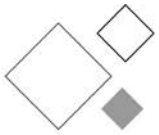

第3章

数据采集

数据采集的渠道包括设备来源、系统导出、网络采集等，通过使用第三方库 Requests 库、Selenium 库，可以对网页数据进行抓取，但采集数据时要注意采集渠道的合法性。本章主要介绍网络公开数据的采集方法。

3.1 爬虫概述

3.1.1 爬虫的基本概念

网络爬虫又称网页蜘蛛、网络机器人，是一种按照一定的规则自动请求互联网网站并提取网络数据的程序或脚本。它可以代替人们自动化浏览网络中的信息，进行数据的采集与整理。它也是一种程序，基本原理是向网站/网络发起请求，获取资源后分析并提取有用数据。

百度、谷歌等人们常用的搜索引擎就属于爬虫的应用，其主要目的是将互联网上的网页下载到本地，从而形成一个互联网内容的镜像备份。百度搜索引擎的爬虫叫作百度蜘蛛(Baiduspider)，360 的爬虫叫作 360Spider，搜狗的爬虫叫作 Sogouspider，必应的爬虫叫作 Bingbot。

3.1.2 爬虫的合法性

1. 法律规范

我国目前并未出台专门针对网络爬虫技术的法律规范，但在司法实践中，相关判决已屡见不鲜。例如，一些人为了获取经济利益，将爬虫作为一种犯罪工具，严重扰乱了计算机信息系统的运行秩序，或者侵害了公民的个人信息，构成了犯罪并触犯刑法。经梳理，在司法实践中，使用爬虫技术的犯罪主要有侵犯公民个人信息罪、非法获取计算机信息系统数据罪、破坏计算机信息系统罪、非法侵入计算机信息系统罪、侵犯著作权罪等。

然而，爬虫本身并没有错，人们应该合法使用爬虫技术来获取数据。判断爬虫合法性边界可以参考以下因素：

(1) 数据是否属于开放数据：数据是否公开不是合法性判断的标准，公开数据不等同于开放数据。如果数据权利方在 Robots 协议或网页中告知了可以爬取的范围及其他应遵

守的义务，则爬取方没有遵守义务时应当承担相应民事责任。

（2）取得数据的手段是否合法：爬虫采用的技术是否突破数据访问控制，法律上是否突破网站或 App 的 Robots 协议。如果突破网站或 App 的 Robots 协议及设置的爬虫检测、加固 Web 站点等限制爬虫的访问权限，则可能会违法，要承担相应的责任。

（3）使用目的是否合法：如果爬虫的目的是实质性替代被爬虫经营者提供的部分产品内容或服务，则会被认为目的不合法。如果爬虫导致经营者增加运营成本，或者破坏系统正常运行，则都属于违法行为。

2. Robots 协议

为了更好地获得推广，同时又能保护网站权益，大部分网站会定义一个 robots.txt 文件作为君子协议。通过 Robots 协议告诉人们哪些页面可以抓取，哪些页面不能抓取。该协议是国际互联网界通行的道德规范，其建立基于以下原则：

（1）搜索技术应服务于人类，同时尊重信息提供者的意愿，并维护其隐私权。

（2）网站有义务保护其使用者的个人信息和隐私不被侵犯。

例如，访问 https://www.qunar.com/robots.txt 去哪儿网的 Robots 协议地址，得到以下结果。

```
# www.qunar.com robots.txt
User-Agent: *
Disallow: /booksystem/
Disallow: /atlas/
Disallow: /twell/flight/
Disallow: /twell2/flight/
Disallow: /suggest/
Disallow: /site/adframe/
Disallow: /site/config/
Disallow: /site/oneway_list.htm
Disallow: /site/roundtrip_list.htm
Disallow: /twell/redirect.jsp
Disallow: /twell/hotel/
Disallow: /twell/scripts/
Disallow: /user/
Disallow: /unsub.htm
Disallow: /*ex_track=*
Disallow: /*bd_source=*
Disallow: /*kwid=*
Disallow: /*cooperate=*
Disallow: /*bdsource=*
```

其中，User-agent 用于描述搜索引擎 robot 的名字。

在 robots.txt 中，至少要有一条 User-agent 记录。如果有多条 User-agent 记录，则说明有多个 robot 会受到该协议的限制。若该项的值设为“*”，则该协议对任何搜索引擎都有效，且这样的记录只能有一条。

Allow 用于描述希望被访问的一组 URL，这个 URL 可以是一条完整的路径，也可以是部分路径。

Disallow 用于描述不希望被访问的一组 URL。任何一条 Disallow 记录为空，都说明该网站的所有部分允许被访问。在 robots.txt 文件中，至少要有一条 Disallow 记录。

访问腾讯网的 Robots 协议地址 https://www.qq.com/robots.txt，得到以下结果。

```
User-agent: *
Disallow:
Sitemap: http://www.qq.com/sitemap_index.xml
```

这里出现了 Sitemap，打开此链接，截取部分结果如图 3-1 所示。

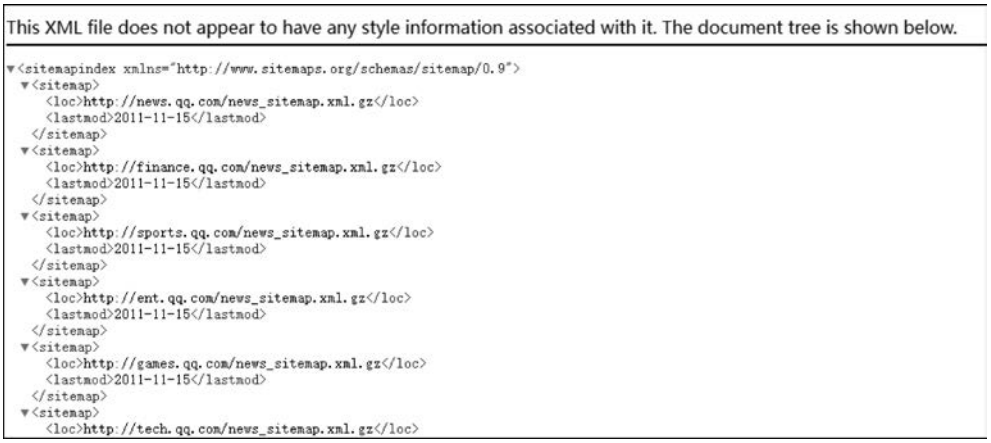

图 3-1　Sitemap 网站地图

在 Sitemap.xml 文件（网站地图）中，列出了网站中的网址和每个网址的其他元数据，如上次更新的时间、更新的频率及相对于网站上其他网址的重要程度等，以便于爬虫可以更加智能地爬取网站。但是，该文件经常会出现缺失或过期的问题，如腾讯 Sitemap 最后更新时间为 2011 年。

