

Python 网络爬虫技术





什么是网络爬虫

- 网络爬虫又称为网页蜘蛛，它是一种程序，可以按照既定规则，自动的抓取网络数据或执行网络请求。
- 互联网上存放了海量的数据，网站上一个一个的网页好比蜘蛛网，爬虫程序可以抓取网页上的数据，然后利用网页之间的关联关系，从一个网页跳转去另一个网页，继续执行抓取任务。
- 网络爬虫程序还可以模拟正常用户的行为，执行特定的网络请求，例如：用户登录，网站内容查询，新建商品订单等等。
- 网络爬虫只是一种技术，无所谓善恶，取决于使用者的目的。例如，搜索引擎采用爬虫技术给人们带来了便利，而各类抢票软件则给广大正常用户造成了不便。



爬虫技术应用

- 搜索引擎, 百度、谷歌
- 刷票软件, 12306、亚航
- 社交平台, 微博僵尸粉
- 电商类, 不同电商平台之间或某平台历史价格比较



爬虫技术应用

刷票类软件的现状

◆ 铁路

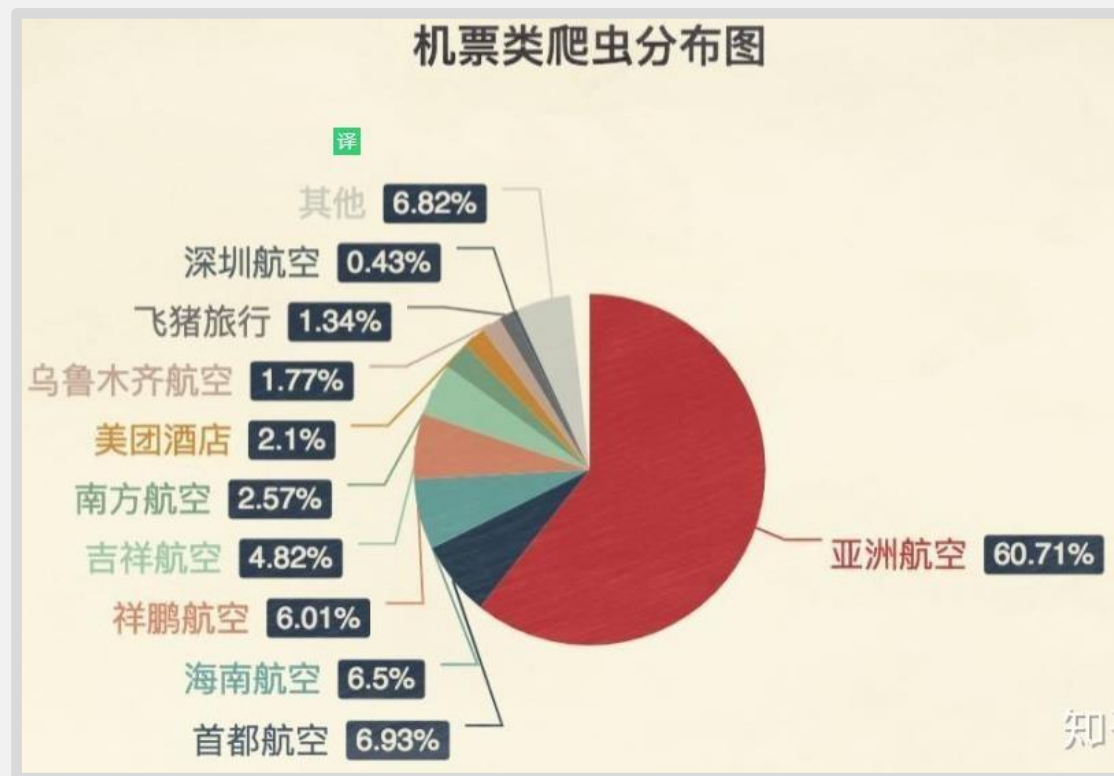
12306公开数据称：“春节前最高峰时1天内页面浏览量达813.4亿次，1小时最高点击量59.3亿次，平均每秒164.8万次”。在每秒164.8万次点击背后，不仅是全国人民急切的回家之心，还有无数刷票软件带来的天量点击。

◆ 航空

最悲情的航空公司：亚航（马来西亚）

亚航的初衷只是随机放出一些便宜的票来吸引游客，黄牛党们则利用爬虫软件，不断刷新亚航的票务接口，一旦出现便宜的票立刻拍下。

亚航规定，机票拍下半小时内不付款，票自动回到票池。而爬虫脚本里设定了精确的时间，到半小时后，爬虫会立刻再把机票拍下来，如此循环。直至有人向黄牛党预定这张机票，黄牛党就利用程序在亚航系统里放弃该票，然后0.001秒之后，用客户的名字预定了这张票。



推荐阅读：<https://zhuanlan.zhihu.com/p/396210073>



本章提要

1.网络基础知识



2.Requests库
的使用



网页爬取



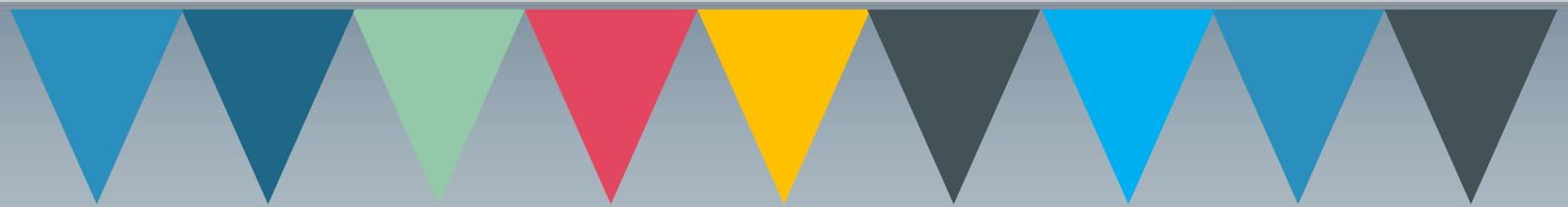
数据解析



3.BeautifulSoup库
的使用

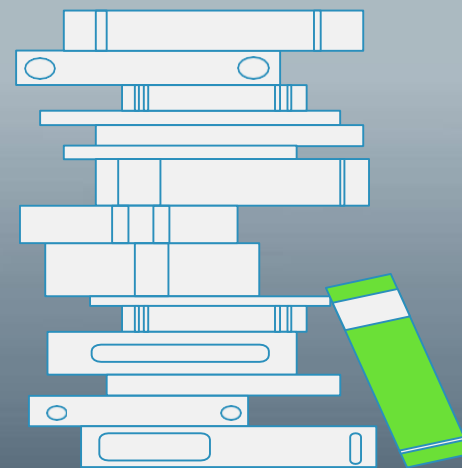


4.综合实例：豆瓣
书评的爬取



11.1

网络基础知识





网站访问流程



1. 用户通过浏览器发起Web请求，又称为（Request）。
2. 网络设备解析网址，寻找到Web服务器。
3. 服务器根据用户的请求内容，返回相关数据及文件，该过程称为响应（Response）。
4. 用户浏览器将所有接收数据整合，以指定方式展现给用户。



网站访问流程





浏览器中所输入的网站或网页地址的真正含义

标准称谓：统一资源定位符，

也可称为：URL (Uniform Resource Locator)

URL通常包含以下几部分信息

网络协议://服务器主机:端口/文件路径?发送数据

①

②

③

④

<http://lib.njtech.edu.cn/list.php?fid=9>

①

②

③

④



URL通常包含以下几部分信息

网络协议://服务器主机:端口/文件路径?发送数据

①

②

③

④

<http://lib.njtech.edu.cn/list.php?fid=9>

①

②

③

④

- ① 利用浏览器查看网页时，最常用的是http 协议（超文本传输协议）。
- ② 服务器地址通常是一个经过注册的域名或者 IP 地址。
- ③ 用户访问的网页文档在服务器主机的存储路径。
- ④ 用户提交到服务器的数据，例如查询关键字、翻页的页码等等。



