网页的 URL
url = 'https://wp.forchange.cn/psychology/11069/'
# 请求网页
res = requests.get(url)
# 打印响应的状态码，查看是否请求成功
print(res.status_code)

# 将响应内容的编码格式设置为utf-8
res.encoding = 'utf-8'

# 解析网页
# 解析请求到的网页，得到 BeautifulSoup 对象
bs = BeautifulSoup(res.text, 'html.parser')


# 用 find() 搜索书籍的信息的父节点
parent_tag = bs.find('div', class_='res-attrs')

```

```python
# 用 find_all() 搜索每条信息的节点
dl_tag = parent_tag.find_all('dl')


# 创建空字典 info_dict ，存储书籍信息
infodict = {}


# 用for循环遍历搜索结果，得到每个Tag对象，并提取文本内容，存储到字典中
for item_tag in dl_tag:
    # 提取提示项，获取元素文本内容，用[:-2]删除后两个字符
    dt_tag = item_tag.find('dt').text[:-2]
    # 提取提示项，获取元素文本内容，用[:-2]删除后两个字符
    dd_tag = item_tag.find('dd').text

    # 将提示项和信息成对添加到字典中
    key = dt_tag
    infodict[key] = dd_tag

# 打印查看字典中的书籍信息
print(infodict)
```

0.3s

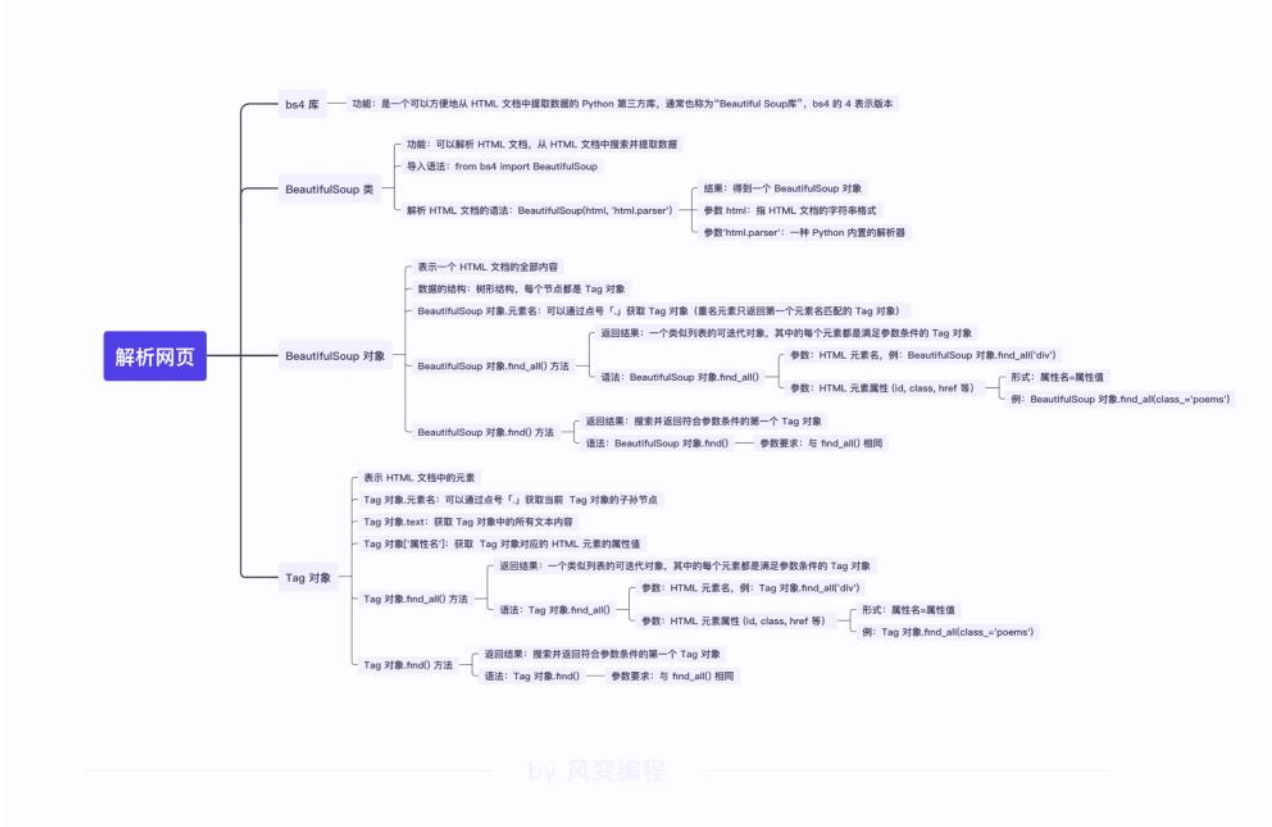
```
200
{'作者': '(法)古斯塔夫·勒庞', '出版社': '上海译文出版社', 'ISBN': '9787532781089', '页数': '202', '出版年': '2019/10/1', '定价': '35'}
```

5.2 本关总结

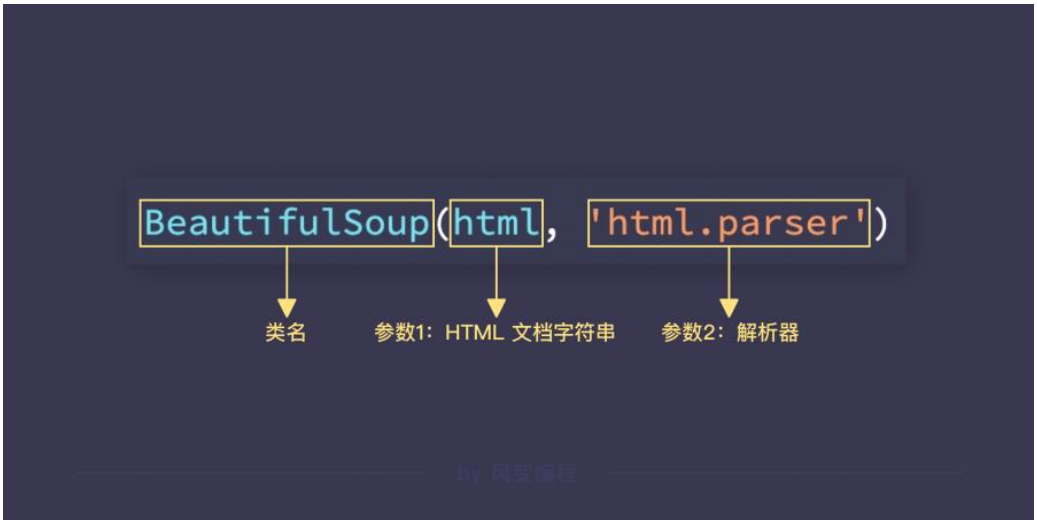
以上是本节课的所有知识点。

打铁需趁热，我们再总结下今天学到的知识，巩固一下记忆。

今天的项目中，我们学习了哪些新知识呢？来认真阅读一下本关的知识汇总图吧（单击 图片可放大）。



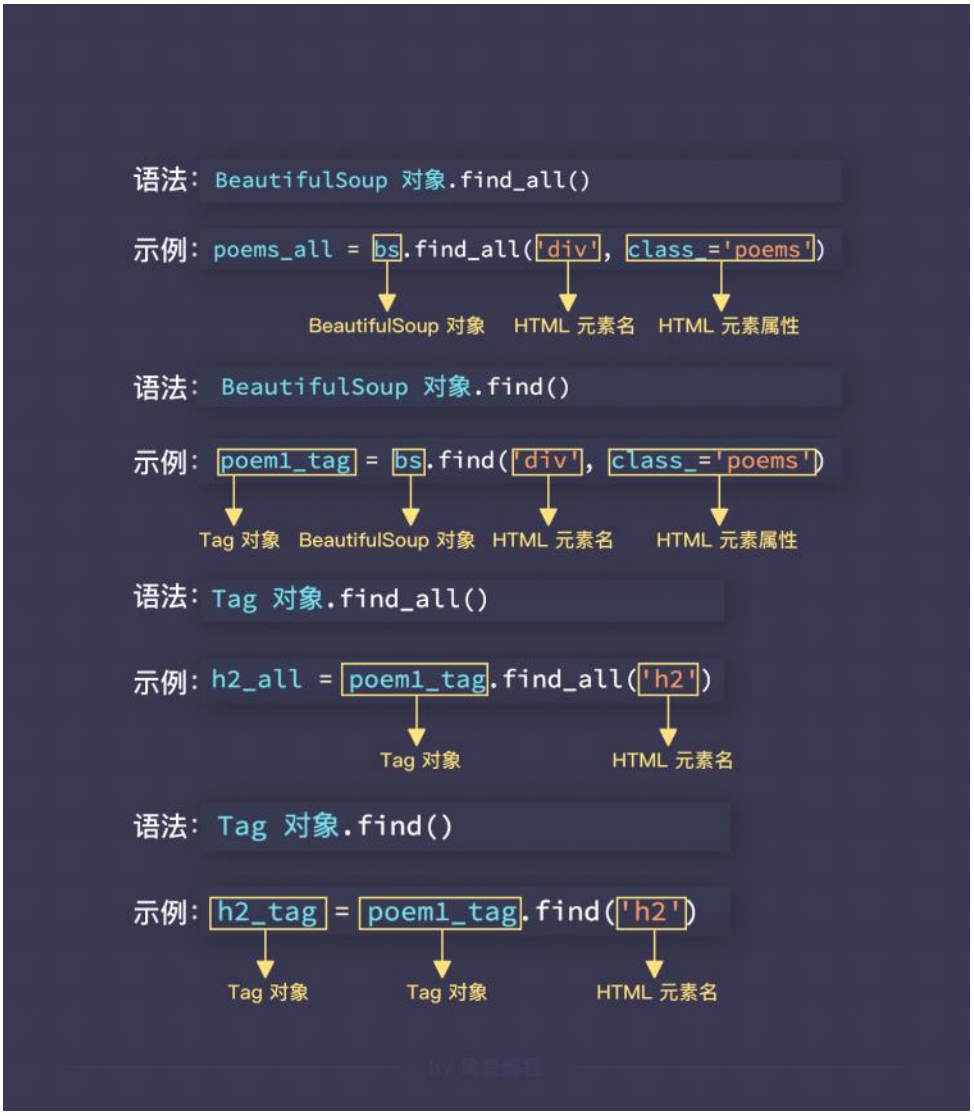
要关注的重点是用 `BeautifulSoup` 类解析网页的方法，以及从 `BeautifulSoup` 对象中提取目标数据的方法。