3.2　网页与爬虫

爬虫是从网页上爬取数据，因此，了解一些 Web 前端的相关知识是必要的。

3.2.1　URL

爬虫最主要的处理对象就是 URL，它根据 URL 地址取得所需要的文件内容，然后对其进行进一步的处理。因此，准确地理解 URL 对理解网络爬虫至关重要。统一资源定位系统（Uniform Resource Locator，URL）是因特网的万维网服务程序上用于指定信息位置的表示方法，它包含了文件的位置及浏览器处理方式等信息。

URL 的结构为“协议://主机地址（IP 地址或域名）：端口号/路径？参数名＝参数值”，如图 3-2 所示。

在 URL 中，协议通常指超文本传输协议。如果协议头是 ftp，表示提供的是文件传输服务。在域名后，有时还会加上端口号，HTTP 默认端口为 80，HTTPS 默认端口为 443。对于大部分 Web 资源，通常使用 HTTP 或其安全版本 HTTPS。

3.2.2　认识网页结构

要从网页中爬取用户需要的数据，除了需要了解 URL 外，还需要认识网页的结构。网

图 3-2　URL 结构

页一般由 3 部分组成，分别是 HTML(超文本置标语言)、CSS(层叠样式表)和 JavaScript(活动脚本语言)。

例如，打开去哪儿网(https://www.qunar.com/)，按 F12 键可以看到一个网页结构，如图 3-3 所示。在图 3-3 中，上半部分为 HTML 文件，下半部分为 CSS 样式。CSS 是在 HTML 中设置字体、表格、图像等元素样式的语言，它不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化(一般用来定义外观)。用< script >标签的是 JavaScript 代码。JavaScript 是微软开发的专门用于 Web 页面脚本中的语言，它描述了网站中各种交互的内容、功能和特效。

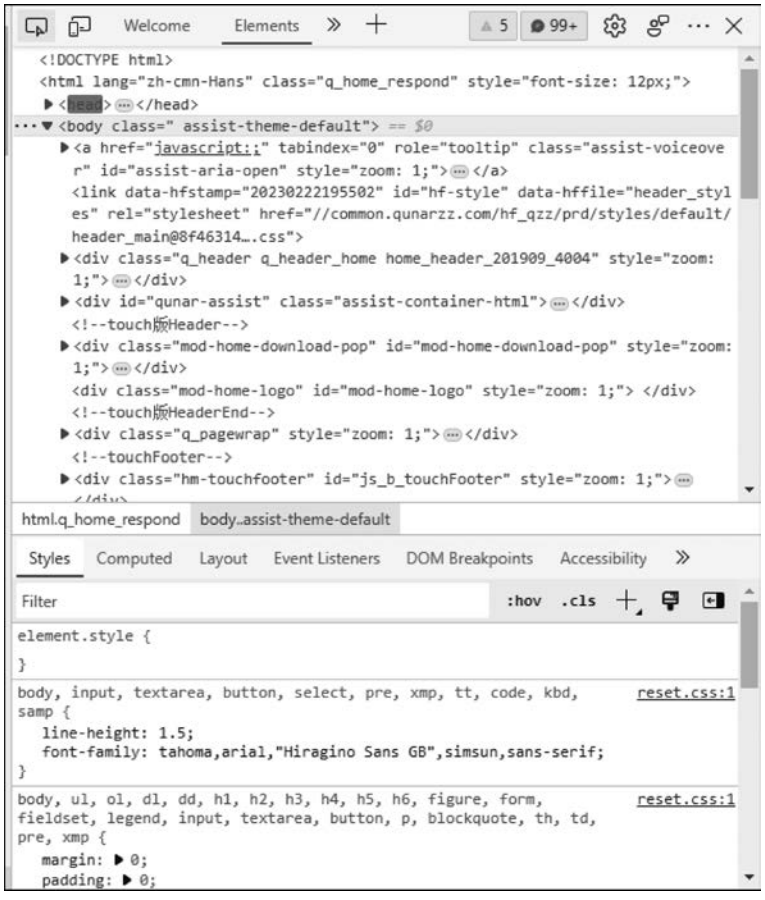

图 3-3　网页结构

HTML 中的常见标记如下。

(1) html 文档标记：<html>…</html>，html 的所有内容需要放置在此标记内。

(2) head 标记：<head>…</head>，head 中的所有内容不直接显示在浏览器中。

(3) 文档标题：<title>…</title>，head 中的标签显示在浏览器的标题栏中。

(4) body 标记：<body>…</body>，body 中的所有内容显示在浏览器中。

(5) 标题标记：<h1>…</h1>(h1～h6)，其中 h1 级文字最大，h6 级文字最小。

(6) 段落标记： … 。

(7) 超链接：链接文本，用于设置网页中的超链接，href 属性指明被超链接的文件地址。用于表示超链接的文本一般显示为蓝色并加下画线。在浏览器中，当鼠标指针指向该文本时，箭头变为手形，并在浏览器的状态栏中显示该链接的地址。

(8) 图像标记：，用于将图片插入网页中，设置图片的大小及相邻文字的排列方式。

(9) 表格标记：<table><tr><td>…</td></tr></table>(table 中可包含多个 tr，td)，tr 表示行，td 表示列。

(10) 换行标记：
，此标记可以强制文本换行，为单标记。

(11) 水平线标记：<hr/>，用于在网页中插入一条水平线。

(12) 字体标记：…。

(13) 字形标记：字形标记用于设置文字的粗体、斜体、下画线、上标、下标等。…表示粗体，…表示斜体，<u>…</u>表示下画线，[…]表示上标，_…表示下标。

(14) 表单标记：<form>…</form>，在 HTML 中与用户进行交互的重要标记。

(15) 输入框标记：<input type="" />，其中，type 可以为 text、password、button、submit、reset、radio、checkbox 和 hidden，分别表示文本框、密码框、按钮、提交按钮、重置按钮、单选框、复选框和隐藏文本域。

