



《Python程序设计基础教程（微课版）》

<http://dbl-lab.xmu.edu.cn/post/python>



第14章 网络爬虫

林子雨 博士/副教授

厦门大学计算机科学与技术系

E-mail: ziyulin@xmu.edu.cn ▶▶

主页: <http://dbl-lab.xmu.edu.cn/linziyu>





主讲教师



2017年度厦门大学奖教金获得者

2020年度厦门大学奖教金获得者

主讲教师：厦门大学 林子雨 博士/副教授

中国高校首个“数字教师”提出者和建设者

2009年7月从事教师职业以来

累计**免费**网络发布超过**1500万**字高价值教学和科研资料

网络浏览量超过**1500万**次



提纲

- 14.1 网络爬虫概述
- 14.2 网页基础知识
- 14.3 用Python实现HTTP请求
- 14.4 定制requests
- 14.5 解析网页
- 14.6 综合实例

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著, 人民邮电出版社
《Python程序设计基础教程（微课版）》教材官方网站：
<http://dblalab.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python 程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著

-  **名师精品**
多年计算机教学实践的厚积薄发
-  **深入浅出**
清晰呈现 Python 语言学习路径
-  **实例丰富**
有效提升编程语言的学习趣味
-  **资源全面**
构建全方位一站式在线服务体系

中国工信出版集团 人民邮电出版社
POSTS & TELECOM PRESS



14.1 网络爬虫概述

14.1.1 什么是网络爬虫

14.1.2 网络爬虫的类型

14.1.3 反爬机制



14.1.1 什么是网络爬虫

网络爬虫是一个自动提取网页的程序，它为搜索引擎从万维网上下载网页，是搜索引擎的重要组成部分。如图14-1所示，爬虫从一个或若干个初始网页的URL开始，获得初始网页上的URL，在抓取网页的过程中，不断从当前页面上抽取新的URL放入队列，直到满足系统的一定停止条件。

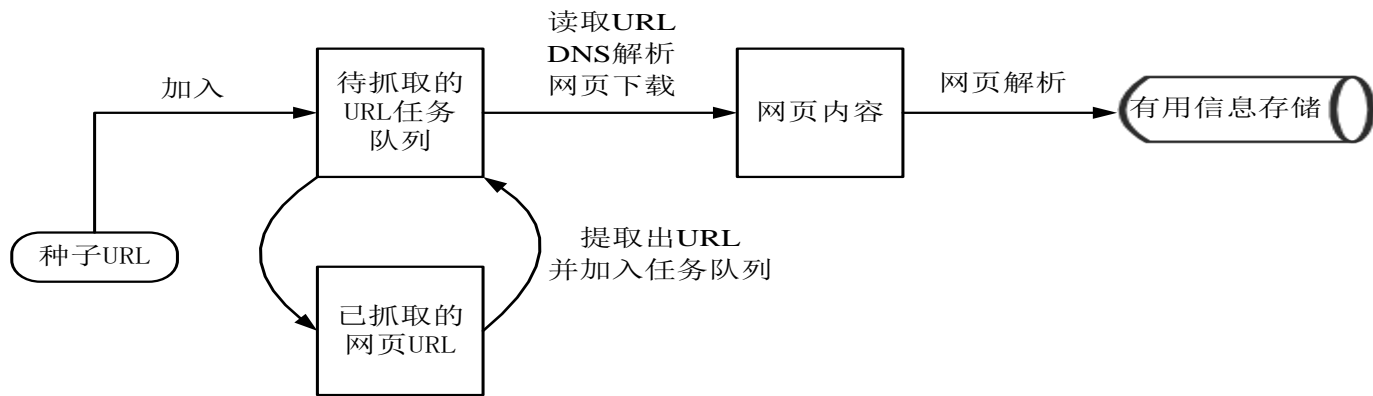


图14-1 网络爬虫的工作原理



14.1.2 网络爬虫的类型

网络爬虫的类型可以分为：

- 通用网络爬虫
- 聚焦网络爬虫
- 增量式网络爬虫
- 深层网络爬虫



14.1.3反爬机制

为什么会有反爬机制？原因主要有两点：

- 第一，在大数据时代，数据是十分宝贵的财富，很多企业不愿意让自己的数据被别人免费获取，因此，很多企业都为自己的网站运用了反爬机制，防止网页上的数据被爬走；
- 第二，简单低级的网络爬虫，数据采集速度快，伪装度低，如果没有反爬机制，它们可以很快地抓取大量数据，甚至因为请求过多，造成网站服务器不能正常工作，影响了企业的业务开展。



14.1.3反爬机制

- 反爬机制也是一把双刃剑，一方面可以保护企业网站和网站数据，但是，另一方面，如果反爬机制过于严格，可能会误伤到真正的用户请求，也就是真正用户的请求被错误当成网络爬虫而被拒绝访问。如果既要和“网络爬虫”死磕，又要保证很低的误伤率，那么又会增加网站研发的成本。
- 通常而言，伪装度高的网路爬虫，速度慢，对服务器造成的负担也相对较小。所以，网站反爬的重点也是针对那种简单粗暴的数据采集。有时反爬机制也会允许伪装度高的网络爬虫获得数据，毕竟伪装度很高的数据采集与真实用户请求没有太大差别。



14.2 网页基础知识

14.2.1 超文本和HTML

14.2.2 HTTP

本PPT是如下教材的配套讲义：

《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《Python程序设计基础教程（微课版）》教材官方网站：

<http://dbl原因.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python

程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著



名师精品

多年计算机教学实践的厚积薄发



深入浅出

清晰呈现 Python 语言学习路径



实例丰富

有效提升编程语言的学习趣味



资源全面

构建全方位一站式在线服务体系



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS



14.2.1 超文本和HTML

超文本（**Hypertext**）是指使用超链接的方法，把文字和图片信息相互联结，形成具有相关信息的体系。超文本的格式有很多，目前最常使用的是超文本标记语言**HTML**（**Hyper Text Markup Language**），我们平时在浏览器里面看到的网页就是由**HTML**解析而成的。下面是网页文件 `web_demo.html` 的**HTML**源代码：

```
<html>
<head><title>搜索指数</title></head>
<body>
<table>
<tr><td>排名</td><td>关键词</td><td>搜索指数</td></tr>
<tr><td>1</td><td>大数据</td><td>187767</td></tr>
<tr><td>2</td><td>云计算</td><td>178856</td></tr>
<tr><td>3</td><td>物联网</td><td>122376</td></tr>
</table>
</body>
</html>
```



14.2.1 超文本和HTML

使用网页浏览器（比如IE、Firefox等）打开这个网页文件，就会看到如图14-2所示的网页内容。

排名	关键词	搜索指数
1	大数据	187767
2	云计算	178856
3	物联网	122376

图14-2 网页文件显示效果



14.2.2 HTTP

- HTTP是由万维网协会（World Wide Web Consortium）和 Internet 工作小组IETF（Internet Engineering Task Force）共同制定的规范。HTTP的全称是“Hyper Text Transfer Protocol”，中文名叫做“超文本传输协议”。
- HTTP协议是用于从网络传输超文本数据到本地浏览器的传送协议，它能保证高效而准确地传送超文本内容。
- HTTP是基于“客户端/服务器”架构进行通信的，HTTP的服务器端实现程序有httpd、nginx等，客户端的实现程序主要是Web浏览器，例如Firefox、Internet Explorer、Google Chrome、Safari、Opera等。Web浏览器和Web服务器之间可以通过HTTP进行通信。