此外，理解爬虫整体流程也很重要，可以通过几张图来巩固一下：
解析数据



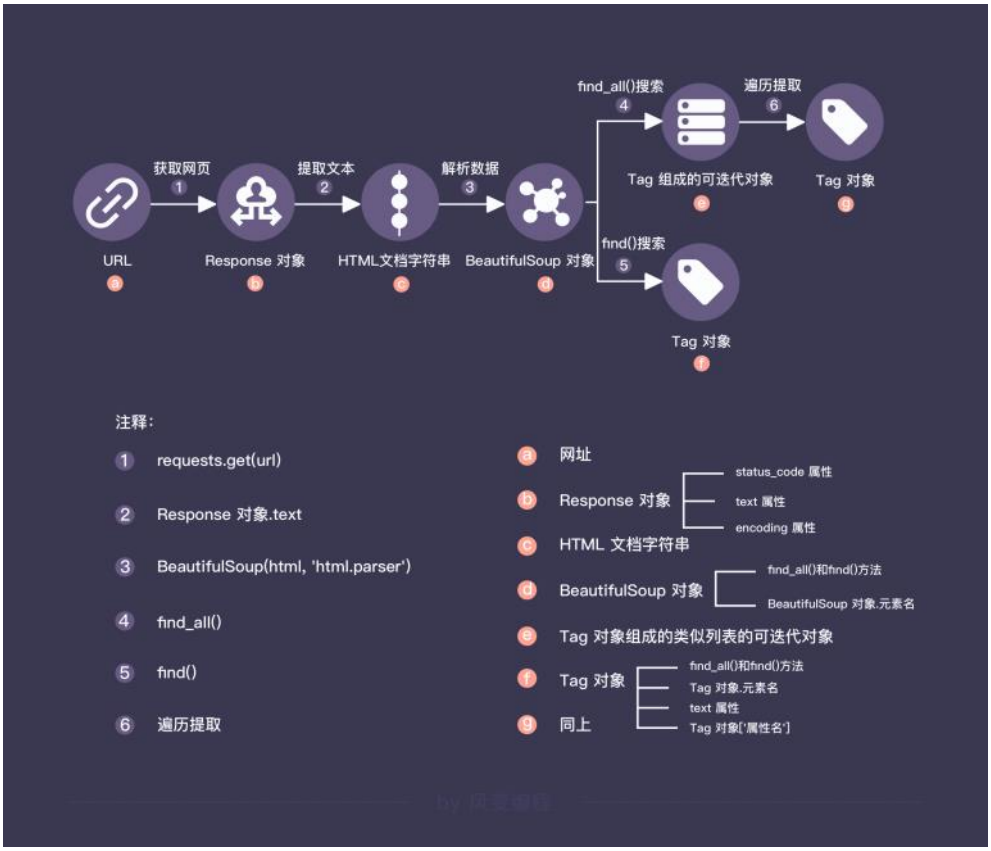
提取数据

「.元素名」操作，find_all()方法和find()方法

功能	使用方法	参数	结果与功能
.元素名	BeautifulSoup 对象.素名 Tag 对象.元素名	—	结果是一个 Tag 对象，获取第一个元素名匹配的 Tag 对象。
find_all()	BeautifulSoup 对象.find_all() Tag 对象.find_all()	HTML 元素名 HTML 元素属性	结果是一个类似列表的可迭代对象，包含所有满足参数条件的 Tag 对象。
find()	BeautifulSoup 对象.find() Tag 对象.find()	HTML 元素名 HTML 元素属性	结果是一个 Tag 对象，只返回第一个满足参数条件的Tag 对象。



爬虫流程



掌握得还好么？或许你觉得知识点没办法马上都记住，这没关系，编程语法都是开卷内容，多实践就好。
需要更多思考和沉淀的，是解决问题的思路。
可以试着养成习惯，从各个大小项目中总结归纳解决问题的逻辑。无论是逻辑本身，还是思考归纳的过程，或许都能给我们的生活注入新的能量。