(16) 选择标记：<select><option id="">…</option></select>(其中 option 标签可以出现多次)。

(17) 文本区域：<textarea>…</textarea>。

(18) 框架：<frameset>…</frameset>，<frame>…</frame>，<iframe>…</iframe>。

3.2.3 爬虫实现过程

1. HTTP 网页请求过程

HTTP 采用了请求/响应模型。网页请求的具体过程分为以下 4 个步骤。

(1) 浏览器向 DNS 服务器发起 IP 地址请求。

DNS 是域名解析系统，可以将用户输入的域名转换为服务器的 IP 地址。

(2) 浏览器从 DNS 处获得 IP 地址。

(3) 浏览器向服务器发送 Request(请求)。

每一个用户打开的网页都必须在最开始由用户向服务器发送访问的请求。请求信息由

请求行、请求头部、空行和请求数据 4 部分组成。在请求行中，包含了请求方法、URL 地址和协议版本。

HTTP 1.0 定义了 3 种请求方法：GET、POST 和 HEAD。HTTP 1.1 新增了 5 种请求方法：OPTIONS、PUT、DELETE、TRACE 和 CONNECT。

在上述请求方法中，GET 用于请求指定的页面信息并返回实体主体，响应速度快，是最常见的方法。POST 比 GET 多了表单形式上传参数的功能，除查询信息外，还可以修改信息。GET 提交的数据会放在 URL 之后（即请求行中），以“?”分割 URL 和传输数据，参数之间以“&”相连；POST 方法将提交的数据放在 HTTP 包的请求体中。GET 提交的数据大小有限制（因为浏览器对 URL 的长度有限制），而 POST 方法提交的数据没有限制。

（4）服务器 Response（响应）。

服务器在接收到用户的请求后，会验证请求的有效性，然后向用户发送相应的内容。客户端接收到服务器的相应内容后，再将此内容展示出来，以供用户浏览。HTTP 响应报文由状态行、响应报头、空行和响应正文组成。响应行表示协议/协议版本号、响应状态码和状态描述，响应头表示服务器的属性，响应体表示服务器向客户端响应返回的结果（JPG/HTML/JSON/TXT 等）。查看去哪儿网首页输入框的请求响应过程如图 3-4 所示。

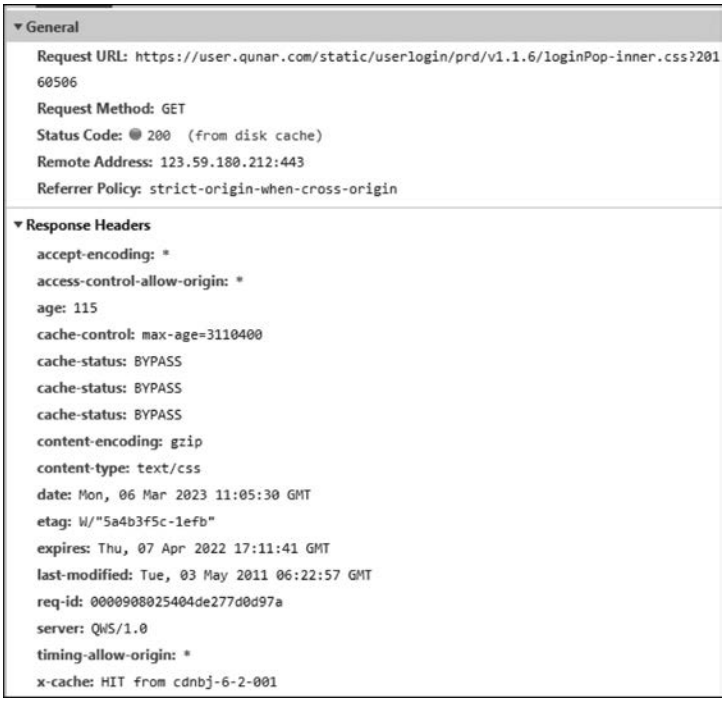

图 3-4　网页请求响应

响应状态码一般由 3 位数字组成，标志着服务器对客户端请求的处理结果。常见的状态码有 200、303、404 等。在本例中，200 表示请求成功，303 表示可通过另一个 URI 找到该请求的响应，404 表示服务器找不到请求的网页，503 表示服务器目前无法使用（由于超载或停机维护）。

2. 爬虫过程

网络爬虫主要的操作对象是 HTTP 请求（Request）和 HTTP 响应（Response）。用户

使用爬虫获取网页数据一般要经过以下 4 步：发送请求、获取响应内容、解析内容和保存数据，如图 3-5 所示。

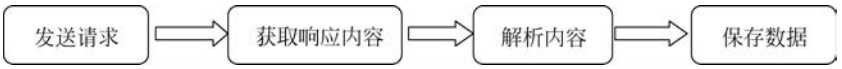

图 3-5　爬虫使用步骤

3.3　Requests 库

Python 凭借其丰富的爬虫框架和强大的多线程处理能力，在爬取网络数据中应用广泛。为了能够访问网络资源，Python 自带了 urllib 模块，但由于 urllib 库的代码编写较为烦琐，目前已经基本被 Requests 库替代。Requests 库采用 Python 语言编写，采用 Apache2 Licensed 开源协议，是基于 urllib 的第三方库。它使用起来更加人性化、更为方便，可以节约大量的工作。