14.2.2 HTTP

一个典型的HTTP请求过程如下（如图14-3所示）

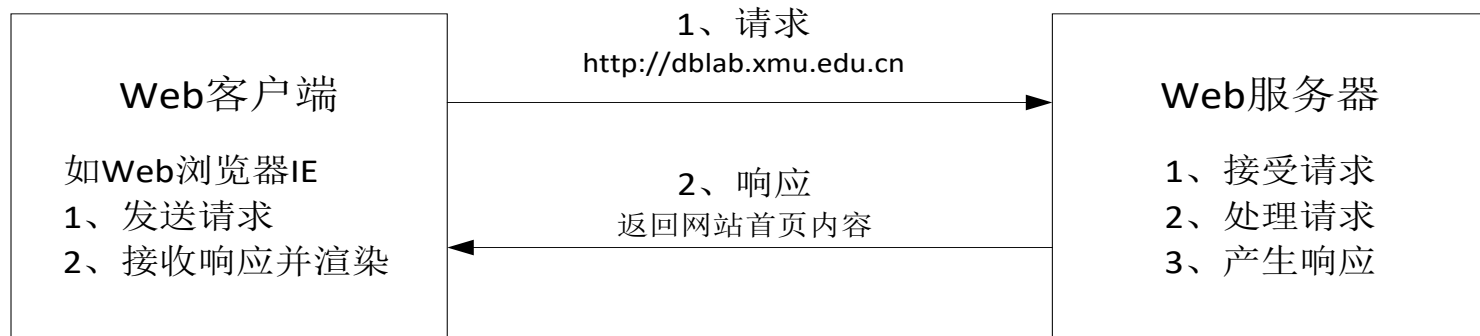


图14-3 一个典型的HTTP请求过程



14.3 用Python实现HTTP请求

14.3.1 urllib模块

14.3.2 urllib3模块

14.3.3 requests模块

本PPT是如下教材的配套讲义：

《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著, 人民邮电出版社

《Python程序设计基础教程（微课版）》教材官方网站：

<http://dblalab.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python

程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著



名师精品

多年计算机教学实践的厚积薄发



深入浅出

清晰呈现 Python 语言学习路径



实例丰富

有效提升编程语言的学习趣味



资源全面

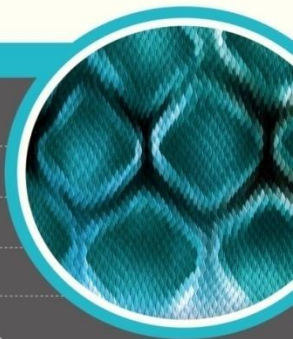
构建全方位一站式在线服务体系



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS





14.3.1 urllib模块

urllib是Python自带模块，该模块提供了一个urlopen()方法，通过该方法指定URL发送HTTP请求来获取数据。urllib提供了多个子模块，具体的模块名称与功能如表14-1所示。

表14-1 urllib中的子模块

模块名称	功能
urllib.request	该模块定义了打开URL（主要是HTTP）的方法和类，如身份验证、重定向和cookie等
urllib.error	该模块中主要包含异常类，基本的异常类是URLError
urllib.parse	该模块定义的功能分为两大类：URL解析和URL引用
urllib.robotparser	该模块用于解析robots.txt文件



14.3.1 urllib模块

下面是通过urllib.request模块实现发送GET请求获取网页内容的实例：

```
>>> import urllib.request
>>> response=urllib.request.urlopen("http://www.baidu.com")
>>> html=response.read()
>>> print(html)
```




14.3.1 urllib模块

下面是通过urllib.request模块实现发送POST请求获取网页内容的实例：

```
>>> import urllib.parse
>>> import urllib.request
>>> # 1.指定url
>>> url = 'https://fanyi.baidu.com/sug'
>>> # 2.发起POST请求之前，要处理POST请求携带的参数
>>> # 2.1 将POST请求封装到字典
>>> data = {'kw':'苹果',}
>>> # 2.2 使用parse模块中的urlencode(返回值类型是字符串类型)进行编码处理
>>> data = urllib.parse.urlencode(data)
>>> # 将步骤2.2的编码结果转换成byte类型
>>> data = data.encode()
```



14.3.1 urllib模块

>>> # 3.发起POST请求:urlopen函数的data参数表示的就是经过处理之后的POST请求携带的参数

```
>>> response = urllib.request.urlopen(url=url,data=data)
```

```
>>> data = response.read()
```

```
>>> print(data)
```

```
b'{"errno":0,"data":[{"k":"\u82f9\u679c","v":"\u540d.
apple"}, {"k":"\u82f9\u679c\u56ed","v":"apple
grove"}, {"k":"\u82f9\u679c\u5934","v":"apple
head"}, {"k":"\u82f9\u679c\u5e72","v":"[\u533b]dried
apple"}, {"k":"\u82f9\u679c\u6728","v":"applewood"}]}'
```



14.3.1 urllib模块

把上面`print(data)`执行的结果，拿到JSON在线格式校验网站（<http://www.bejson.com/>）进行处理，使用“Unicode转中文”功能可以得到如下结果：

```
b'{"errno":0,"data":[{"k":"苹\果","v":"\名. apple"}, {"k":"苹\果\园","v":"apple grove"}, {"k":"苹\果\头","v":"apple head"}, {"k":"苹\果\干","v":"[\医]dried apple"}, {"k":"苹\果\木","v":"applewood"}]}'
```



14.3.2 urllib3模块

- urllib3是一个功能强大、条理清晰、用于HTTP客户端的Python库，许多Python的原生系统已经开始使用urllib3。
- urllib3提供了很多Python标准库里所没有的重要特性，包括：线程安全、连接池、客户端SSL/TLS验证、文件分部编码上传、协助处理重复请求和HTTP重定位、支持压缩编码、支持HTTP和SOCKS代理、100%测试覆盖率等。



14.3.2 urllib3模块

在使用urllib3之前，需要打开一个cmd窗口使用如下命令进行安装：

```
> pip install urllib3
```

下面是通过GET请求获取网页内容的实例：

```
>>> import urllib3
>>> #需要一个PoolManager实例来生成请求，由该实例对象处理与线程池的连接以及线程安全的所有细节，不需要任何人为操作
>>> http = urllib3.PoolManager()
>>> response = http.request('GET','http://www.baidu.com')
>>> print(response.status)
>>> print(response.data)
```



14.3.2 urllib3模块

下面是通过POST请求获取网页内容的实例：

```
>>> import urllib3
>>> http = urllib3.PoolManager()
>>> response = http.request('POST',
                             'https://fanyi.baidu.com/sug'
                             ,fields={'kw':'苹果',})
>>> print(response.data)
```



14.3.3 requests模块

requests库是一个非常好用的HTTP请求库，可用于网络请求和网络爬虫等。在使用requests之前，需要打开一个cmd窗口使用如下命令进行安装：

```
> pip install requests
```

以GET请求方式为例，打印多种请求信息的代码如下：

```
>>> import requests
```

```
>>> response = requests.get('http://www.baidu.com') #对需要爬取的网页  
发送请求
```

```
>>> print('状态码:',response.status_code) #打印状态码
```

```
>>> print('url:',response.url) #打印请求url
```

```
>>> print('header:',response.headers) #打印头部信息
```

```
>>> print('cookie:',response.cookies) #打印cookie信息
```

```
>>> print('text:',response.text) #以文本形式打印网页源码
```

```
>>> print('content:',response.content) #以字节流形式打印网页源码
```



14.3.3 requests模块

以POST请求方式发送HTTP网页请求的示例代码如下：

```
>>> #导入模块
>>> import requests
>>> #表单参数
>>> data = {'kw':'苹果',}
>>> #对需要爬取的网页发送请求
>>> response = requests.post('https://fanyi.baidu.com/sug',data=data)
>>> #以字节流形式打印网页源码
>>> print(response.content)
```




14.4 定制requests

14.4.1 传递URL参数

14.4.2 定制请求头

14.4.3 网络超时

本PPT是如下教材的配套讲义：

《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《Python程序设计基础教程（微课版）》教材官方网站：