网页= HTML文档，绝大多数展现在用户面前的网页都是由HTML脚本所构成的文件

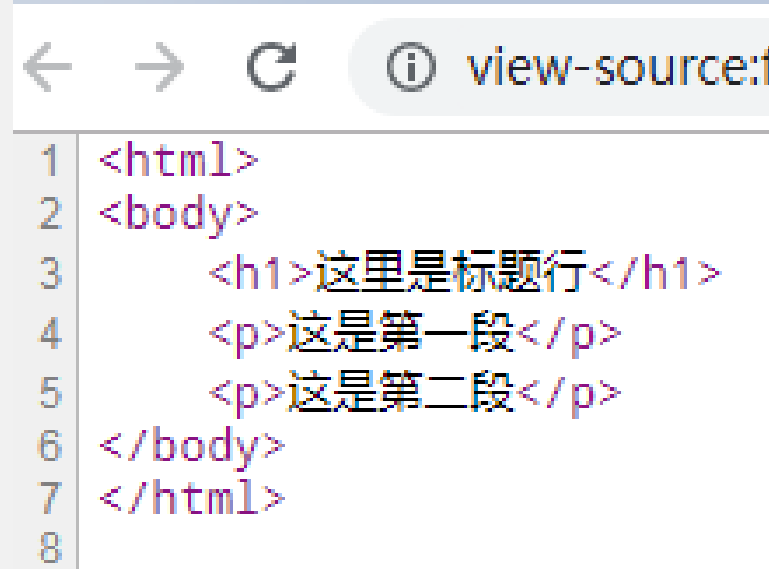
- ◆ HTML 指的是超文本标记语言 (Hyper Text Markup Language)
- ◆ HTML 不是一种编程语言，而是一种标记语言 (markup language)
- ◆ 标记语言是一套标记标签 (markup tag)
- ◆ HTML 使用**标记标签**来描述网页



HTML 标签

标签就是脚本中由尖括号包围的关键词，例如：`<h1>这里是标题行</h1>`

- ◆ 标签通常成对出现，`<h1>` 称为开始标签，`</h1>` 称为结束标签
- ◆ 开始标签到结束标签之间所有的脚本称为一个元素，`<h1>这里是标题行</h1>`
- ◆ 文本“`这里是标题行`”是这个元素的内容
- ◆ HTML文档中不同类型的标签，可以展现不同的样式，例如：文档主体标签 `<body>`、标题标签 `<h1>`、段落标签 `<p>`





常见的标签及属性

新建了一个名为
css1 的样式类

```
1 <html>
2 <body bgcolor="#eeeeee">
3
4   <style>
5     .css1 { background-color:yellow; color:green; font-style:italic;}
6   </style>
7   <h1 align="center">这里是标题行</h1>
8   <p name="p1" class="css1">这是第一段</p>
9   <p name="p2" class="css1">这是第二段</p>
10
11   </img>
12   <a id='link' href="http://baidu.com">点我跳去百度</a>
13 </body>
14 </html>
```

标签的各种属性描述了标签的不同特征，例如：

- ◆ name、id 属性对标签进行命名和标识
- ◆ align属性：标签内容的对齐方式
- ◆ class属性：标签采用哪种样式类，例如 css1
- ◆ href 属性：超链接标签<a>的跳转路径





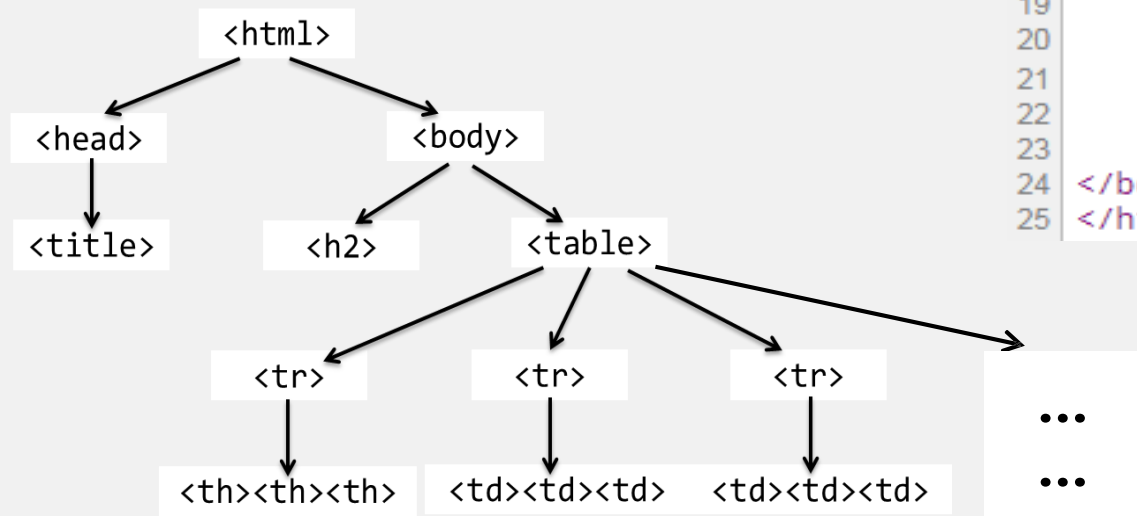
HTML文档的树形结构

网页标题

file:///C:/Page2.html

金庸群侠传

书名	人物	年份
《射雕英雄传》	郭靖	1959年
《倚天屠龙记》	张无忌	1961年
《笑傲江湖》	令狐冲	1967年
《鹿鼎记》	韦小宝	1972年



```
1 <html>
2 <head>
3   <title>网页标题</title>
4 </head>
5 <body>
6   <h2>金庸群侠传</h2>
7   <table width="400px" border="1">
8     <tr>
9       <th>书名</th> <th>人物</th> <th>年份</th>
10    </tr>
11    <tr>
12      <td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td>
13    </tr>
14    <tr>
15      <td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td>
16    </tr>
17    <tr>
18      <td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td>
19    </tr>
20    <tr>
21      <td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td>
22    </tr>
23  </table>
24 </body>
25 </html>
```

标签 `<table>`、`<tr>`、`<td>`
组合起来可展现表格数据

HTML文档中标签的嵌套体现了包含关系
所有标签的嵌套关系可以映射为一个树状结构



作者: 金庸

出版社: 生活·读书·新知三联书店

出版年: 1999-04

页数: 1263

定价: 47.00元

裴平 裴頓

丛书：金庸作品集（三联口袋本）

ISBN: 9787108012586

豆瓣评分

8.9



37628人评价

57.5%

34.8%

返回(B)

Alt+向左箭头

前进(F)

Alt+向右箭头

重新加载(R)

Ctrl+R

另存为(A)...

Ctrl+S

打印(P)...

Ctrl+P

投射(C)...