3.3.1　Requests 库的安装

1. 命令行方式

打开 Windows 的命令行或 PyCharm 的 Terminal 终端，输入如下命令：

```
pip install requests
```

2. 菜单方式

启动 PyCharm，单击 File 菜单，选择 Settings 选项，进入 Python Interpreter 的设置界面，如图 3-6 所示。

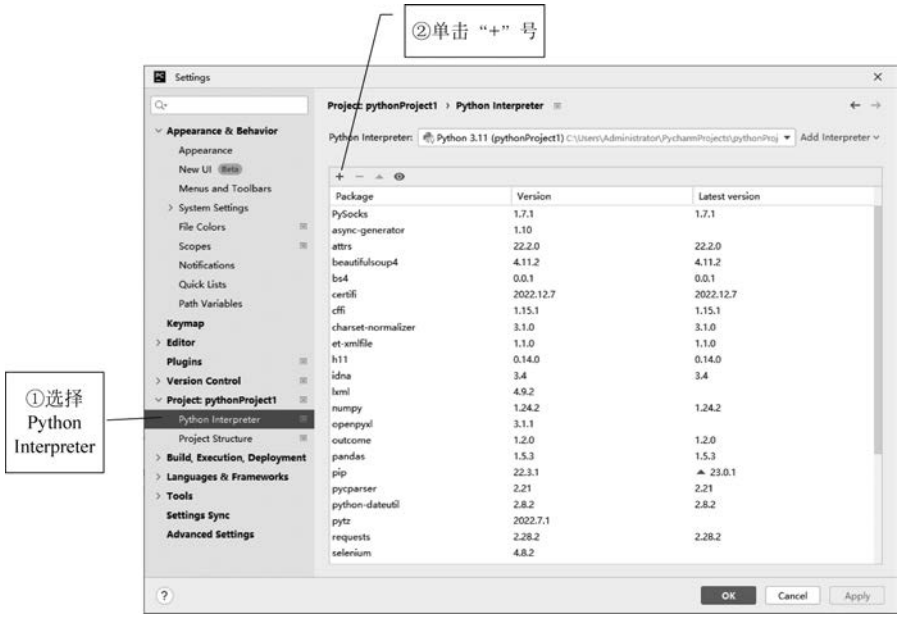

图 3-6　Settings 界面

单击“＋”号后，打开的界面如图 3-7 所示。在输入框中输入 requests，单击 Install Package 按钮，即可自动下载安装 Requests 库。使用 pip 命令行安装第三方库时，有时会由于诸多原因出现安装失败，对于初学者来说使用菜单操作较为方便。

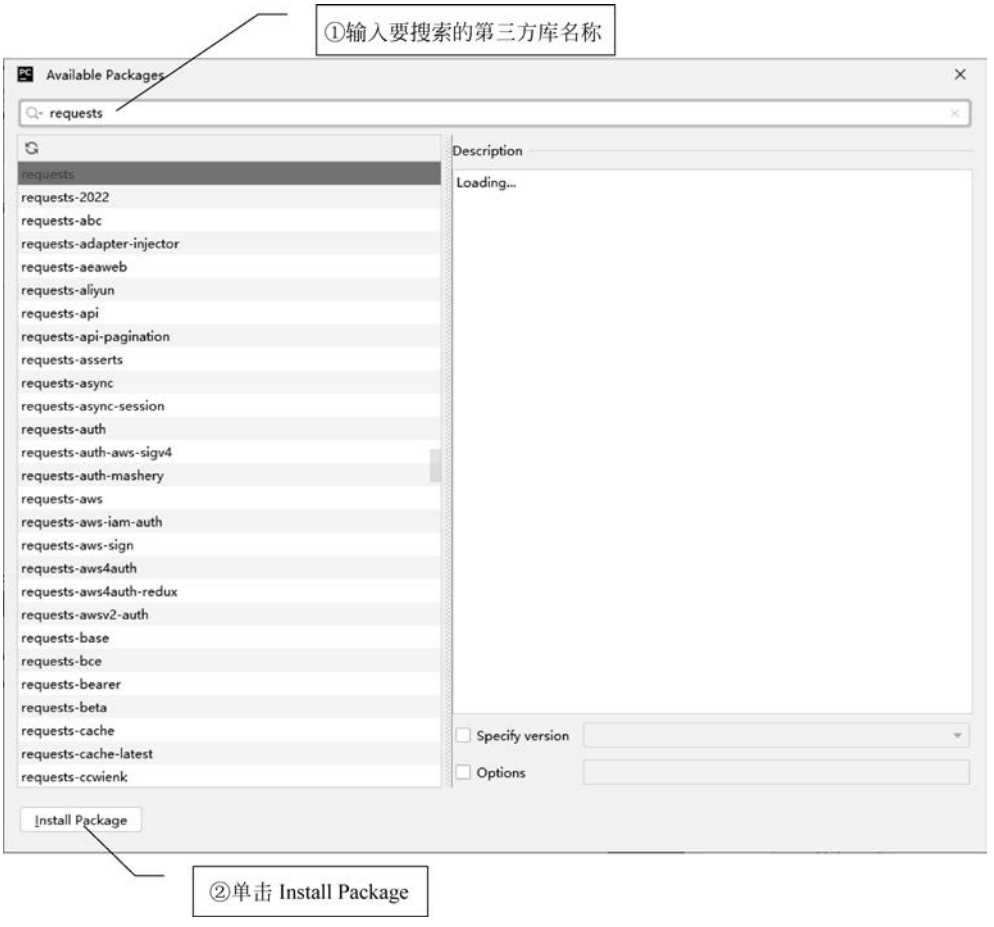

图 3-7　第三方库的搜索安装界面

3.3.2　Requests 库的功能介绍

1. 常用类

Requests 库提供了以下 3 个常用的类。

（1）requests.Request：表示请求对象，一旦请求发送完毕，该请求包含的内容就会被释放掉。

（2）requests.Response：表示响应对象，其中包含服务器对 HTTP 请求的响应。

（3）requests.Session：表示请求对话，提供 Cookie 持久性、连接池和配置。

2. 请求函数

Requests 库提供了很多发送 HTTP 请求的函数，如表 3-1 所示。函数的功能是构建一个 Requests 类型的对象，该对象将被发送到某个服务器上，以请求或查询一些资源；在得到服务器返回的响应时，产生一个 Response 对象，该对象包含了服务器返回的所有信息，也包括原来创建的 Requests 对象。

表 3-1 Requests 库的请求函数

函　数	功 能 说 明
requests.request()	构造一个请求，支撑以下各方面的基础方法
requests.get()	获取 HTML 网页的主要方法，对应于 HTTP 的 GET 请求方式
requests.head()	获取 HTML 网页头信息的方法，对应于 HTTP 的 HEAD 请求方式
requests.post()	向 HTML 网页提交 POST 请求的方法，对应于 HTTP 的 POST 请求方式
requests.put()	向 HTML 网页提交 PUT 请求的方法，对应于 HTTP 的 PUT 请求方式
requests.patch()	向 HTML 网页提交局部修改请求，对应于 HTTP 的 PATCH 请求方式
requests.delete()	向 HTML 网页提交删除请求，对应于 HTTP 的 DELETE 请求方式