<http://dbl原因.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python

程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著



名师精品

多年计算机教学实践的厚积薄发



深入浅出

清晰呈现 Python 语言学习路径



实例丰富

有效提升编程语言的学习趣味



资源全面

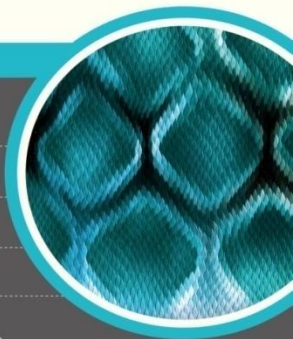
构建全方位一站式在线服务体系



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS





14.4.1 传递URL参数

为了请求特定的数据，我们需要在URL（Uniform Resource Locator）的查询字符串中加入一些特定数据。这些数据一般会跟在一个问号后面，并且以键值对的形式放在URL中。在requests中，我们可以直接把这些参数保存在字典中，用params构建到URL中。具体实例如下：

```
>>> import requests
>>> base_url = 'http://httpbin.org'
>>> param_data = {'user':'xmu','password':'123456'}
>>> response = requests.get(base_url+'/get',params=param_data)
>>> print(response.url)
http://httpbin.org/get?user=xmu&password=123456
>>> print(response.status_code)
200
```



14.4.2 定制请求头

- 在爬取网页的时候，输出的信息中有时候会出现“抱歉，无法访问”等字眼，这就是禁止爬取，需要通过定制请求头**Headers**来解决这个问题。
- 定制**Headers**是解决**requests**请求被拒绝的方法之一，相当于我们进入这个网页服务器，假装自己本身在爬取数据。
- 请求头**Headers**提供了关于请求、响应或其他发送实体的消息，如果没有定制请求头或请求的请求头和实际网页不一致，就可能无法返回正确结果。



14.4.2 定制请求头

获取一个网页的Headers的方法如下：使用360、火狐或谷歌浏览器打开一个网址（比如“<http://httpbin.org/>”），在网页上单击鼠标右键，在弹出的菜单中选择“查看元素”，然后刷新网页，再按照如图14-4所示的步骤，先点击“Network”选项卡，再点击“Doc”，接下来点击“Name”下方的网址，就会出现类似如下的Headers信息：

User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36

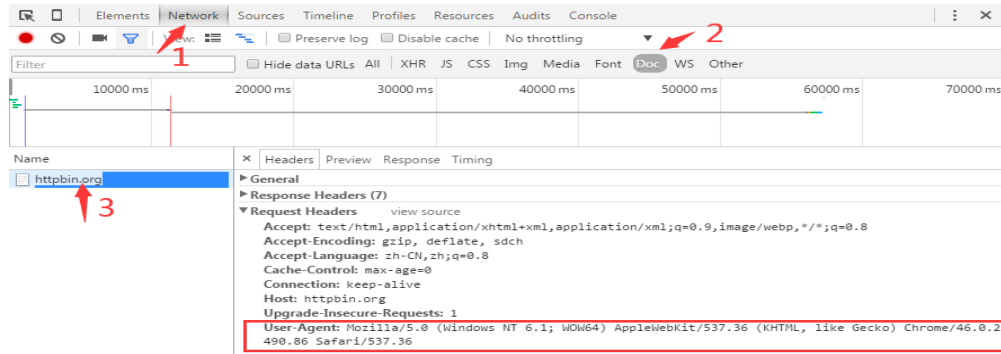


图14-4 查看请求头Headers



14.4.2 定制请求头

Headers中有很多内容，主要常用的就是“User-Agent”和“Host”，它们是以键对的形式呈现的，如果把“User-Agent”以字典键值对形式作为Headers的内容，往往就可以顺利爬取网页内容。

下面是添加了Headers信息的网页请求过程：

```
>>> import requests
>>> url='http://httpbin.org'
>>> # 创建头部信息
>>> headers={'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86
Safari/537.36'}
>>> response = requests.get(url,headers=headers)
>>> print(response.content)
```



14.4.3 网络超时

网络请求不可避免会遇上请求超时的情况，这个时候，网络数据采集的程序会一直运行等待进程，造成网络数据采集程序不能很好地顺利执行。因此，可以为requests的timeout参数设定等待秒数，如果服务器在规定时间内没有应答就返回异常。具体代码如下：

```
01 # time_out.py
02 import requests
03 from requests.exceptions import ReadTimeout,ConnectTimeout
04 try:
05     response = requests.get("http://www.baidu.com", timeout=0.5)
06     print(response.status_code)
07 except ReadTimeout or ConnectTimeout:
08     print('Timeout')
```



14.5解析网页

- 14.5.1 BeautifulSoup简介
- 14.5.2 BeautifulSoup四大对象
- 14.5.3遍历文档树
- 14.5.4 搜索文档树
- 14.5.5 CSS选择器

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社
《Python程序设计基础教程（微课版）》教材官方网站：
<http://dbl原因.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python 程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著

-  **名师精品**
多年计算机教学实践的厚积薄发
-  **深入浅出**
清晰呈现 Python 语言学习路径
-  **实例丰富**
有效提升编程语言的学习趣味
-  **资源全面**
构建全方位一站式在线服务体系

中国工信出版集团 人民邮电出版社
POSTS & TELECOM PRESS



14.5.1 BeautifulSoup简介

BeautifulSoup提供一些简单的、Python式的函数来处理导航、搜索、修改分析树等。BeautifulSoup是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。BeautifulSoup自动将输入文档转换为Unicode编码，输出文档转换为UTF-8编码。BeautifulSoup3已经停止开发，目前推荐使用BeautifulSoup4，不过它已经被移植到bs4当中了，所以，在使用BeautifulSoup4之前，需要安装bs4：

```
> pip install bs4
```




14.5.1 BeautifulSoup简介

使用BeautifulSoup解析HTML比较简单，API非常人性化，支持CSS选择器、Python标准库中的HTML解析器，也支持lxml的XML解析器和HTML解析器，此外还支持html5lib解析器，表14-2给出了每个解析器的优缺点。

解析器	用法	优点	缺点
Python标准库	<code>BeautifulSoup(markup,"html.parser")</code>	Python标准库 执行速度适中	文档容错能力差
lxml的HTML解析器	<code>BeautifulSoup(markup,"lxml")</code>	速度快 文档容错能力强	需要安装C语言库
lxml的XML解析器	<code>BeautifulSoup(markup,"lxml-xml")</code> <code>BeautifulSoup(markup,"xml")</code>	速度快 唯一支持XML的解析器	需要安装C语言库
html5lib	<code>BeautifulSoup(markup,"html5lib")</code>	兼容性好 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



14.5.1 BeautifulSoup简介

下面给出一个BeautifulSoup解析网页的简单实例，使用了lxml解析器，在使用之前，需要执行如下命令安装lxml解析器：

```
> pip install lxml
```

下面是实例代码：

```
>>> html_doc = """
<html><head><title>BigData Software</title></head>
<p class="title"><b>BigData Software</b></p>
<p class="bigdata">There are three famous bigdata softwares; and their names
are
<a href="http://example.com/hadoop" class="software" id="link1">Hadoop</a>,
<a href="http://example.com/spark" class="software" id="link2">Spark</a> and
<a href="http://example.com/flink" class="software" id="link3">Flink</a>;
and they are widely used in real applications.</p>
<p class="bigdata">...</p>
"""
```



14.5.1 BeautifulSoup简介

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(html_doc,"lxml")
>>> content = soup.prettify()
>>> print(content)
<html>
<head>
  <title>
    BigData Software
  </title>
</head>
<body>
  <p class="title">
    <b>
      BigData Software
    </b>
  </p>
```



14.5.1 BeautifulSoup简介

```
<p class="bigdata">
  There are three famous bigdata softwares; and their names are
  <a class="software" href="http://example.com/hadoop" id="link1">
    Hadoop
  </a>
  ,
  <a class="software" href="http://example.com/spark" id="link2">
    Spark
  </a>
  and
  <a class="software" href="http://example.com/flink" id="link3">
    Flink
  </a>
  ;
  and they are widely used in real applications.
</p>
<p class="bigdata">
  ...
</p>
</body>
</html>
```



14.5.1 BeautifulSoup简介

如果要更换解析器，比如要使用Python标准库的解析器，只需要把上面的“`soup = BeautifulSoup(html_doc, "lxml")`”这行代码替换成如下代码即可：

```
soup = BeautifulSoup(html_doc, "html.parser")
```



14.5.2 BeautifulSoup四大对象

BeautifulSoup将复杂HTML文档转换成一个复杂的树形结构，每个节点都是Python对象，所有对象可以归纳为4种：Tag、NavigableString、BeautifulSoup、Comment。



14.5.2 BeautifulSoup四大对象

1.Tag

Tag就是HTML中的一个标签，例如：