翻成中文(简体)(T)

[查看网页源代码\(V\)](#)

Ctrl+U

検査(IV)

Ctrl+Shift+I

推荐

内容简介

《射雕英雄传》是金庸的代表作之一，作于一九五七年到一九五九年，在《香港商报》连载。《射雕》中的人物个性单纯，郭靖质朴厚重、黄蓉机智狡黠，读者容易印象深刻。这是中国传统小说和戏剧的特征，但不免缺乏人物内

以 Chrome 浏览器为例，
点击鼠标右键，
弹出快捷菜单

<https://book.douban.com/subject/1789837/>



浏览器的开发者工具

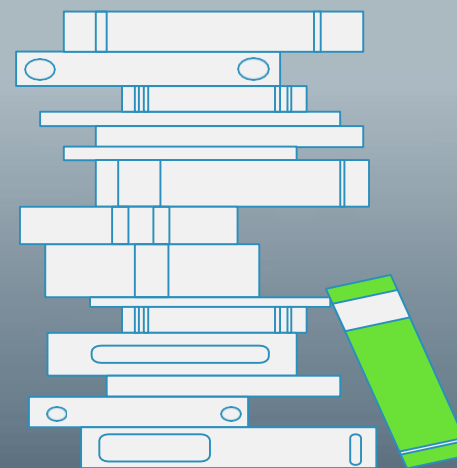
以 Chrome 浏览器为例，选中网页文本，
单点击鼠标右键，在弹出快捷菜单中选择“**检查**”，
或者按下 F12 键，直接调出“**开发者工具**”窗口。

The screenshot shows a web browser window with the address bar displaying `https://book.douban.com/subject/1044547/`. The page content includes a title bar for '射雕英雄传 (全四册) (豆瓣)', a navigation bar with buttons like '想读', '在读', '读过', and a rating section. The main content area shows the book's introduction, which is highlighted in blue. The developer tools are open on the right side, showing the 'Elements' panel with the following HTML structure:

```
<div class="indent" id="link-report">
  <div class=
    <style type="text/css" media="screen">
      .intro p{text-indent:2em;word-break:normal;}
    </style>
    <div class="intro">
      <p>
        "《射雕英雄传》是金庸的代表作之一，作于一九五七年到一九五九年，在《香港商报》
        个性单纯，郭靖诚朴厚重、黄蓉机智狡狴，读者容易印象深刻。这是中国传统小说和戏剧的
        心世界的复杂性。由于人物性格单纯而情节热闹，所以《射雕》比较得到欢迎，被拍成各种
        球众多国家和地区热播。"
      </p>
    </div>
  </div>
  <link rel="stylesheet" type="text/css" href="https://img3.doubanio.com/f/shire/c4c6dd2/_css/report.css">
  <link rel="stylesheet" type="text/css" href="https://img3.doubanio.com/f/
```




Requests 库的使用





Requests库简介

- 正常用户大多通过浏览器来访问网页，由浏览器负责网络请求的发送，以及数据的接收和展示。
- Python语言则提供了多个成熟的程序库，用来实现网络访问，本节我们学习比较容易使用的requests库。
- requests库不是内置库，使用之前需单独安装：
`pip install requests`
- 安装之后，使用时需要单独引入：
`import requests`



HTTP 请求方法

HTTP协议中，对用户访问网络资源，定义了多种请求方法，与之对应，requests库也提供了与各请求方法匹配的请求函数，其中最常用的是GET、POST方法。

序号	HTTP 请求方法	requests 请求函数	功 能 描 述
1	GET	get (url)	请求指定的页面信息，并返回页面主体数据，多用于查看网页。
2	POST	post (url , data)	向指定资源提交数据进行处理请求，例如提交表单或者上传文件。常用于新增数据或修改数据。
3	HEAD	head (url)	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
4	PUT	put (url, data)	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	delete (url)	请求服务器删除指定的页面。
7	OPTIONS	options (url)	允许客户端查看服务器的性能。



GET方式请求网页

get() 函数可以根据给定参数获取网页主体数据，函数原型：

```
response = requests.get(url, [headers,  
params,...])
```

常用参数如下表所示：

参数名称	参数类型	参数功能
url	str	指明所请求的网页路径
headers	dict	GET请求包含的头部数据
params	dict	GET请求所需提交的参数数据（键值对）
timeout	int	超时设定(秒)，超过指定时间未返回就抛错



GET方式请求网页

执行get()函数会返回一个Response类型的对象，如下例中的r、r2，该对象中包含了服务器的所有响应数据。

```
import requests
r = requests.get('http://www.baidu.com')

# 字典 myParam 存储了将要提交到网页的参数，本例中是提交给百度查询的关键字
myParam = {'wd': 'python'}
r2 = requests.get('http://www.baidu.com/s',
params=myParam)
```



Response返回对象

Response对象中包含了服务器的所有响应数据，通过访问该对象的不同属性，可以获取相关数据。Response对象的常用属性如下表：

属性名称	属性描述
status_code	响应状态码，以整数表示，例如：200表示请求成功， 403表示没有访问权限。
text	服务器响应内容的字符串形式
content	服务器响应内容的二进制形式
encoding	服务器响应内容的编码方式，允许更改
headers	服务器响应的头部信息，以字典形式存储



Response对象的text属性以字符串形式保存了网页文档，不同网站的文档可能采用不同的编码格式，如下图百度首页采用了ISO-8859-1编码方式，所以中文显示异常，如何解决？

```
>>> import requests
>>> r = requests.get('http://www.baidu.com')
>>> r.status_code
200
>>> r.encoding
'ISO-8859-1'
>>> r.text
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv=conten
t-type content=text/html; charset=utf-8><meta http-equiv=X-UA-Compatible
content=IE=Edge><meta content=always name=referrer><link rel=stylesheet
type=text/css href=http://s1.bdstatic.com/r/www/cache/bdorz/baidu.min.cs
s><title>ç\x99%å°|ä, \x80ä, \x8bï¼\x8cä½\xa0å°±ç\x9f¥é\x81\x93</title></he
ad> <body link=#0000cc> <div id=wrapper> <div id=head> <div class=head_w
```



我们可以重新设定Response对象的编码格式，采用python默认支持的utf-8编码方式，这样text属性中的中文字符就能够正常展示。

```
>>> r.encoding = 'utf-8'
>>> r.text
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv=conten
t-type content=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible
content=IE=Edge><meta content=always name=referrer><link rel=stylesheet
type=text/css href=http://s1.bdstatic.com/r/www/cache/bdorz/baidu.min.cs
s><title>百度一下，你就知道</title></head> <body link=#0000cc> <div id=wr
apper> <div id=head> <div class=head_wrapper> <div class=s_form> <div cl
```




Response对象的text和content属性都存储了服务器的响应内容，两者区别在于：

- text属性以字符串形式存储响应内容，常用于分析处理html文档，获取文档中数据。
- content属性以二进制（字节）形式存储响应内容，常用于存储图片、视频及其他非文本格式的文件。

```
import requests
imgUrl = 'http://www.baidu.com/img/bd_logo1.png'
r = requests.get(imgUrl)

path = r'D:\baidu.png'
with open(path, 'wb') as file:
    file.write(r.content)
```





Response对象 - raise_for_status()方法

案例 12-5

在连续爬取多个网页的数据时，如果其中个别网页爬取失败，程序会抛出异常，导致运行终止。如何跳过这种偶发性的错误，继续跳往下一个网页进行爬取，Response对象提供了raise_for_status()方法。以豆瓣网《天龙八部》的书评为例，我们尝试采集前10页的评论。

https://book.douban.com/subject/1255625/comments/

天龙八部 短评

全部共 11455 条

热门 / 最新 / 好友

> 我来写短评

> 天龙八部



恶魔奶爸Sam ★★★★★ 2014-01-04

1315 有用

经典武学起于春秋，到北宋达到极盛，南宋有所衰落，元末因明教兴盛中兴。明代更有东方不败等大师。但期间伽利略的出现已预示其衰落。康熙年间出现牛顿，梯云纵灭绝。道光年间出现卡诺，玄冰掌灭绝。尤其是力学定理建立后，内力和轻功全部消失，梯云纵这种只需吐纳练便可自带反重力系统的轻功彻底消失



郭发财 ★★★★★ 2013-03-24

779 有用

因为众生皆苦，便觉得自己也不是很苦。



豪气士财主 ★★★★★ 2016-06-06

769 有用

慕容复是没落皇族子孙，毕生追求皇位，对王语嫣视如草芥，段誉是正牌皇帝继承人，却爱美人宁弃江山。你所追的，正是我所弃的，尽道人生的可笑荒谬。



作者: 金庸

isbn: 7108006723

书名: 天龙八部

页数: 1978

译者: 有1996年11月北

定价: 96.0

https://book.douban.com/subject/1255625/comments/

https://book.douban.com/subject/1255625/comments/hot?p=2

https://book.douban.com/subject/1255625/comments/hot?p=10

通过浏览器进行翻页，分析评论页URL的变化可以看出：URL主体部分不变，最后的参数 p 每次加一，因此可以构建一个循环来生成URL。

https://book.douban.com/subject/1255625/comments/?limit=20&status=P&sort=new_score