3. 响应属性

Response 库获取的服务器响应属性如表 3-2 所示。

表 3-2 服务器响应属性

属　性	说　明
status_code	HTTP 请求地返回状态，200 表示连接成功，404 表示失败
text	HTTP 响应内容的字符串形式，即 URL 对应的页面内容
encoding	从 HTTP 请求头中猜测的响应内容编码方式
apparent_encoding	从内容中分析出的响应编码的方式(备选编码方式)
content	HTTP 响应内容的二进制形式

3.3.3 用 Requests 爬取旅游网站数据

1. 去哪儿网首页信息爬取

打开去哪儿网站，复制首页链接，编写代码如下。

```
import requests
r = requests.get("https://www.qunar.com/", timeout = 1)
print(r.text)
```

上述代码语句含义如下。

(1) 第 1 行：表示导入 Requests 库。

(2) 第 2 行：访问去哪儿网首页，将获取的数据保存到变量 r 中，并设置响应时间为 1s。如果服务器在 1s 内没有应答，将会引发一个异常。

(3) 第 3 行：输出网页源代码。

运行程序如图 3-8 所示。

2. 12306 旅行车次爬取

访问 12306 网站，查找“北京—上海”的火车，编写代码如下。

```
import requests
res = requests.get(https://kyfw.12306.cn/otn/leftTicket/init?linktypeid = dc&fs = %E4%B8%8A%E6%B5%B7,SHH&ts = %E5%8C%97%E4%BA%AC,BJP&date = 2023 - 03 - 10&flag = N,N,Y')
print(res.text)
```

上述代码语句含义如下。

(1) 第 1 行：导入 Requests 库。

图 3-8　去哪儿网首页爬取结果

(2) 第 2 行：请求 URL 路径并发送 GET 请求，返回一个响应对象。

(3) 第 3 行：查看响应的内容。

运行结果如图 3-9 所示。

图 3-9　12306 车次爬取结果

根据上述代码可以查看到当前网页中所有的 HTML 信息，这为下一步的数据解析做好了准备。

3. 大众点评商品评价爬取

找到大众点评网页中的一个页面，复制网页链接，编写代码如下。

```
import requests
url = r'https://www.dianping.com/chongqing/ch10/g111'
res = requests.get(url)
print(res.text)
```

得到的结果如下。

```
<html>
<head><title>403 Forbidden</title></head>
<body>
```

```
<center><h1>403 Forbidden</h1></center>
<hr><center>openresty</center>
</body>
</html>
```

可以看到服务器出现403禁止访问，这是因为有些网站会设置反爬机制，对于非浏览器的访问拒绝响应。这时，需要在爬虫程序中修改请求的headers伪装浏览器访问，或者使用代理发起请求。网页的headers中声明了HTML事务中的操作参数，如Accept、Accept-Encoding、Connection、Host、Cookies和User-Agent等。在定义headers时，要使用到以上参数。因此，需要在代码中增加headers信息，可以通过开发者工具的network找到headers信息。此外，由于网站需要登录后才能查看数据，因此需要Cookies的信息，修改后的代码如下。

```
import requests
url = r'https://www.dianping.com/chongqing/ch10/g111'
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/109.0.0.0 Safari/537.36",
    "Connection":"keep-alive",
    "Host":"www.dianping.com",
    "Referer": "https://www.dianping.com/",          "cookies":"fspop=test;
    _lxsdk_cuid=186c5dcbd0ec8-03b48a23989918-26021051-1fa400-186c5dcbd0fc8;
    _lxsdk=186c5dcbd0ec8-03b48a23989918-26021051-1fa400-186c5dcbd0fc8;
    _hc.v=d2da9947-d324-e4c9-e38d-286a7e33bb2c.1678356823; s_ViewType=10;
    WEBDFPID=y82408vuzz06523zy118632yv7341z2y813x4wxv89w97958014216z3-1993716832696-
1678356831651EUQGICMfd79fef3d01d5e9aadc18ccd4d0c95073316;
    dplet=48528c5b10fbe8754d2a861ef68815b7; ua=QI_370172907;
    ctu=d227a367436b71dc3f00c90d3f45843240157ef264ad13013ad755480fa18cb0; cy=9;
    cye=chongqing; _lx_utm=utm_source=Baidu&utm_medium=organic;
    Hm_lvt_602b80cf8079ae6591966cc70a3940e7=1678356824,1678426556;
    dper=8fe93b0b326b400e0838da9fce9d4ef76da64a9636f7267b99990c421af52f8d46167ef274721
f603c719b40e476bc93fe538945afd4d76104158848188cd2f5;
    qruuid=24d2b88b-8613-4312-9f6c-25bf33b77443; ll=7fd06e815b796be3df069dec7836c3df;
Hm_lpvt_602b80cf8079ae6591966cc70a3940e7=1678426676;
    _lxsdk_s=186ca04c48f-950-e97-f1d||48"
}
res=requests.get(url,headers=headers)
res.encoding=res.apparent_encoding
print(res.text)
```

运行结果如图3-10所示，可以看到当前网页中的HTML源码已经被获取到。

图3-10　大众点评网页代码

3.4 Selenium 抓取动态页面

3.4.1 Selenium 概述

静态数据是服务器已经渲染好的内容，由浏览器直接解释执行。随着 Web 前端技术的发展，JavaScript 技术应用越来越广泛，有交互性的动态网页更能满足网站的需求。如何判断一个网页是静态还是动态？最简单的方法是，如果单击下一页地址栏没有发生任何变化，则说明是动态网页。

Selenium 是支持 Web 浏览器自动化的一系列工具和库的综合项目，其核心是 WebDriver，利用它可以驱动 Firefox、Chrome、IE 等浏览器执行特定的动作，如单击、下拉等操作，同时还可以获取浏览器当前呈现页面的源代码，从而做到可见即可爬。对于一些 JavaScript 动态渲染的页面，此种抓取方式非常有效。

3.4.2 Selenium 的安装

1. 安装 Selenium

为了在 Python 中使用 Selenium，需要安装 Selenium 包，本书编写时使用的版本 4.8.2。

(1) 命令行方式。

打开 Windows 的命令行或 PyCharm 的 Terminal 终端，输入命令如下。

```
pip install selenium == 4.8.2
```