```
<title>BigData Software</title>  
<a href="http://example.com/hadoop" class="software"  
id="link1">Hadoop</a>
```

上面的<title>、<a>等标签加上里面包括的内容就是Tag，利用soup加标签名可以轻松地获取这些标签的内容。作为演示，我们可以在上一节的实例代码之后继续执行以下代码：

```
>>> print(soup.a)  
<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>  
>>> print(soup.title)  
<title>BigData Software</title>
```



14.5.2 BeautifulSoup四大对象

Tag有两个重要的属性，即name和attrs。下面继续执行如下代码：

```
>>> print(soup.name)
[document]
>>> print(soup.p.attrs)
{'class': ['title']}
```

如果想要单独获取某个属性，比如要获取“class”属性的值，可以执行如下代码：

```
>>> print(soup.p['class'])
['title']
```

还可以利用get方法获得属性的值，代码如下：

```
>>> print(soup.p.get('class'))
['title']
```




14.5.2 BeautifulSoup四大对象

2. NavigableString

`NavigableString`对象用于操纵字符串。在网页解析时，已经得到了标签的内容以后，如果我们想获取标签内部的文字，则可以使用`.string`方法，其返回值就是一个`NavigableString`对象，具体实例如下：

```
>>> print(soup.p.string)
BigData Software
>>> print(type(soup.p.string))
<class 'bs4.element.NavigableString'>
```



14.5.2 BeautifulSoup四大对象

3. BeautifulSoup

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，是一个特殊的Tag。例如，可以分别获取它的类型、名称以及属性：

```
>>> print(type(soup.name))
<class 'str'>
>>> print(soup.name)
[document]
>>> print(soup.attrs)
{}
```



14.5.2 BeautifulSoup四大对象

4.Comment

Comment对象是一种特殊类型的NavigableString对象，输出的内容不包括注释符号。如果它处理不好，可能会对文本处理造成意想不到的麻烦。为了演示Comment对象，这里重新创建一个代码文件bs4_example.py:



14.5.2 BeautifulSoup四大对象

```
01 # bs4_example.py
02 html_doc = """
03 <html><head><title>The Dormouse's story</title></head>
04 <p class="title"><b>The Dormouse's story</b></p>
05 <p class="story">Once upon a time there were three little sisters; and their
names were
06 <a href="http://example.com/elsie" class="sister" id="link1"><!-- Elsie --></a>,
07 <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
08 <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
09 and they lived at the bottom of a well.</p>
10 <p class="story">...</p>
11 """
12 from bs4 import BeautifulSoup
13 soup = BeautifulSoup(html_doc,"lxml")
14 print(soup.a)
15 print(soup.a.string)
16 print(type(soup.a.string))
```



14.5.2 BeautifulSoup四大对象

该代码文件的执行结果如下：

```
<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>
Elsie
<class 'bs4.element.Comment'>
```

从上面执行结果可以看出，`a`标签里的内容“`<!-- Elsie -->`”实际上是注释，但是使用语句`print(soup.a.string)`输出它的内容以后会发现，它已经把注释符号去掉了，只输出了“`Elsie`”，所以这可能会给我们带来不必要的麻烦。另外我们打印输出它的类型，发现它是一个`Comment`类型。



14.5.3 遍历文档树

遍历文档树就是从根节点html标签开始遍历，直到找到目标元素为止。

1. 直接子节点

(1) .contents属性

Tag对象的.contents属性可以将某个Tag的子节点以列表的方式输出，当然列表会允许用索引的方式来获取列表中的元素。下面是示例代码：

```
>>> html_doc = """
<html><head><title>BigData Software</title></head>
<p class="title"><b>BigData Software</b></p>
<p class="bigdata">There are three famous bigdata softwares; and their names are
<a href="http://example.com/hadoop" class="software" id="link1">Hadoop</a>,
<a href="http://example.com/spark" class="software" id="link2">Spark</a> and
<a href="http://example.com/flink" class="software" id="link3">Flink</a>;
and they are widely used in real applications.</p>
<p class="bigdata">...</p>
"""
```



14.5.3 遍历文档树

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(html_doc,"lxml")
>>> print(soup.body.contents)
[<p class="title"><b>BigData Software</b></p>, '\n', <p
class="bigdata">There are three famous bigdata softwares; and their
names are
<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>,
<a class="software" href="http://example.com/spark"
id="link2">Spark</a> and
<a class="software" href="http://example.com/flink"
id="link3">Flink</a>;
and they are widely used in real applications.</p>, '\n', <p
class="bigdata">...</p>, '\n']
```



14.5.3 遍历文档树

可以使用索引的方式来获取列表中的元素：

```
>>> print(soup.body.contents[0])  
<p class="title"><b>BigData Software</b></p>
```




14.5.3 遍历文档树

(2) .children属性

Tag对象的.children属性是一个迭代器，可以使用for循环进行遍历，代码如下：

```
>>> for child in soup.body.children:  
    print(child)
```



14.5.3 遍历文档树

上面代码的执行结果如下：

```
<p class="title"><b>BigData Software</b></p>
```

```
<p class="bigdata">There are three famous bigdata softwares; and their names are
```

```
<a class="software" href="http://example.com/hadoop" id="link1">Hadoop</a>,
```

```
<a class="software" href="http://example.com/spark" id="link2">Spark</a> and
```

```
<a class="software" href="http://example.com/flink" id="link3">Flink</a>;  
and they are widely used in real applications.</p>
```

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



14.5.3 遍历文档树

2. 所有子孙节点

在获取所有子孙节点时，可以使用`.descendants`属性，与Tag对象的`.children`和`.contents`仅包含Tag对象的直接子节点不同，该属性是将Tag对象的所有子孙结点进行递归循环，然后生成生成器。示例代码如下：

```
>>> for child in soup.descendants:  
    print(child)
```

上面代码的执行结果较多，因此这里没有给出。在执行结果中，所有的节点都会被打印出来，先生成最外层的`html`标签，其次从`head`标签一个个剥离，依此类推。



14.5.3 遍历文档树

3. 节点内容

(1) Tag对象内没有标签的情况。

```
>>> print(soup.title)
<title>BigData Software</title>
>>> print(soup.title.string)
BigData Software
```

(2) Tag对象内有一个标签的情况。

```
>>> print(soup.head)
<head><title>BigData
Software</title></head>
>>> print(soup.head.string)
BigData Software
```



14.5.3 遍历文档树

(3) Tag对象内有多个标签的情况。

```
>>> print(soup.body)
<body><p class="title"><b>BigData Software</b></p>
<p class="bigdata">There are three famous bigdata softwares; and
their names are
<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>,
<a class="software" href="http://example.com/spark"
id="link2">Spark</a> and
<a class="software" href="http://example.com/flink"
id="link3">Flink</a>;
and they are widely used in real applications.</p>
<p class="bigdata">...</p>
</body>
```



14.5.3 遍历文档树

从上面的执行结果中可以看出，`body`标签内包含了多个

标签，这时如果使用`.string`获取子节点内容，就会返回`None`，代码如下：

```
>>> print(soup.body.string)
None
```

也就是说，如果`Tag`包含了多个子节点，`Tag`就无法确定`.string`应该调用哪个子节点的内容，因此`.string`的输出结果是`None`。这时应该使用`.strings`属性或`.stripped_strings`属性，它们获得的都是一个生成器，示例代码如下：

```
>>> print(soup.strings)
<generator object Tag._all_strings at 0x0000000002C4D190>
```

可以用`for`循环对生成器进行遍历，代码如下：

```
>>> for string in soup.strings:
    print(repr(string))
```



14.5.3 遍历文档树

上面代码的执行结果如下：

```
'BigData Software'  
\n'  
'BigData Software'  
\n'  
'There are three famous bigdata softwares; and their names are\n'  
'Hadoop'  
,\n'  
'Spark'  
' and\n'  
'Flink'  
,\nand they are widely used in real applications.'  
\n'  
'...'  
\n'
```



14.5.3 遍历文档树

使用Tag对象的`.stripped_strings`属性，可以获得去掉空白行的标签内的众多内容，示例代码如下：

```
>>> for string in soup.stripped_strings:  
    print(string)
```

上面代码的执行结果如下：

BigData Software

BigData Software

There are three famous bigdata softwares; and their names are

Hadoop

,

Spark

and

Flink

;

and they are widely used in real applications.