Response对象 - raise_for_status()方法

案例 12-5

基本思路，通过循环生成前10页的URL，获取每页评论写入文件，如遇异常则打印显示。

```
import time
import requests

url = 'https://book.douban.com/subject/1255625/comments/hot?p='
for i in range(1, 11):
    try:
        r = requests.get(url + str(i))
        r.raise_for_status()
        r.encoding = 'utf-8'

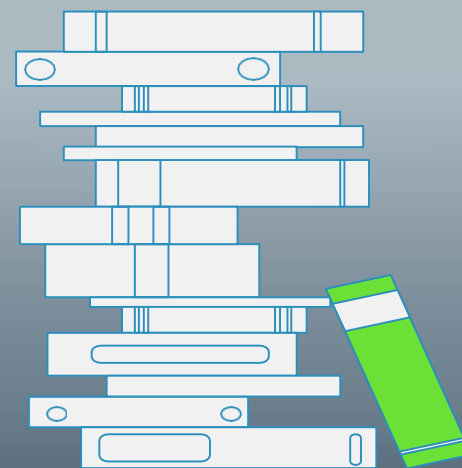
        path = 'C:\\\\评论第{}页.html'.format(i)
        with open(path, 'w', encoding='utf-8') as file:
            file.write(r.text)
        time.sleep(3)  # 抓取一页评论数据后，休眠3秒再抓取下一页

    except Exception as ex:
        print("第{}页采集出错，出错原因:{}".format(i, ex))
```

该方法主要用于当请求某个网页出现异常时，允许用户捕捉并记录该异常，转而执行之后的任务。



BeautifulSoup库的使用





BeautifulSoup库简介

- BeautifulSoup（美味汤），是一个用Python写的优秀的第三方库，主要用来解析和处理 HTML/XML 格式的数据。

- 安装方法：

```
pip install beautifulsoup4 / bs4
```

- 导入方法：

```
import bs4
```

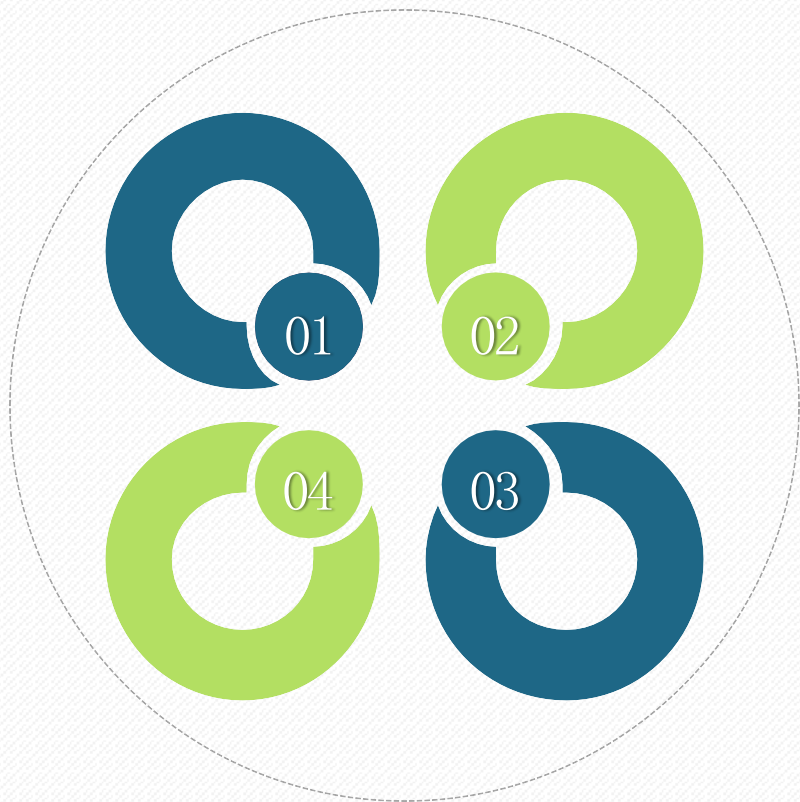
```
from bs4 import BeautifulSoup
```

- 官方文档（中英文）：

<https://www.crummy.com>



BeautifulSoup库简介



- 1 BeautifulSoup 解析器
- 2 BeautifulSoup 四类对象
- 3 遍历文档树
- 4 搜索文档树



为了从HTML文档中解析目标数据，首先需根据文档创建一个BeautifulSoup对象，此时需选择一种解析器类型，基本用法如下：

```
import requests
import bs4

r =
requests.get('http://www.baidu.com')
r.encoding = 'utf-8'

# 创建BeautifulSoup对象:soup
soup = bs4.BeautifulSoup(r.text, 'html.parser')
```

HTML文档字符串

解析器类型



BeautifulSoup 解析器

BeautifulSoup支持python标准库的HTML解析器，还支持一些第三方解析器(需安装)，常用的几种解析器如下表所示。

解析器	参数写法	优点	缺点
Python 标准库	html.parser	<ul style="list-style-type: none">Python内置标准库解析速度适中HTML文档容错性强	在旧版本Python中容错性不好
lxml html 解析器	lxml	<ul style="list-style-type: none">解析速度快解析HTML文档容错性好	需安装C语言库
lxml XML 解析器	xml	<ul style="list-style-type: none">解析速度快唯一支持XML的解析器	需安装C语言库
html5lib	html5lib	<ul style="list-style-type: none">容错性最好以浏览器方式解析文档可以生成HTML5格式的文档	解析速度慢



BeautifulSoup的四种对象

BeautifulSoup 能够将复杂的HTML文档映射为一个树形结构，文档中的各种节点则被转换为各种对象。主要包含以下四类：

- Tag 对象
- BeautifulSoup 对象
- NavigableString 对象
- Comment 对象



Tag 对象 -> 标签对象

Tag 对象对应了HTML文档中的各种标签，在创建BeautifulSoup对象后，文档中的各种标签会自动转换为Tag对象存放在树形结构的相应位置。

Tag对象常用的属性：

tag.name : 标签类型

tag.attrs : 标签属性字典

tag['属性名']: 标签单个属性



Tag 对象 -> 标签对象

Tag对象示例程序

案例 12-7

HTML文档存入变量code

根据code新建BeautifulSoup对象

```
soup = bs4.BeautifulSoup(code, 'html.parser')
```

```
print(soup.p)      # 输出标签对象: <p class="css1" name="p1">这是第一段</p>
```

```
print(type(soup.p)) # 输出对象类型: <class 'bs4.element.Tag'>
```

```
print(soup.p.name)  # 输出标签类型: 'p'
```

```
print(soup.p.attrs) # 输出标签属性字典: {'name': 'p1', 'class': ['css1']}
```

```
print(soup.p['name']) # 输出标签name属性: p1
```

```
print(soup.p['class']) # 输出标签class属性: ['css1']
```

为什么是个列表

```
<p name="p1" class="css1">这是第一段</p>
```

```
1 <html>
2 <body bgcolor="#eeeeee">
3     <style>
4         .css1 { background-color:yellow; color:green; font-style:italic;}
5     </style>
6     <h1 align="center">这里是标题行</h1>
7     <p name="p1" class="css1">这是第一段</p>
8     <p name="p2" class="css1">这是第二段</p>
9
10    </img>
11    <a id='link' href="http://baidu.com">点我跳去百度</a>
12 </body>
13 </html>
```



标签的class（样式类）是个多值属性，一个标签上会应用多个样式，此时class就会被赋予多个值。

```
html = '<p class="css1 css2"></p>'
soup2 = bs4.BeautifulSoup(html, 'html.parser')
print(soup2.p['class'])          # 输出 ['css1', 'css2']
```