(2) 菜单方式。

安装方法可参照 3.3.1 节第三方库的安装。

2. 安装驱动 geckodriver

浏览器的驱动文件至关重要，没有驱动文件，在程序中将无法调取浏览器。根据浏览器的不同，可以在对应官网或开源平台下载对应的驱动。

Chrome 浏览器的驱动文件为 chromedriver. exe，Firefox 浏览器的驱动文件为 geckodriver. exe，Edge 浏览器的驱动文件为 msedgedriver. exe。

以 Firefox 浏览器为例说明具体安装过程。

(1) 下载 Firefox 的 driver 驱动。

GitHub 是一个由微软开发的面向开源及私有软件项目的托管平台，提供了大量的开源代码。打开 GitHub 网站，根据所用 Firefox 的版本，找到对应的驱动文件下载。

(2) 解压下载好的 geckodriver. zip。

(3) 将解压完成的 geckodriver. exe 放入 Firefox 安装包的根目录下。

(4) 配置环境变量。编辑 path，将 Firefox 安装包的路径写入。

(5) 找到 Python 的安装包，将 geckodriver. exe 复制到 python. exe 的同级目录下。

3.4.3 Selenium 的基本用法

1. 调用浏览器

在 PyCharm 中运行以下代码，代码将启动浏览器并打开百度首页。

```
from selenium import webdriver
#打开 Firefox 浏览器
browser = webdriver.Firefox()
#打开 Chrome 浏览器
browser = webdriver.Chrome()
#打开 Edge 浏览器
browser = webdriver.Edge()
#获取百度首页地址
browser.get("https://www.baidu.com")
#关闭浏览器
browser.quit()
```

2. 元素定位

元素定位方法包含了以下两个系列。

(1) find_element()系列：用于定位单个页面元素。

(2) find_elements()系列：用于定位一组页面元素，并获取一组列表。

在调用页面元素时，需要在代码前导入 By 模块，代码如下。

```
from selenium import webdriver
from selenium.webdriver.common.by import By
```

元素定位的相关操作如下。

(1) 通过 ID 属性定位。

基本格式：find_element(By.ID,'XX')，表示根据元素的 ID 属性定位。

(2) 通过 name 属性定位。

基本格式：find_element(By.NAME,'XX')，表示根据元素的 name 值定位。

(3) 通过 class 定位。

基本格式：find_element_by(By.CLASS_NAME,'XX')，表示根据元素的 class 属性定位，但可能受 JavaScript 影响动态变化。

(4) 通过 tag 定位。

基本格式：find_element(By.TAG_NAME,'XX')，表示根据元素的标签名定位。

(5) 通过 link 定位。

link 表示含有属性 href 的标签元素，如< a href="https://www.csdn.net"> linktext ，可以通过 LINK-TEXT 进行定位。

基本格式：find_element(By.LINK_TEXT,'XX')，表示链接文本全匹配进行精确定位。

基本格式：find_element(By.PARTIAL_LINK_TEXT,'XX')，表示链接文本模糊匹配进行定位。

(6) 通过 XPath 定位。

XPath 是一种在 XML 文档中定位元素的语言，其详细内容将在第 4 章进行介绍。

基本格式：find_element(By.XPATH,'XX')，表示根据元素的 XPath 表达式定位，可以准确定位任何元素。

(7) 通过 CSS 选择器定位。

基本格式：find_element(By.CSS_SELECTOR,'XX')，表示根据元素的 CSS 选择器定

位，可以准确定位任何元素。

(8) 文本输入、清空与提交。

send_keys('XXX')表示文本输入，clear()表示文本清空，submit()表示文本提交。

(9) 获取页面内容。

title 表示页面标题，page_source 表示页面源码，current_url 表示当前页面链接，text 表示标签内文本。

例如，调用 Firefox 访问百度首页，并输出页面标题、页面链接，代码如下。

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
browser = webdriver.Firefox()
browser.get("http://www.baidu.com")
#获取标题
title = browser.title
#输出标题
print(title)
#获取源代码
page_code = browser.page_source
#获取页面链接
url = browser.current_url
#输出页面链接
print(url)
#等待 3s,关闭页面
time.sleep(3)
browser.quit()
```

(10) 等待。

等待有以下 3 种方法。

① sleep(*N*)。强制等待，需要导入 time 包，*N* 表示等待时长。该方法常用于避免因元素未加载出来而定位失败的情况。

② WebDriverWait(browser，*N*，*H*)。显式等待，browser 表示浏览器对象，*N* 是等待时长，*H* 是频率。显式等待只针对指定的元素生效，可以根据需要定位的元素来设置显式等待。

③ implicitly_wait(*N*)。隐式等待，*N* 表示页面元素加载的最大等待时长。如果加载到需要的内容，就结束等待，执行下一步操作；如果超出最大等待时间元素仍未加载，就抛出异常。

(11) 调整浏览器窗口尺寸。

maximize_window()表示窗口最大化，minimize_window()表示窗口最小化，set_window_size(width，height)表示调整窗口为指定尺寸。

(12) 前进后退。

forward()表示前进一页，back()表示后退一页。

(13) 页面刷新。

refresh()表示页面刷新。

(14) 窗口切换。

① current_window_handle：表示获取当前窗口的句柄。

② window_handles：表示获取所有打开页面的句柄，形式为列表。

③ switch_to.window("XX")：表示切换到指定页面，XX 代表页面句柄。

④ switch_to.frame(XX)：表示切换到内敛框架页面，XX 代表内敛框架标签的定位对象。

⑤ swith_to.parent_frame()：表示切回到内敛框架的上一级，即从内敛框架切出。

⑥ switch_to.alert：表示切换到页面弹窗。

(15) 获取标签元素的属性值。

get_attribute("XX")：表示获取标签属性值，XX 为标签属性名。

(16) 下拉列表操作。

下拉列表操作涉及的类为 Select 类，需要在程序前输入以下代码导入模块。

```
from selenium.webdriver.support.select import Select
```

在使用 Select 类时，需要用到以下方法。

① Select("XX")：表示判断标签元素 XX 是否为下拉列表元素。

② select_by_value("XX")：表示通过下拉列表 value 的值 XX 来选择选项。

③ select_by_visible_text("XX")：表示通过下拉列表中的文本内容 XX 来选择选项。

④ select_by_index(*N*)或 options[*N*].click()：表示通过下拉列表索引号 *N* 来选择选项，从 0 开始计算。

⑤ Options：表示下拉列表内的 options 标签。

(17) 弹窗操作。

switch_to.alert 表示获取弹窗对象，text 表示弹窗内容，accept()表示接受弹窗，dismiss()表示取消弹窗。

(18) 鼠标操作。

鼠标操作涉及的类为 ActionChains 类，需要在程序前输入以下代码导入模块。