...



14.5.3 遍历文档树

4. 直接父节点

使用Tag对象的.parent属性可以获得父节点，使用Tag对象的.parents属性可以获得从父到根的所有节点。

下面是获取标签的父节点：

```
>>> p = soup.p
>>> print(p.parent.name)
body
```

下面是获取内容的父节点：

```
>>> content = soup.head.title.string
>>> print(content)
BigData Software
>>> print(content.parent.name)
title
```



14.5.3 遍历文档树

Tag对象的.parents属性，得到的也是一个生成器：

```
>>> content = soup.head.title.string
>>> print(content)
BigData Software
>>> for parent in content.parents:
    print(parent.name)
```

上面语句的执行结果如下：

```
title
head
html
[document]
```



14.5.3 遍历文档树

5. 兄弟节点

可以使用Tag对象的`.next_sibling`和`.previous_sibling`属性分别获取下一个兄弟结点和获取上一个兄弟结点。需要注意的是，实际文档中Tag的`.next_sibling`和`.previous_sibling`属性通常是字符串或空白，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。



14.5.3 遍历文档树

示例代码如下：

```
>>> print(soup.p.next_sibling)
```

```
# 此处返回为换行
```

```
>>> print(soup.p.prev_sibling)
```

```
None #没有前一个兄弟节点，返回None
```

```
>>> print(soup.p.next_sibling.next_sibling)
```

```
<p class="bigdata">There are three famous bigdata softwares; and their names are
```

```
<a class="software" href="http://example.com/hadoop" id="link1">Hadoop</a>,
```

```
<a class="software" href="http://example.com/spark" id="link2">Spark</a> and
```

```
<a class="software" href="http://example.com/flink" id="link3">Flink</a>;
```

```
and they are widely used in real applications.</p>
```



14.5.3 遍历文档树

6. 全部兄弟节点

可以使用Tag对象的`.next_siblings`和`.previous_siblings`属性对当前的兄弟结点迭代输出。示例代码如下：

```
>>> for next in soup.a.next_siblings:  
    print(repr(next))
```

执行结果如下：

```
'\n'  
<a class="software" href="http://example.com/spark"  
id="link2">Spark</a>  
' and\n'  
<a class="software" href="http://example.com/flink" id="link3">Flink</a>  
';\nand they are widely used in real applications.'
```



14.5.3 遍历文档树

7. 前后节点

Tag对象的`.next_element`和`.previous_element`属性，用于获得不分层次的前后元素，示例代码如下：

```
>>> print(soup.head.next_element)
<title>BigData Software</title>
```



14.5.3 遍历文档树

8. 所有前后节点

使用Tag对象的`.next_elements`和`.previous_elements`属性可以向前或向后解析文档内容，示例代码如下：

```
>>> for element in soup.a.next_elements:  
    print(repr(element))
```



14.5.3 遍历文档树

执行结果如下：

```
'Hadoop'  
,\n<a class="software" href="http://example.com/spark"  
id="link2">Spark</a>  
'Spark'  
' and\n<a class="software" href="http://example.com/flink"  
id="link3">Flink</a>  
'Flink'  
';\nand they are widely used in real applications.'  
\n<p class="bigdata">...</p>  
'...'  
\n
```




14.5.4 搜索文档树

搜索文档树是通过指定标签名来搜索元素，另外还可以通过指定标签的属性值来精确定位某个节点元素，最常用的两个方法就是`find()`和`find_all()`，这两个方法在`BeautifulSoup`和`Tag`对象上都可以被调用。

1.find_all()

`find_all()`方法搜索当前`Tag`的所有`Tag`子节点，并判断是否符合过滤器的条件，它的函数原型是：

```
find_all( name , attrs , recursive , text , **kwargs )
```

`find_all()`的返回值是一个`Tag`组成的列表，方法调用非常灵活，所有的参数都是可选的。



14.5.4 搜索文档树

(1) name参数

name参数可以查找所有名字为name的Tag，字符串对象会被自动忽略掉。

①传入字符串

查找所有名字为a的Tag，代码如下：

```
>>> print(soup.find_all('a'))  
[<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>, <a class="software" href="http://example.com/spark"  
id="link2">Spark</a>, <a class="software" href="http://example.com/flink"  
id="link3">Flink</a>]
```



14.5.4 搜索文档树

②传入正则表达式

如果传入正则表达式作为参数，**BeautifulSoup**会通过正则表达式的`match()`来匹配内容.下面的例子中我们要找出所有以**b**开头的标签，这意味着**<body>**和****标签都应该被找到：

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



14.5.4 搜索文档树

执行结果如下：

```
<body><p class="title"><b>BigData Software</b></p>
<p class="bigdata">There are three famous bigdata softwares; and their
names are
<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>,
<a class="software" href="http://example.com/spark" id="link2">Spark</a>
and
<a class="software" href="http://example.com/flink" id="link3">Flink</a>;
and they are widely used in real applications.</p>
<p class="bigdata">...</p>
</body>
<b>BigData Software</b>
```



14.5.4 搜索文档树

③传入列表

如果传入参数是列表，BeautifulSoup会将与列表中任一元素匹配的内容返回。下面代码会找到文档中所有<a>标签和标签：

```
>>> print(soup.find_all("a", "b"))  
[<b>BigData Software</b>, <a class="software"  
href="http://example.com/hadoop" id="link1">Hadoop</a>, <a  
class="software" href="http://example.com/spark" id="link2">Spark</a>,  
<a class="software" href="http://example.com/flink" id="link3">Flink</a>]
```



14.5.4 搜索文档树

④传入True

传入True可以找到所有的标签。下面的例子是在文档树中查找所有包含id属性的标签，无论id的值是什么：

```
>>> print(soup.find_all(id=True))  
[<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>, <a class="software"  
href="http://example.com/spark" id="link2">Spark</a>, <a  
class="software" href="http://example.com/flink" id="link3">Flink</a>]
```



14.5.4 搜索文档树

⑤传入方法

如果没有合适过滤器，那么还可以定义一个方法，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，如果不是则返回**False**。下面方法对当前元素进行校验，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')
```



14.5.4 搜索文档树

将这个作为参数传入find_all()方法，将得到所有<p>标签：

```
>>> print(soup.find_all(has_class_but_no_id))  
[<p class="title"><b>BigData Software</b></p>, <p  
class="bigdata">There are three famous bigdata softwares; and their  
names are  
<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>,  
<a class="software" href="http://example.com/spark"  
id="link2">Spark</a> and  
<a class="software" href="http://example.com/flink"  
id="link3">Flink</a>;  
and they are widely used in real applications.</p>, <p  
class="bigdata">...</p>]
```




14.5.4 搜索文档树

(2) keyword参数

通过name参数是搜索Tag的标签类型名称，如a、head、title等。如果要通过标签内属性的值来搜索，要通过键值对的形式来指定，实例如下：

```
>>> import re
>>> print(soup.find_all(id='link2'))
[<a class="software" href="http://example.com/spark" id="link2">Spark</a>]
>>> print(soup.find_all(href=re.compile("spark")))
[<a class="software" href="http://example.com/spark" id="link2">Spark</a>]
```



14.5.4 搜索文档树

使用多个指定名字的参数可以同时过滤Tag的多个属性:

```
>>> soup.find_all(href=re.compile("hadoop"), id='link1')
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>]
```

如果指定的key是Python的关键词, 则后面需要加下划线:

```
>>> print(soup.find_all(class_="software"))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>, <a class="software"
href="http://example.com/spark" id="link2">Spark</a>, <a
class="software" href="http://example.com/flink" id="link3">Flink</a>]
```



14.5.4 搜索文档树

(3) text参数

`text`参数的作用和`name`参数类似，但是`text`参数的搜索范围是文档中的字符串内容（不包含注释），并且是完全匹配，当然也接受正则表达式、列表、`True`。实例如下：