BeautifulSoup对象 -> HTML文档

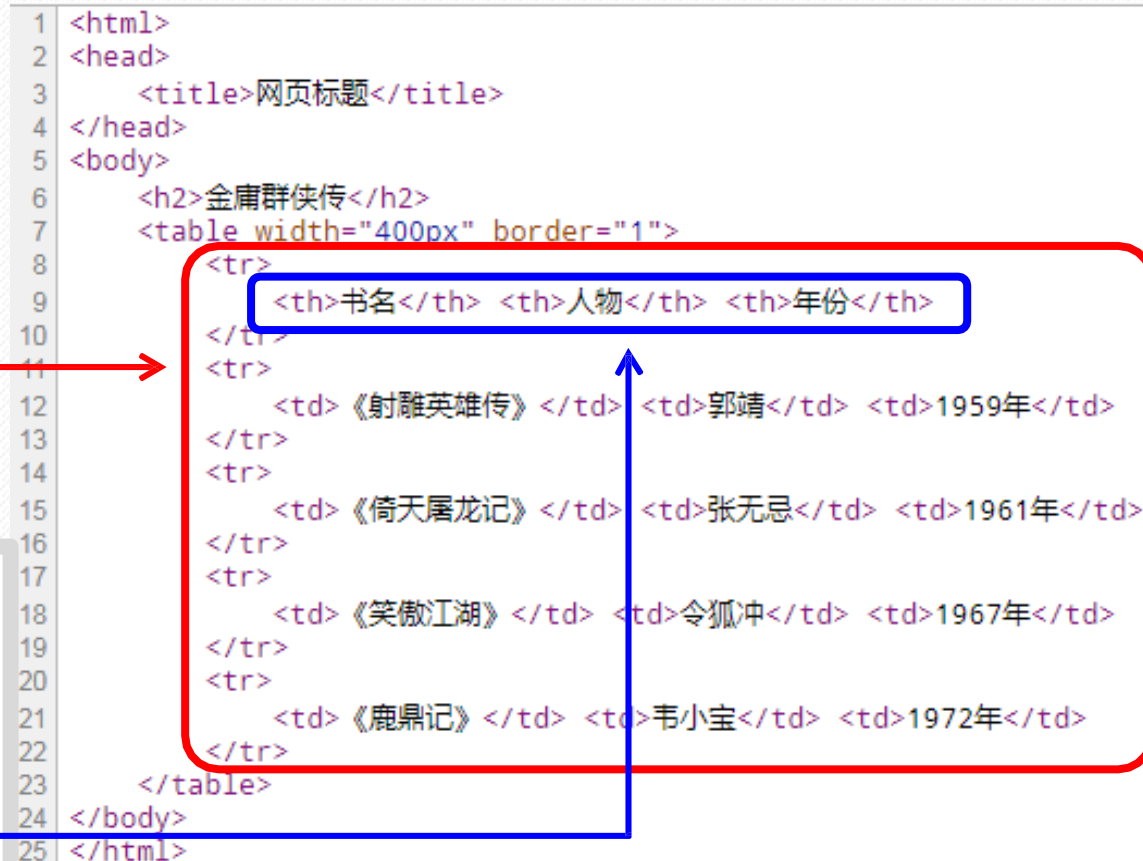
BeautifulSoup对象对应整个HTML文档，可以看做HTML文档树的根，作为一个顶层节点，文档中所有标签及内容都是它的后代节点。

大多数时候，从BeautifulSoup对象开始向下搜索或者遍历文档树。



书名	人物	年份
《射雕英雄传》	郭靖	1959年
《倚天屠龙记》	张无忌	1961年
《笑傲江湖》	令狐冲	1967年
《鹿鼎记》	韦小宝	1972年

```
1 <html>
2 <head>
3   <title>网页标题</title>
4 </head>
5 <body>
6   <h2>金庸群侠传</h2>
7   <table width="400px" border="1">
8     <tr>
9       <th>书名</th> <th>人物</th> <th>年份</th>
10    </tr>
11    <tr>
12      <td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td>
13    </tr>
14    <tr>
15      <td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td>
16    </tr>
17    <tr>
18      <td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td>
19    </tr>
20    <tr>
21      <td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td>
22    </tr>
23  </table>
24 </body>
25 </html>
```



```
soup = bs4.BeautifulSoup(html, 'html.parser')
print(soup.table.contents) # 输出表格子节点
print(soup.find_all('th')) # 输出表格标题行
```



NavigableString对象 -> 可遍历的字符串

NavigableString称为可以遍历的字符串对象，用来操作那些被包含在标签内的字符串，实际上就是那些我们想要的的数据。

```
soup = bs4.BeautifulSoup(html, 'html.parser')
print(soup.h1.string)           # 输出h1标签包含的字符串: '这里是标题行'
print(soup.p.string)           # 输出p标签包含的字符串: '这是第一段'
print(type(soup.p.string))     # <class 'bs4.element.NavigableString'>
```

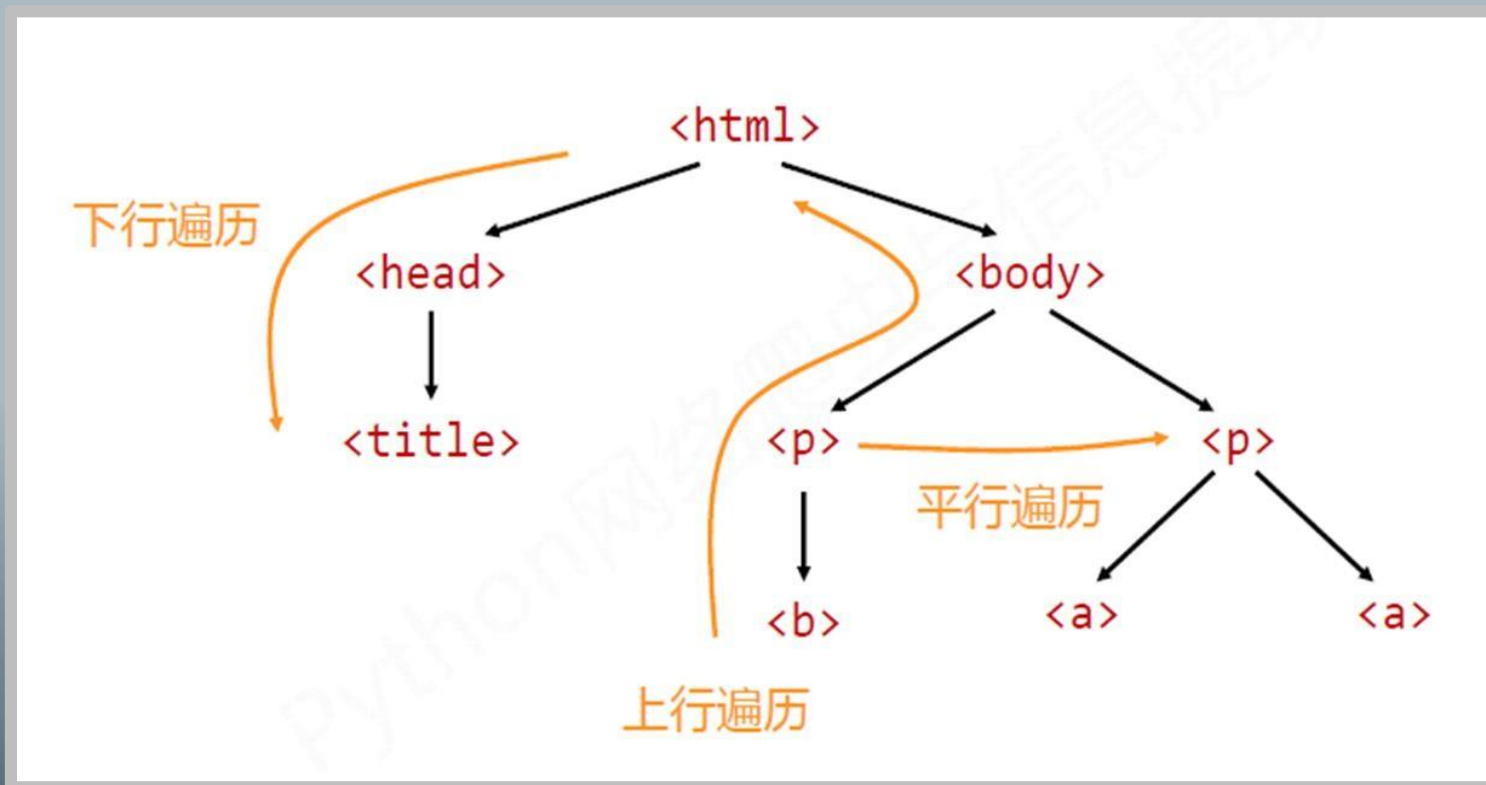
← → ↻ ⓘ view-source:file:///C://page1.html

```
1 <html>
2 <body bgcolor="#eeeeee">
3     <style>
4         .css1 { background-color:yellow; color:green; font-style:italic;}
5     </style>
6     <h1 align="center">这里是标题行</h1>
7     <p name="p1" class="css1">这是第一段</p>
8     <p name="p2" class="css1">这是第二段</p>
9
10    </img>
11    <a id='link' href="http://baidu.com">点我跳去百度</a>
12 </body>
13 </html>
```