```
from selenium.webdriver import ActionChains
```

在使用 ActionChains 类时，需要用到以下方法。

① move_to_element(*X*)：表示鼠标悬停，*X* 表示定位到的标签。

② double_click(*X*)：表示双击。

③ context_click(*X*)：表示右击。

④ perform()：表示执行所有存储在 ActionChains()类中的行为，做最终的提交。

(19) 键盘操作。

键盘操作涉及的类为 Keys 类，需要在程序前输入以下代码导入模块。

```
from selenium.webdriver import Keys
```

在使用 keys 类时，需要用到以下方法。

send_keys(Keys.BACK_SPACE)表示执行回退键，Backspace.send_keys(Keys.CONTROL,'a')表示全选，send_keys(Keys.CONTROL,'x')表示剪切，send_keys(Keys.CONTROL,'c')表示复制，send_keys(Keys.CONTROL,'v')表示粘贴。

(20) JavaScript 代码执行。

execute_script(*X*),表示执行 JavaScript 代码,*X* 是要执行的 JavaScript 代码。

(21) 窗口截图。

基本格式:get_screenshot_as_file(XX),表示浏览器窗口截屏,XX 是文件保存地址、文件名及格式。

3.4.4　用 Selenium 爬取旅游网站数据

1. 在 12306 爬取北京旅游信息

实现效果:调用 Firefox 浏览器打开 12306 的旅游子页面,在搜索框中输入“北京”,然后关闭浏览器。

需要提前做好的准备工作如下。

(1) 确定目标页面 URL。

在浏览器的地址栏中,复制 12306 的旅游页面的 URL。

(2) 搜索框的 ID 值。

在浏览器中按 F12 键进入开发者工具,单击 图标,再单击网页中要定位的搜索框。在开发者界面中会看到搜索框对应的 HTML 代码,复制 ID 的值“commom_keyword”,为程序代码的编写做好准备。开发者工具界面如图 3-11 所示。

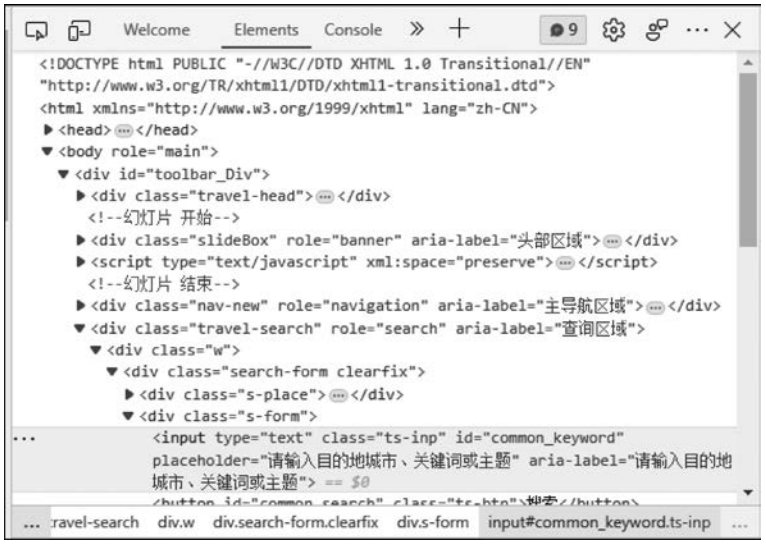

图 3-11　开发者工具界面

(3) 搜索按钮的 ID 值。

搜索按钮的 ID 值的查找方式与搜索框相同。

做好以上准备后,在 PyCharm 中编写代码如下。

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.support.wait import WebDriverWait
```

```
from selenium.webdriver.support import expected_conditions as EC
#调取 Firefox 浏览器
browser = webdriver.Firefox()
try:
    #获取 URL
    browser.get('https://travel.12306.cn/portal/travel/list')
    #查找搜索框的 ID 元素并赋值
    input = browser.find_element(By.ID,'common_keyword')
    #在搜索框中输入"北京"
    input.send_keys('北京')
    click = browser.find_element(By.ID, 'common_search')
    #等待 2s
    time.sleep(2)
    #单击搜索键
    click.send_keys(Keys.ENTER)
    wait = WebDriverWait(browser,20)
    wait.until(EC.presence_of_element_located((By.ID,'common_keyword')))
    #输出当前页面的 URL
    print(browser.current_url)
    #输出当前页面的 Cookies
    print(browser.get_cookies())
    #输出当前页面的页面资源
    print(browser.page_source)
finally:
    browser.close()
```

运行代码,在浏览器中可查到的界面如图 3-12 所示。

图 3-12　12306“北京”旅游信息

2. 爬取景点点评

实现效果：访问携程网上的重庆欢乐谷景点,输出前 10 页点评,点评内容包括用户名、评分、评语、点评时间和 IP 地址。

需要提前做好的准备工作如下。

（1）确定目标页面 URL。

在浏览器的地址栏中，打开携程网上的重庆欢乐谷景点页面，复制 URL 链接 https://you.ctrip.com/sight/chongqing158/2486251.html。

（2）选择翻页方法。

由于本例中要输出 10 页的点评内容，因此需要观察每个页面的 URL。如果每个页面的 URL 不同，则需要分析 URL 链接的规律，并用 for 循环获取页面地址；如果切换页面后发现 URL 地址没有变化，则需要通过鼠标选择下一页，以此获取后续页的内容。本案例属于后者，故选择用鼠标进行切换。