```
>>> import re
>>> print(soup.a)
<a class="software" href="http://example.com/hadoop" id="link1">Hadoop</a>
>>> print(soup.find_all(text="Hadoop"))
['Hadoop']
>>> print(soup.find_all(text=["Hadoop", "Spark", "Flink"]))
['Hadoop', 'Spark', 'Flink']
>>> print(soup.find_all(text="bigdata"))
[]
>>> print(soup.find_all(text="BigData Software"))
['BigData Software', 'BigData Software']
>>> print(soup.find_all(text=re.compile("bigdata")))
['There are three famous bigdata softwares; and their names are\n']
```



14.5.4 搜索文档树

(4) limit参数

可以通过limit参数来限制使用name参数或者attrs参数过滤出来的条目的数量，实例如下：

```
>>> print(soup.find_all("a"))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>, <a class="software"
href="http://example.com/spark" id="link2">Spark</a>, <a class="software"
href="http://example.com/flink" id="link3">Flink</a>]
>>> print(soup.find_all("a",limit=2))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>, <a class="software"
href="http://example.com/spark" id="link2">Spark</a>]
```



14.5.4 搜索文档树

(5) recursive 参数

调用Tag的find_all()方法时，BeautifulSoup会检索当前Tag的所有子孙节点，如果只想搜索Tag的直接子节点，可以使用参数recursive=False，实例如下：

```
>>> print(soup.body.find_all("a",recursive=False))  
[]
```

在这个例子中，a标签都是在p标签内的，所以在body的直接子节点下搜索a标签是无法匹配到a标签的。



14.5.4 搜索文档树

2.find()

`find()`与`find_all()`的区别是，`find_all()`将所有匹配的条目组合成一个列表，而`find()`仅返回第一个匹配的条目，除此以外，二者的用法都相同。



14.5.5 CSS选择器

BeautifulSoup支持大部分的CSS选择器，在Tag或BeautifulSoup对象的select()方法中传入字符串参数，即可使用CSS选择器的语法找到标签。

①通过标签名查找

```
>>> print(soup.select('title'))
[<title>BigData Software</title>]
>>> print(soup.select('a'))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>, <a class="software"
href="http://example.com/spark" id="link2">Spark</a>, <a
class="software" href="http://example.com/flink" id="link3">Flink</a>]
>>> print(soup.select('b'))
[<b>BigData Software</b>]
```



14.5.5 CSS选择器

②通过类名查找

```
>>> print(soup.select('.software'))  
[<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>, <a class="software"  
href="http://example.com/spark" id="link2">Spark</a>, <a  
class="software" href="http://example.com/flink" id="link3">Flink</a>]
```

③通过id名查找

```
>>> print(soup.select('#link1'))  
[<a class="software" href="http://example.com/hadoop"  
id="link1">Hadoop</a>]
```




14.5.5 CSS选择器

④组合查找

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

```
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>]
```

```
>>> print(soup.select("head > title"))
```

```
[<title>BigData Software</title>]
```

```
>>> print(soup.select("p > a:nth-of-type(1)"))
```

```
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>]
```

```
>>> print(soup.select("p > a:nth-of-type(2)"))
```

```
[<a class="software" href="http://example.com/spark" id="link2">Spark</a>]
```

```
>>> print(soup.select("p > a:nth-of-type(3)"))
```

```
[<a class="software" href="http://example.com/flink" id="link3">Flink</a>]
```

在上面的语句中，"p > a:nth-of-type(n)"的含义是选择p标签下面的第n个a标签。



14.5.5 CSS选择器

⑤属性查找

查找时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。

```
>>> print(soup.select('a[class="software"]'))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>, <a class="software"
href="http://example.com/spark" id="link2">Spark</a>, <a class="software"
href="http://example.com/flink" id="link3">Flink</a>]
>>> print(soup.select('a[href="http://example.com/hadoop"]'))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>]
>>> print(soup.select('p a[href="http://example.com/hadoop"]'))
[<a class="software" href="http://example.com/hadoop"
id="link1">Hadoop</a>]
```



14.5.5 CSS选择器

以上的select方法返回的结果都是列表形式，可以以遍历的形式进行输出，然后用 `get_text()` 方法来获取它的内容，实例如下：

```
>>> print(type(soup.select('title')))
<class 'bs4.element.ResultSet'>
>>> print(soup.select('title')[0].get_text())
BigData Software
>>> for title in soup.select('title'):
    print(title.get_text())
```

上面语句的执行结果如下：

```
BigData Software
```



14.6 综合实例

14.6.2 采集网页数据保存到MySQL数据库

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社
《Python程序设计基础教程（微课版）》教材官方网站：
<http://dbl原因.xmu.edu.cn/post/python>

高等院校程序设计新形态精品系列

PYTHON

Python Programming Language

Python 程序设计基础教程

| 微课版 |

林子雨 赵江声 陶继平 编著



名师精品

多年计算机教学实践的厚积薄发



深入浅出

清晰呈现 Python 语言学习路径



实例丰富

有效提升编程语言的学习趣味



资源全面

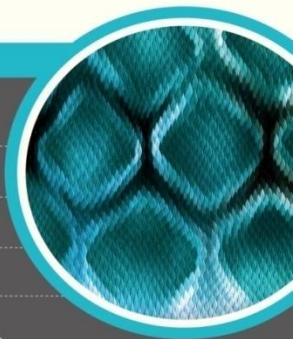
构建全方位一站式在线服务体系



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS





14.6.1 采集网页数据保存到文本文件

访问古诗文网站（<https://so.gushiwen.org/mingju/>），会显示如图14-5所示的页面，里面包含了很多名句，点击某一个名句（比如“山有木兮木有枝，心悦君兮君不知”），就会出现完整的古诗（如图14-6所示）。



图14-5 名句页面



14.6.1 采集网页数据保存到文本文件



图14-6 完整古诗页面



14.6.1 采集网页数据保存到文本文件

下面编写网络爬虫程序，爬取名句页面的内容，保存到一个文本文件中，然后，再爬取每个名句的完整古诗页面，把完整古诗保存到一个文本文件中。可以打开一个浏览器，访问要爬取的网页，然后在浏览器中查看网页源代码，找到诗句内容所在的位置，总结出它们共同的特征，就可以将它们全部提取出来了，具体实现代码如下：



14.6.1 采集网页数据保存到文本文件

```
01 #parse_poem.py
02 import requests
03 from bs4 import BeautifulSoup
04 import time
05
06 #函数1: 请求网页
07 def page_request(url,ua):
08     response = requests.get(url,headers = ua)
09     html = response.content.decode('utf-8')
10     return html
11
```




14.6.1 采集网页数据保存到文本文件

```
12 #函数2: 解析网页
13 def page_parse(html):
14     soup = BeautifulSoup(html,'lxml')
15     title = soup('title')
16     sentence = soup.select('div.left > div.sons > div.cont > a:nth-of-type(1)')
17     poet = soup.select('div.left > div.sons > div.cont > a:nth-of-type(2)')
18     sentence_list=[]
19     href_list=[]
20     for i in range(len(sentence)):
21         temp = sentence[i].get_text()+ "---"+poet[i].get_text()
22         sentence_list.append(temp)
23         href = sentence[i].get('href')
24         href_list.append("https://so.gushiwen.org"+href)
25     return [href_list,sentence_list]
26
```



14.6.1 采集网页数据保存到文本文件

```
27 #函数3: 写入文本文件
28 def save_txt(info_list):
29     import json
30     with open(r'C:\\sentence.txt','a',encoding='utf-8') as txt_file:
31         for element in info_list[1]:
32             txt_file.write(json.dumps(element,ensure_ascii=False)+'\n\n')
33
```



14.6.1 采集网页数据保存到文本文件