遍历文档树

HTML文档可以映射为一棵树，各种嵌套标签对应树的各层节点，我们就可以对树进行遍历访问。遍历时会出现三种路径：

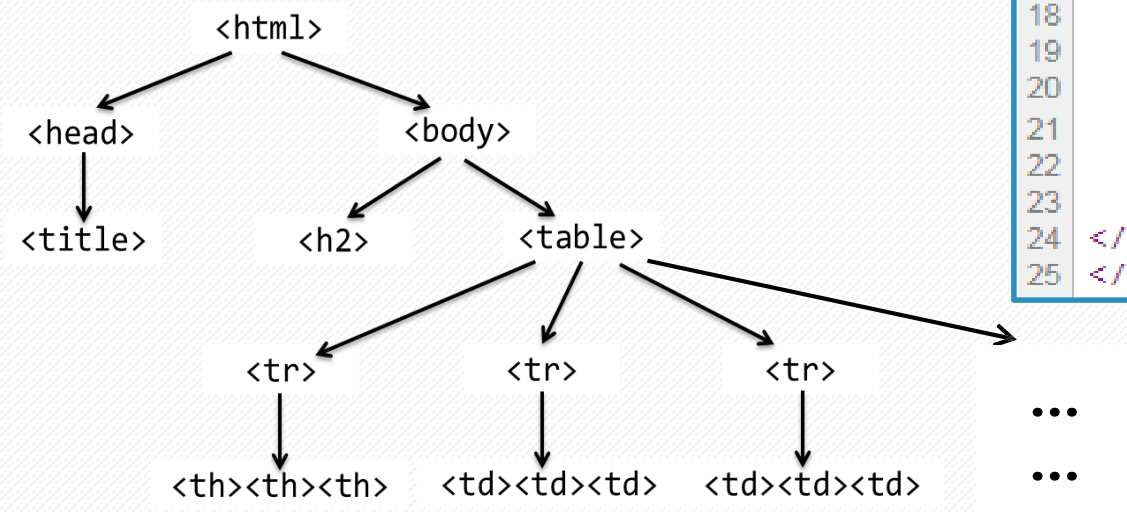
- 自上而下遍历
- 水平方向遍历
- 自下而上的遍历





遍历文档树

金庸群侠传		
书名	人物	年份
《射雕英雄传》	郭靖	1959年
《倚天屠龙记》	张无忌	1961年
《笑傲江湖》	令狐冲	1967年
《鹿鼎记》	韦小宝	1972年



```
1 <html>
2 <head>
3   <title>网页标题</title>
4 </head>
5 <body>
6   <h2>金庸群侠传</h2>
7   <table width="400px" border="1">
8     <tr>
9       <th>书名</th> <th>人物</th> <th>年份</th>
10    </tr>
11    <tr>
12      <td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td>
13    </tr>
14    <tr>
15      <td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td>
16    </tr>
17    <tr>
18      <td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td>
19    </tr>
20    <tr>
21      <td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td>
22    </tr>
23  </table>
24 </body>
25 </html>
```

注意标签的上下级关系



自上而下遍历的属性：

子节点 : children, `<tr>` 是 `<table>` 标签的子节点

子节点 : contents, `<td>` 是 `<tr>` 标签的子节点

后代节点: descendants, `<tr>`、`<td>` 都是 `<table>` 标签的后代节点

属性 contents 和 children 的区别：

contents 属性返回子节点组成的列表，可以采用列表方式访问子节点标签；

children 属性返回的是子节点所构成的生成器，还需要经过循环才能取出子节点。



遍历文档树 -> 下行遍历

案例 12-8

```
from bs4 import BeautifulSoup, element
```

```
code = '''<html><head>
<title>网页标题</title></head>
<body>
  <h2>金庸群侠传</h2>
  <table width="400px" border="1">
    <tr><th>书名</th> <th>人物</th> <th>年份</th></tr>
    <tr><td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td></tr>
    <tr><td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td></tr>
    <tr><td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td></tr>
    <tr><td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td></tr>
  </table>
</body></html>'''
```

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

```
print(soup.table.contents)
```

```
['\n', <tr><th>书名</th> <th>人物</th> <th>年份</th></tr>, '\n', <tr>
<td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td></tr>, '\n', <tr>
<td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td></tr>, '\n', <t
r><td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td></tr>, '\n', <t
r><td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td></tr>, '\n']
```



遍历文档树 -> 下行遍历

案例 12-9

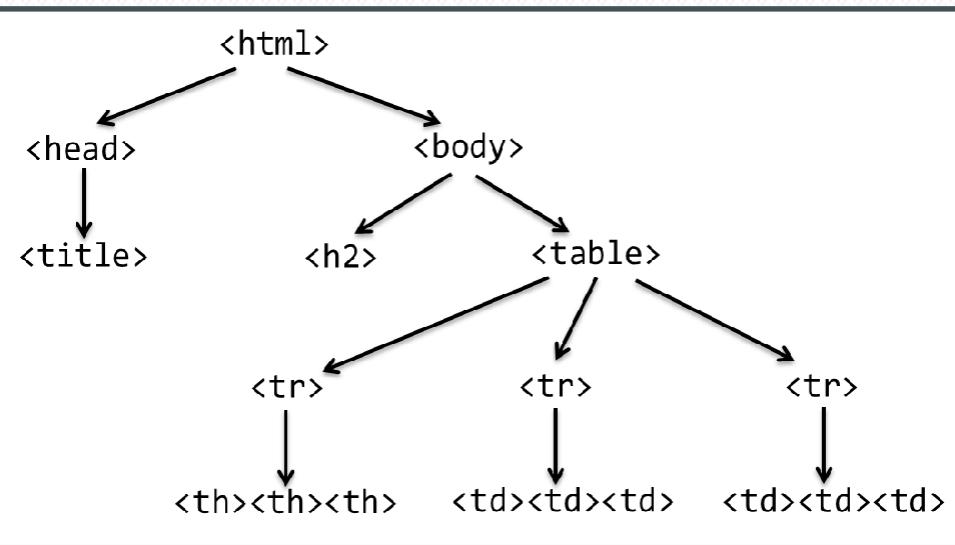
对children属性做循环，输出<table>标签的子标签

```
for child in soup.table.children:
```

过滤标签之间的换行符

```
if type(child) != element.NavigableString:
```

```
    print(child)
```



```
<tr><th>书名</th> <th>人物</th> <th>年份</th></tr>
<tr><td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td></tr>
<tr><td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td></tr>
<tr><td>《笑傲江湖》</td> <td>令狐冲</td> <td>1967年</td></tr>
<tr><td>《鹿鼎记》</td> <td>韦小宝</td> <td>1972年</td></tr>
```