（3）查找 XPath。

按 F12 键进入开发者工具，单击网页中要定位的元素，在开发者界面中找到与其对应的 HTML 代码，右击该代码可以复制 XPath 路径。

做好以上准备后，在 PyCharm 中编写代码如下。

```
import time
from selenium import webdriver
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
from lxml import etree
base_url = r'https://you.ctrip.com/sight/chongqing158/2486251.html'
#打开浏览器处理
browser = webdriver.Firefox()
browser.get(base_url)
print("正在获取数据 …… 请稍等 …… ")
time.sleep(4)
#循环获取 10 页评价
for x in range(10):
    print("第{0}页数据加载中，请稍等 …… ".format(x + 1))
    time.sleep(5)
    print('--------- 正在获取第{0}页数据 --------- '.format(x + 1))
    url1 = browser.current_url
    res = browser.page_source
    html = etree.HTML(res)
    #评价用户名
    result1 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/div[4]/
div/div[5]/div/div[1]/div[2]/text()')
    #评分
    result2 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/div[4]/
div/div[5]/div/div[2]/div[1]/span/text()')
    #评价语
    result3 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/div[4]/
div/div[5]/div/div[2]/div[2]/text()')
    #评价图片(当图片变化时，评价时间和评价 IP 归属地 XPath 有变化，需要单独处理)
    l1 = []
    l2 = []
    l3 = []
    for y in range(1,11):
```

```
        #评价图片
        result4 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/div
[4]/div/div[5]/div[{0}]/div[2]/div[3]/a/@href'.format(y))
        if len(result4) == 0:
            #评价时间
            result5 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/
div[4]/div/div[5]/div[{0}]/div[2]/div[3]/div[1]/text()'.format(y))
            #评价 IP 归属地
            result6 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/
div[4]/div/div[5]/div[{0}]/div[2]/div[3]/div[1]/span/text()'.format(y))
            l2.append(result5[0])
            l3.append(result6[0])
            l3.append(result6[1])
        else:
            l1.append(result4)
            #评价时间及 IP 归属地
            result5 = html.xpath('/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/
div[4]/div/div[5]/div[{0}]/div[2]/div[4]/div[1]/text()'.format(y))
            result6 = html.xpath( '/html/body/div[2]/div[2]/div/div[3]/div/div[4]/div[1]/
div[4]/div/div[5]/div[{0}]/div[2]/div[4]/div[1]/span/text()'.format(y))
            l2.append(result5[0])
            l3.append(result6[0])
            l3.append(result6[1])
    print('用户名:',result1)
    print('评分:',result2)
    print('评语:',result3)
    print('图片链接:',l1)
    print('评价时间:',l2)
    print('IP 归属地:',l3)
    #换页操作
    #获取底部下一页
    canzhao = browser.find_elements(By.CLASS_NAME, 'seotitle1')
    nextpage = browser.find_elements(By.CLASS_NAME,'ant - pagination - item - comment')
    time.sleep(2)
    #移动到元素 element 对象的"顶端",与当前窗口的"D 底部"对齐
    browser.execute_script("arguments[0].scrollIntoView(false);", canzhao[0])
    time.sleep(2)
    #鼠标移至下一页
    ActionChains(browser).move_to_element(nextpage[1]).perform()
    time.sleep(2)
    #鼠标单击下一页
    nextpage[1].click()
    time.sleep(4)
# 数据爬取完成,关闭浏览器
print('--------- 获取数据完成 --------- ')
time.sleep(4)
browser.close()
```

部分输出结果如图 3-13 所示。

本案例用到了 XPath,具体使用原理将在第 4 章进行详细介绍。

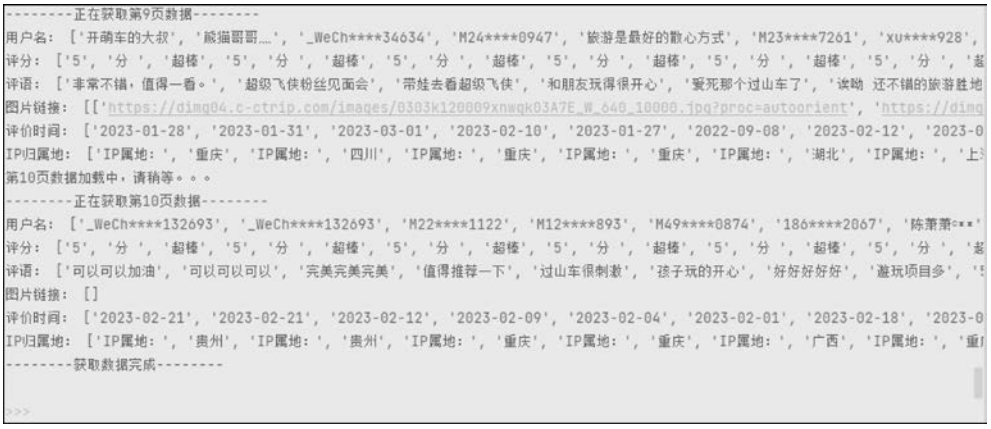
--------正在获取第9页数据--------
用户名：['开萌车的大叔', '戴猫哥哥…', '_WeCh****34634', 'M24****0947', '旅游是最好的散心方式', 'M23****7261', 'xu****928',
评分：['5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超
评语：['非常不错，值得一看。', '超级飞侠粉丝见面会', '带娃去看超级飞侠', '和朋友玩得很开心', '爱死那个过山车了', '诶呦 还不错的旅游胜地
图片链接：[['https://dimg04.c-ctrip.com/images/0303k120009xnwqk03A7E_W_640_10000.jpg?proc=autoorient', 'https://dimg
评价时间：['2023-01-28', '2023-01-31', '2023-03-01', '2023-02-10', '2023-01-27', '2022-09-08', '2023-02-12', '2023-0
IP归属地：['IP属地：', '重庆', 'IP属地：', '四川', 'IP属地：', '重庆', 'IP属地：', '重庆', 'IP属地：', '湖北', 'IP属地：', '上
第10页数据加载中，请稍等。。。
--------正在获取第10页数据--------
用户名：['_WeCh****132693', '_WeCh****132693', 'M22****1122', 'M12****893', 'M49****0874', '186****2067', '陈萧萧***'
评分：['5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超棒', '5', '分 ', '超
评语：['可以可以加油', '可以可以可以', '完美完美完美', '值得推荐一下', '过山车很刺激', '孩子玩的开心', '好好好好好', '遊玩项目多', '5
图片链接：[]
评价时间：['2023-02-21', '2023-02-21', '2023-02-12', '2023-02-09', '2023-02-04', '2023-02-01', '2023-02-18', '2023-0
IP归属地：['IP属地：', '贵州', 'IP属地：', '贵州', 'IP属地：', '重庆', 'IP属地：', '重庆', 'IP属地：', '广西', 'IP属地：', '重
--------获取数据完成--------
>>>

图 3-13　部分爬取结果