```
34 #子网页处理函数：进入并解析子网页/请求子网页
35 def sub_page_request(info_list):
36     subpage_urls = info_list[0]
37     ua = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86
Safari/537.36'}
38     sub_html = []
39     for url in subpage_urls:
40         html = page_request(url,ua)
41         sub_html.append(html)
42     return sub_html
43
```



14.6.1 采集网页数据保存到文本文件

```
44 #子网页处理函数：解析子网页，爬取诗句内容
45 def sub_page_parse(sub_html):
46     poem_list=[]
47     for html in sub_html:
48         soup = BeautifulSoup(html,' lxml ')
49         poem = soup.select('div.left > div.sons > div.cont > div.contson')
50         poem = poem[0].get_text()
51         poem_list.append(poem.strip())
52     return poem_list
53
```



14.6.1 采集网页数据保存到文本文件

```
54 #子网页处理函数：保存诗句到txt
55 def sub_page_save(poem_list):
56     import json
57     with open(r'C:\\poems.txt','a',encoding='utf-8') as txt_file:
58         for element in poem_list:
59             txt_file.write(json.dumps(element,ensure_ascii=False)+'\n\n')
60
```



14.6.1 采集网页数据保存到文本文件

```
61 if __name__ == '__main__':
62     print("*****开始爬取古诗文网站*****")
63     ua = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/46.0.2490.86 Safari/537.36'}
64     for i in range(1,4):
65         url = 'https://so.gushiwen.org/mingju/default.aspx?p=%d&c=&t=%'(i)
66         time.sleep(1)
67         html = page_request(url,ua)
68         info_list = page_parse(html)
69         save_txt(info_list)
70         #处理子网页
71         print("开始解析第%d"%(i)+"页")
72         #开始解析名句子网页
73         sub_html = sub_page_request(info_list)
74         poem_list = sub_page_parse(sub_html)
75         sub_page_save(poem_list)
76
77     print("*****爬取完成*****")
78     print("共爬取%d"%(i*50)+"个古诗词名句，保存在如下路径：C:\\sentence.txt")
79     print("共爬取%d"%(i*50)+"个古诗词，保存在如下路径：C:\\poem.txt")
```



14.6.2 采集网页数据保存到MySQL数据库

由于很多网站设计了反爬机制，会导致爬取网页失败，因此，这里直接采集一个本地网页文件web_demo.html，它记录了不同关键词的搜索次数排名，其内容如下：

```
<html>
<head><title>搜索指数</title></head>
<body>
<table>
<tr><td>排名</td><td>关键词</td><td>搜索指数</td></tr>
<tr><td>1</td><td>大数据</td><td>187767</td></tr>
<tr><td>2</td><td>云计算</td><td>178856</td></tr>
<tr><td>3</td><td>物联网</td><td>122376</td></tr>
</table>
</body>
</html>
```



14.6.2 采集网页数据保存到MySQL数据库

参照第11章的内容，在Windows系统中启动MySQL服务进程，打开MySQL命令行客户端，执行如下SQL语句创建数据库和表：

```
mysql > CREATE DATABASE webdb;  
mysql > USE webdb;  
mysql > CREATE TABLE search_index(  
    -> id int,  
    -> keyword char(20),  
    -> number int);
```




14.6.2 采集网页数据保存到MySQL数据库

编写网络爬虫程序，读取网页内容进行解析，并把解析后的数据保存到MySQL数据库中，具体代码如下：

```
01 # html_to_mysql.py
02 import requests
03 from bs4 import BeautifulSoup
04
05 # 读取本地HTML文件
06 def get_html():
07     path = 'C:/web_demo.html'
08     htmlfile= open(path,'r')
09     html = htmlfile.read()
10     return html
11
```



14.6.2 采集网页数据保存到MySQL数据库

```
12 # 解析HTML文件
13 def parse_html(html):
14     soup = BeautifulSoup(html,'html.parser')
15     all_tr=soup.find_all('tr')[1:]
16     all_tr_list = []
17     info_list = []
18     for i in range(len(all_tr)):
19         all_tr_list.append(all_tr[i])
20     for element in all_tr_list:
21         all_td=element.find_all('td')
22         all_td_list = []
23         for j in range(len(all_td)):
24             all_td_list.append(all_td[j].string)
25         info_list.append(all_td_list)
26     return info_list
27
```



14.6.2 采集网页数据保存到MySQL数据库

```
28 # 保存数据库
29 def save_mysql(info_list):
30     import pymysql.cursors
31     # 连接数据库
32     connect = pymysql.Connect(
33         host='localhost',
34         port=3306,
35         user='root', # 数据库用户名
36         passwd='123456', # 密码
37         db='webdb',
38         charset='utf8'
39     )
40
41     # 获取游标
42     cursor = connect.cursor()
43
```



14.6.2 采集网页数据保存到MySQL数据库

```
44 # 插入数据
45 for item in info_list:
46     id = int(item[0])
47     keyword = item[1]
48     number = int(item[2])
49     sql = "INSERT INTO search_index(id,keyword,number) VALUES ('%d', '%s', %d)"
50     data = (id,keyword,number)
51     cursor.execute(sql % data)
52     connect.commit()
53     print('成功插入数据')
54
55 # 关闭数据库连接
56 connect.close()
57
58 if __name__ == '__main__':
59     html = get_html()
60     info_list = parse_html(html)
61     save_mysql(info_list)
```



14.6.2 采集网页数据保存到MySQL数据库

执行代码文件，然后到MySQL命令行客户端执行如下SQL语句查看数据：