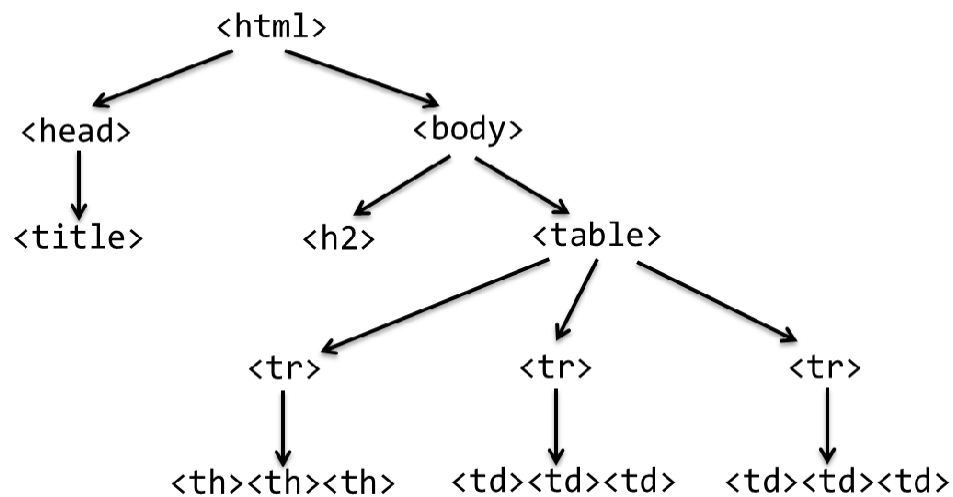


遍历文档树 -> 下行遍历

案例 12-9

```
# 对descendants属性做循环，输出<table>标签的后代标签
for des in soup.table.descendants:
    if type(des) != element.NavigableString:
        print(des)
```

```
<tr><th>书名</th> <th>人物</th> <th>年份</th></tr>
<th>书名</th>
<th>人物</th>
<th>年份</th>
<tr><td>《射雕英雄传》</td> <td>郭靖</td> <td>1959年</td></tr>
<td>《射雕英雄传》</td>
<td>郭靖</td>
<td>1959年</td>
<tr><td>《倚天屠龙记》</td> <td>张无忌</td> <td>1961年</td></tr>
<td>《倚天屠龙记》</td>
<td>张无忌</td>
<td>1961年</td>
```





水平方向解析

兄弟节点, next_siblings、previous_sibling 属性。

```
for child in soup.table.tr.next_siblings:      # 获取第一行向后的兄弟标签
    if type(child) != element.NavigableString: # 过滤标签之间的换行
        print(child)
```

```
<tr>
<td>《射雕英雄传》</td><td>郭靖</td><td>1959年</td>
</tr>
<tr>
<td>《倚天屠龙记》</td><td>张无忌</td><td>1961年</td>
</tr>
<tr>
<td>《笑傲江湖》</td><td>令狐冲</td><td>1967年</td>
</tr>
<tr>
<td>《鹿鼎记》</td><td>韦小宝</td><td>1972年</td>
</tr>
```

没有标题行!!!

第一行是标题行, 标题行向后的兄弟标签不包含自身。

搜索文档树

现实中，复杂网页的文档树上下级关系可能多达几十层，数据经常藏在若干层标签之下，此时一级级的去找数据效率很低。

BeautifulSoup库提供了另一种**文档树搜索**的方法，根据目标数据所在的标签区域特征，定位到目标标签附近，再利用局部的上下级节点关系，对局部区域进行标签遍历，获取目标数据。

这就类似于我们在电商平台购物，通过商品类别一级一级寻找商品，不如直接搜索关键字。



搜索文档树

核心方法：

`find (name , attrs , recursive , text , **kwargs)` # 返回第一匹配项

`find_all (name, attrs, recursive, text, limit, **kwargs)` # 返回所有匹配项的列表

函数的参数充当过滤器，对标签进行筛选匹配，往往是我们先分析文档，找到目标标签的特征，再设置对应参数的值。

参数名	参数功能	说明
name	标签类型	通过标签类型进行匹配
**kwargs	键值参数	通过特定属性值进行匹配
attrs	属性参数	通过传入字典进行多属性匹配
recursive	是否搜索后代节点	设为False时只查找子节点
text	文档内容匹配	基于标签中字符串的匹配
limit	返回结果的数量限制	



name 参数：

查找类型为name值的标签。

```
← → ↻ ⓘ view-source:file:///C://page1.html
1 <html>
2 <body bgcolor="#eeeeee">
3   <style>
4     .css1 { background-color:yellow; color:green; font-style:italic;}
5   </style>
6   <h1 align="center">这里是标题行</h1>
7   <p name="p1" class="css1">这是第一段</p>
8   <p name="p2" class="css1">这是第二段</p>
9
10  </img>
11  <a id='link' href="http://baidu.com">点我跳去百度</a>
12 </body>
13 </html>
```

```
>>> soup.find_all('h1')
[<h1 align="center">这里是标题行</h1>]
```

```
>>> soup.find_all('h2')
[]
```

```
>>> soup.find_all('p')
[<p class="css1" name="p1">这是第一段</p>, <p class="css1" name="p2">这是第二段</p>]
```

```
>>> soup.find_all(['p', 'a'])
[<p class="css1" name="p1">这是第一段</p>, <p class="css1" name="p2">这是第二段</p>, <a href="http://baidu.com" id="link">点我跳去百度</a>]
```




**kwargs参数:

键值 (keyword) 参数,
直接设定标签的属性名称和值
进行查找。

view-source:file:///C://page1.html

```
1 <html>
2 <body bgcolor="#eeeeee">
3     <style>
4         .css1 { background-color:yellow; color:green; font-style:italic;}
5     </style>
6     <h1 align="center">这里是标题行</h1>
7     <p name="p1" class="css1">这是第一段</p>
8     <p name="p2" class="css1">这是第二段</p>
9
10    </img>
11    <a id='link' href="http://baidu.com">点我跳去百度</a>
12 </body>
13 </html>
```

比较常见的情况是, 如果目标标签具备 **id属性**, 我们会优先采用id属性进行标签搜索, 因为标签的id属性值在文档中大多是唯一的。

```
>>> soup.find_all(id='link')
[<a href="http://baidu.com" id="link">点我跳去百度</a>]
```

```
>>> soup.find_all('h1', align="center")
[<h1 align="center">这里是标题行</h1>]
```



**kwargs参数:

注意一下标签的class（样式类）属性，因为class本身就是Python的关键字，所以采用class属性作为keyword参数搜索时，需要加一个下划线写成 **class_**。

```
>>> soup2 = BeautifulSoup('<p class="css1 css2"></p>' , 'html.parser')
```

```
>>> soup2.find_all(class_='css1')
```

```
>>> soup2.find_all(class_='css2')
```

```
>>> soup2.find_all(class_='css1 css2')
```

以上三条语句都输出:

```
[<p class="css1" name="p1">这是第一段</p>, <p class="css1" name="p2">这是第二段</p>]
```



attrs 参数:

attrs参数定义了一个字典参数来实现对多个标签属性的查找匹配。

```
<  >  ↻  ⓘ  view-source:file:///C://page1.html
1  <html>
2  <body bgcolor="#eeeeee">
3      <style>
4          .css1 { background-color:yellow; color:green; font-style:italic;}
5      </style>
6      <h1 align="center">这里是标题行</h1>
7      <p name="p1" class="css1">这是第一段</p>
8      <p name="p2" class="css1">这是第二段</p>
9
10     </img>
11     <a id='link' href="http://baidu.com">点我跳去百度</a>
12 </body>
13 </html>
```

```
>>> soup.find_all(attrs={'class':'css1'})
[<p class="css1" name="p1">这是第一段</p>, <p align="center" class="css1"
name="p2">这是第二段</p>]
```

```
>>> soup.find_all('p', {'name':'p2','class':'css1'})
[<p align="center" class="css1" name="p2">这是第二段</p>]
```



text 参数:

text 参数是针对文本内容的查找，通过它可以搜索文档中的字符串内容。

直接对text参数赋值进行查找，需要标签内的字符串与参数值完全一样才算作匹配。所以大多时候，对于文本字符串的搜索大多是基于关键字的模糊查询，此时需要用到正则表达式知识，需要导入正则库re。

```
>>> soup.find_all(text='这是第一段')
['这是第一段']
>>> soup.find_all(text='是')
[]
>>> import re
>>> soup.find_all(text=re.compile('是'))
['这里是标题行', '这是第一段', '这是第二段']
```



recursive 参数:

函数默认是查找当前对象的所有后代节点，如果只想搜索当前标签对象的子节点，无须再查找子节点的后代，对recursive参数赋值为False即可。

limit 参数:

find_all()方法默认返回对当前对象搜索的全部匹配结果，如果后代标签很多，搜索会很慢。如果不需要全部结果时，可以使用limit参数限制返回结果的数量，只返回限定数量的匹配结果。

综合实例：爬取豆瓣《天龙八部》书评

以豆瓣网上三联书店1994年版本的《天龙八部》短评为例，抓取短评网页的源码，解析HTML文档获取其中实际有效的评论内容，并以此生成一个词云图片。

为了让程序结构更清晰，我们对代码进行拆分，将抓取网页源码、分析网页源码抽取评论内容、生成词云图片文件三部分功能分别定义成独立函数，由主程序进行调用。





综合实例 – 豆瓣书评

书评页面的标签结构

★★★★★ 2007-12-06 有缘皆孽	<pre>▼<div class="article"> ▼<div class="comments-wrapper"> ▶<div class="nav-tab title_line clearfix">...</div> ▼<div id="comment-list-wrapper"> ▼<div id="comments" class="comment-list hot "> ▼<div id="comments" class="comment-list hot "> ▼ ▶<li class="comment-item" data-cid="34917952">... ▶<li class="comment-item" data-cid="25529667">... ▶<li class="comment-item" data-cid="936571468">... ▶<li class="comment-item" data-cid="75776866">... ▼<li class="comment-item" data-cid="36188244"> ::before ▶<div class="avatar">...</div> ▼<div class="comment"> ▶<h3>...</h3> ▼<p class="comment-content"> ... 文学中英雄形象的创造，表明一个民族对某种理想的追求， 对某种价值的向往。比如乔峰。 == \$0 </p> ▶<div class="comment-report" style="visibility: hidden;">...</div> </div> ▶<li class="comment-item" data-cid="279352415">...</pre>
★★★★★ 2015-07-01 回目成词，金老爷子的天龙八部真是把武侠小说写绝了	
★★★★★ 2008-11-28 想当年看到人落泪	
2008-04-06 文学中英雄形象的创造，表明一个民族对某种理想的追求，对某种价值的向往。比如	



综合实例 – 豆瓣书评

重点标签:

- <div>
-
- <p>
-

```
▼<div class="comments-wrapper">
  ▶<div class="nav-tab title_line clearfix">...</div>
  ▼<div id="comment-list-wrapper">
    ▼<div id="comments" class="comment-list hot ">
      ▼<div id="comments" class="comment-list hot ">
        ▼<ul>
          ▶<li class="comment-item" data-cid="34917952">...</li>
          ▶<li class="comment-item" data-cid="25529667">...</li>
          ▶<li class="comment-item" data-cid="936571468">...</li>
          ▶<li class="comment-item" data-cid="75776866">...</li>
          ▼<li class="comment-item" data-cid="36188244">
            ::before
            ▶<div class="avatar">...</div>
            ▼<div class="comment">
              ▶<h3>...</h3>
              ▼<p class="comment-content">
                ▼<span class="short">文学中英雄形象的创造，表明一个民族对某种理想的追求，
                  对某种价值的向往。比如乔峰。</span> == $0
                </p>
                ▶<div class="comment-report" style="visibility: hidden;">...</div>
              </div>
            </li>
            ▶<li class="comment-item" data-cid="279352415">...</li>
```




综合实例 – 1. 爬取网页

```
import time, requests, jieba
from bs4 import BeautifulSoup
from wordcloud import WordCloud

def getHtmlDoc(url, page):
    '''# 函数1: 爬取给定url的html文档'''
    try:
        url = url + page
        r = requests.get(url, timeout=30)
        r.raise_for_status()
        r.encoding = 'utf-8'
        return r.text
    except Exception as ex:
        print("第{}页采集出错, 出错原因: {}".format(page, ex))
        return ""
```



综合实例 – 2. 解析网页抽取评论

```
def getComment(html):  
    '''# 函数2: 获取给定html文档中的评论内容, 返回评论列表'''  
    comment = []          # 该列表用于存储当前页面的所有评论  
    soup = BeautifulSoup(html, 'html.parser')  
    div = soup.find('div', id='comments')  
  
    # 获取<div>标签内部的所有列表项标签<li>, 再获取后代标签<p>、<span>  
    for li in div.find_all('li', {'class': 'comment-item'}):  
        p = li.find('p', {'class': 'comment-content'})  
        text = p.span.string  
        comment.append(text)  
    return comment
```



综合实例 – 3. 根据文件生成词云图片

```
def createWordCloud(fileName):  
    '''# 函数3: 根据给定评论文件, 利用jieba库分词后生成词云文件'''  
    with open(fileName, 'r', encoding='utf-8') as file:  
        text = file.read()  
        ls_word = jieba.lcut(text)          # 利用jieba库对所有评论进行分词  
        all_words = ','.join(ls_word)      # 所有词语以逗号连接成一个长字符串  
        wcloud = WordCloud(font_path = r'C:\Windows\fonts\simhei.ttf',  
                             width = 1000, height = 800,  
                             background_color = 'white',  
                             max_words = 200,  
                             margin = 2)  
        wcloud.generate(all_words)  
        # 生成词云图片文件, 主文件名同文本文件名  
        fileCloud = fileName.split('.')[0] + '.png'  
        wcloud.to_file(fileCloud)
```

综合实例 – 4. 主程序

```
# 以下为主程序
url = 'https://book.douban.com/subject/1255625/comments/hot?p='
all_comment = []          # 存储全部评论的列表

for p in range(1, 201):
    html = getHtmlDoc(url, str(p))      # 循环爬取前200页html文档
    page_comment = getComment(html)     # 从html文档中抽取评论内容
    all_comment.extend(page_comment)     # 每页的评论列表添加到总列表中
    time.sleep(2)                       # 每爬取一页暂停2秒
    print('第{}页处理完成。'.format(p))

print('网页采集结束，开始写入文件、生成词云。')

# 评论列表全部写入文件
fileName = 'd:\\\\天龙八部评论.txt'
with open(fileName, 'w', encoding='utf-8') as file:
    file.writelines(all_comment)

# 根据评论文件生成词云
createWordCloud(fileName)
print('词云生成结束。')
```



综合实例 – 运行结果

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

第187页处理完成。
第188页处理完成。
第189页处理完成。
第190页处理完成。
第191页处理完成。
第192页处理完成。
第193页处理完成。
第194页处理完成。
第195页处理完成。
第196页处理完成。
第197页处理完成。
第198页处理完成。
第199页处理完成。
第200页处理完成。
网页采集结束，开始
Building prefix
Loading model fr
Loading model co
Prefix dict has
词云生成结束。
>>>

天龙八部评论.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V)

经典武学起于春秋，到北宋方不败等大师。但期间伽叶纳练便可自带反重力系追是看外归天霸峰之作，萧峰这汉子，萧龙八摩一世，我是智和段正淳，下有情人都失散多年，让他照顾阿紫，也是没有她陪伴的日子。阿紫虽不讨喜，可她却都爹给坑。关于此书教化的世界里，佛教的贪婪、放纵、残暴。玄奘自己比较惨，接触了……没看完。五册内就是老直男癌意淫的江湖道义儿女情长。不如看金庸一本正经地胡说八道。唯一的一次儿偶就是一天一夜一口气看完。武侠小说第一。天龙八部在武功、思想深度（佛家哲学性）、人文情感和民族历史中的集大成者，我个人感觉是金庸最佳，其次是笑傲和鹿鼎记并列第二。鹿鼎