```
mysql> select * from search_index;
```

可以看到（如图14-7所示），有3条数据被成功插入了数据库中。

```
mysql> select * from search_index;
+----+-----+-----+
| id  | keyword | number |
+----+-----+-----+
| 1   | 大数据 | 187767 |
| 2   | 云计算 | 178856 |
| 3   | 物联网 | 122376 |
+----+-----+-----+
3 rows in set (0.21 sec)
```

图14-7 search_index表中的记录



附录A：主讲教师林子雨简介



主讲教师：林子雨

单位：厦门大学计算机科学与技术系

E-mail: ziyulin@xmu.edu.cn

个人网页: <http://dmlab.xmu.edu.cn/post/linziyu>

数据库实验室网站: <http://dmlab.xmu.edu.cn>

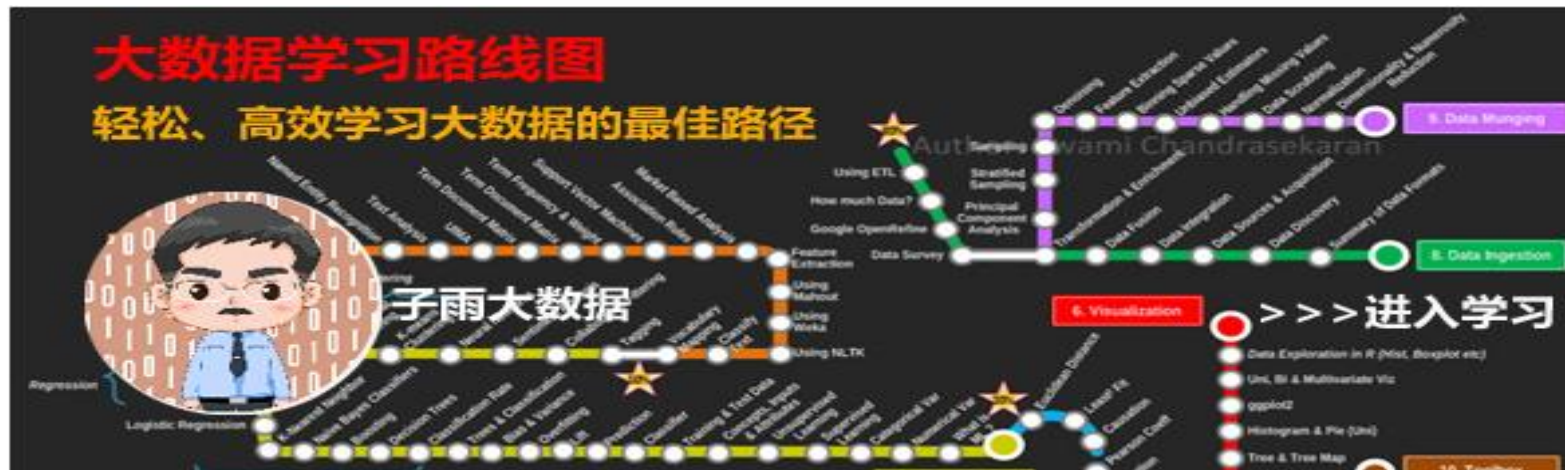


扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），全国高校知名大数据教师，现为厦门大学计算机科学系副教授，厦门大学信息学院实验教学中心主任，曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度、2017年度和2020年度厦门大学教学类奖教金获得者，荣获2019年福建省精品在线开放课程、2018年厦门大学高等教育成果特等奖、2018年福建省高等教育教学成果二等奖、2018年国家精品在线开放课程。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学基金青年基金项目(No.61303004)、1项福建省自然科学基金青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过1000万字高价值的研究和教学资料，累计网络访问量超过1000万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过400万次，累计访问量超过1500万次。



附录B：大数据学习路线图



大数据学习路线图访问地址：<http://dblab.xmu.edu.cn/post/10164/>



附录C：林子雨大数据系列教材



林子雨大数据系列教材

用于导论课、专业课、实训课、公共课

了解全部教材信息：<http://dbllab.xmu.edu.cn/post/bigdatabook/>



附录D：《大数据导论（通识课版）》教材

开设全校公共选修课的优质教材



本课程旨在实现以下几个培养目标：

- 引导学生步入大数据时代，积极投身大数据的变革浪潮之中
- 了解大数据概念，培养大数据思维，养成数据安全意识
- 认识大数据伦理，努力使自己的行为符合大数据伦理规范要求
- 熟悉大数据应用，探寻大数据与自己专业的应用结合点
- 激发学生基于大数据的创新创业热情

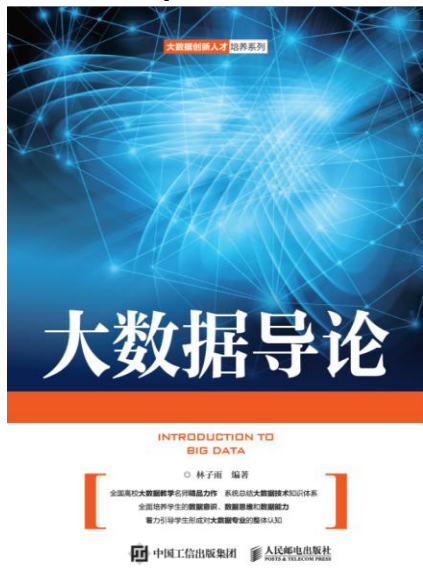
高等教育出版社 ISBN:978-7-04-053577-8 定价：32元 版次：2020年2月第1版
教材官网：<http://dbl原因.xmu.edu.cn/post/bigdataintroduction/>



附录E：《大数据导论》教材

- 林子雨 编著《大数据导论》
- 人民邮电出版社，2020年9月第1版
- ISBN:978-7-115-54446-9 定价：49.80元

教材官网：<http://dblab.xmu.edu.cn/post/bigdata-introduction/>



开设大数据专业导论课的优质教材



扫一扫访问教材官网



附录F：《大数据技术原理与应用（第3版）》教材

《大数据技术原理与应用——概念、存储、处理、分析与应用（第3版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社 ISBN:978-7-115-54405-6 定价：59.80元

全书共有17章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、Flink、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase、MapReduce、Spark和Flink等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

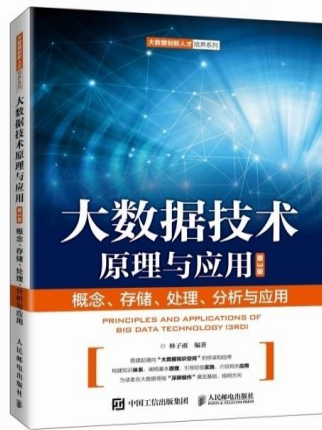
本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：

<http://dbllab.xmu.edu.cn/post/bigdata3>



扫一扫访问教材官网





附录G：《大数据基础编程、实验和案例教程（第2版）》

本书是与《大数据技术原理与应用（第3版）》教材配套的唯一指定实验指导书

大数据教材



1+1黄金组合
厦门大学林子雨编著

配套实验指导书



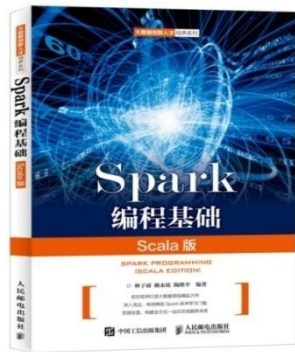
- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，八套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

林子雨编著《大数据基础编程、实验和案例教程（第2版）》

清华大学出版社 ISBN:978-7-302-55977-1 定价：69元 2020年10月第2版



附录H: 《Spark编程基础 (Scala版)》



《Spark编程基础 (Scala版)》

厦门大学 林子雨, 赖永炫, 陶继平 编著

披荆斩棘, 在大数据丛林中开辟学习捷径
填沟削坎, 为快速学习Spark技术铺平道路
深入浅出, 有效降低Spark技术学习门槛
资源全面, 构建全方位一站式在线服务体系

人民邮电出版社出版发行, ISBN:978-7-115-48816-9

教材官网: <http://dbllab.xmu.edu.cn/post/spark/>

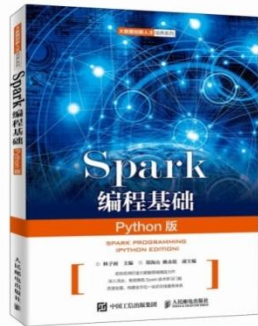


本书以Scala作为开发Spark应用程序的编程语言, 系统介绍了Spark编程的基础知识。全书共8章, 内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作, 以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源, 包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。



附录I: 《Spark编程基础 (Python版)》

《Spark编程基础 (Python版)》



厦门大学 林子雨, 郑海山, 赖永炫 编著

披荆斩棘, 在大数据丛林中开辟学习捷径
填沟削坎, 为快速学习Spark技术铺平道路
深入浅出, 有效降低Spark技术学习门槛
资源全面, 构建全方位一站式在线服务体系



人民邮电出版社出版发行, ISBN:978-7-115-52439-3

教材官网: <http://dbllab.xmu.edu.cn/post/spark-python/>

本书以Python作为开发Spark应用程序的编程语言, 系统介绍了Spark编程的基础知识。全书共8章, 内容包括大数据技术概述、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Structured Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作, 以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源, 包括讲义PPT、习题、源代码、软件、数据集、上机实验指南等。



附录J：高校大数据课程公共服务平台



高校大数据课程

公 共 服 务 平 台

<http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/>



扫一扫访问平台主页



扫一扫观看3分钟FLASH动画宣传片



附录K：高校大数据实训课程系列案例教材

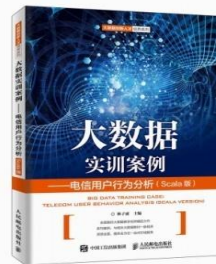
为了更好地满足高校开设大数据实训课程的教材需求，厦门大学数据库实验室林子雨老师团队联合企业共同开发了《高校大数据实训课程系列案例》，目前已经完成开发的系列案例包括：

- 《电影推荐系统》（已经于2019年5月出版）
- 《电信用户行为分析》（已经于2019年5月出版）
- 《实时日志流处理分析》
- 《微博用户情感分析》
- 《互联网广告预测分析》
- 《网站日志处理分析》



系列案例教材将于2019年陆续出版发行，教材相关信息，敬请关注网页后续更新！

<http://dblab.xmu.edu.cn/post/shixunkecheng/>



扫一扫访问大数据实训课程系列案例教材主页

The background is a solid blue color with faint, light-blue silhouettes of people. At the top, there are two groups of people holding hands, suggesting a community or team. On the right side, there is a silhouette of a person standing with their hand on their head. In the bottom left, there are silhouettes of people sitting at a table, possibly in a meeting or classroom setting.

Thank You!

Department of Computer Science, Xiamen University, 2022