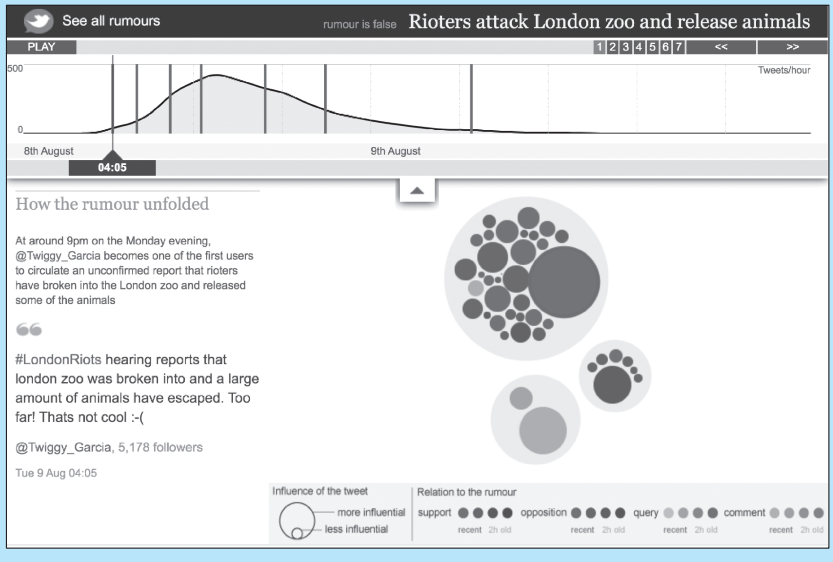
第三章

网络数据获取

数据获取是数据新闻实践至关重要的一步，它关系到数据质量，并最终影响数据呈现。第二章中我们已介绍了通过公开数据、第三方数据、自建数据库、众包等方式获取数据，本章聚焦于网络数据的自行获取与采集。网络数据主要指存储于网络服务器并呈现于网页上的数据，这些数据通常是非结构化的、未经过预先处理的，也包括大量由用户自主生产的数据，比如社交媒体平台数据。随着互联网的发展和数据开放的推进，网络数据成为数据生态中日益重要的组成部分，也是数据新闻生产的重要原材料。

案例档案卡 3-1

- **刊发媒体：**英国《卫报》（*The Guardian*）
- **案例名称：**《骚乱谣言如何在推特上传播》（*How riot rumors spread on Twitter*）
- **案例作者：**《卫报》互动团队，罗伯·普罗克特（Rob Procter）、法里达·维斯（Farida Vis）、埃里克斯·沃斯（Alex Voss）



《骚乱谣言如何在推特上传播》示例

● **案例特点：**这则新闻通过获取社交媒体平台数据，分析了社交媒体平台信息传播的特点，提供了仅靠采访无法获得的洞见。2011 年英国南部爆发骚乱，警方指责推特和脸书等社交媒体平台散播谣言，助长了骚乱的蔓延，一度考虑关闭社交媒体平台。为此《卫报》抓取了推特上的 257 万条推文，与学者合作，分析其中几条典型谣言的传播过程。分析发现，随时间流逝，谣言热度降低，社交媒体平台上亦有大量推文澄清谣言，有助于信息的净化。


扫码播放节选

上述案例是新闻生产中运用社交媒体平台大数据的一次早期尝试。如第一章所述，大数据不仅意味着数据量大，而且主要强调两层更重要的含义：首先，它是数字时代人类生活留下的电子脚印，是日常生活的副产品[①]，即这些数据是伴随着日常生活产生的，借由这些数据可以分析人类的群体性行为[②]。比如交通卡刷卡数据可分析人群流动的规律，以更有效地调配交通资源。其次，大数据打破了数据的垄断和封闭。以往数据大多由政府部门或社会机构采集，比如政府会定期进行人口普查获取人口构成数据，医院进行临床实验自行记录实验结果数据，这些数据往往不为普通人所知，如果不是采用内部合作的方式，很难获取这些数据用于新闻报道。而数字时代人类的行为被电子化，人们的网络发言留下了文本记录，购买商品留下了点击记录，出行活动留下了出行轨迹等记录，这些记录数量巨大，并以电子化的形式存储于网络服务器，从而可被获取。同时，随着互联网的发展以及开放数据运动的推动，也有越来越多的政府部门或社会机构通过网络发布数据，这些数据也需要通过技术方式获取。

网络数据的获取方式主要有三种：一是通过网络高级搜索，快速高效锁定所需数据；二是通过工具软件进行抓取，比如使用八爪鱼、后羿采集器等；三是通过编程语言编写程序抓取。这些数据可能存储于某个数据库中，也可能存储在网络服务器上，并显示在网页中。无论使用何种方式，首先都需要理解网页的数据交互方式。这是因为网络数据的展示和存储是分开的，我们通常访问的网页实际是前端展示部分，这些内容以数据的形式存储于后台服务器。要获取这些数据，就需要认识网页结构，理解网页前端如何展示；随后通过访问、解析网址，对网页提出请求来调用接口，获取后台数据。

第一节　认识网页结构

网页是构成网站的基本要素，前端网页呈现与后端数据存储共同构成了网站。网页是指适用于万维网，能够被浏览器识别的文件，通常由超文本标记语言（HTML，Hyper-text

① 《专访谢宇教授（上）：大数据的重要价值不是“大”》，微信公众号“严肃的人口学八卦”，2018 年 8 月 24 日。

② ［英］维克托·迈尔－舍恩伯格、肯尼迪·库克耶：《大数据时代：生活、工作与思维的大变革》，周涛等译，浙江人民出版社 2013 年版。

Markup Language）创建。网页上包含文本数据、图像数据、超链接等元素。除了 HTML 语言，如需构建样式丰富、功能多样的网页，还要借助 CSS 级联样式表以及 JavaScript 脚本语言。形象地说：HTML 语言搭建起网页基本结构，确立了网页的“骨架”；CSS 级联样式表定义了网页的样式，也即装饰了网页的“皮肤”；JavaScript 脚本规定了网页的行为方式，也即构建了网页的“肌肉”。

这些编程语言共同写就了网页的源代码（Source Code），即一系列人类可读的计算机语言指令。读取源代码，可以获知数据的存储路径，进而通过不同方式获取网络数据。源代码中最核心的是 HTML 代码，形象地说，它比普通文本文件高级，高级之处即在于可对文本文件里的文字进行标记（Markup），比如标记其中一段为列表，另外一段为超链接，这些对普通文本的修饰构成了一套规则，这套规则就是 HTML。[①] 读取源代码有三个步骤：调阅、探索和认识，下面依次介绍。

一、调阅源代码

调阅源代码的方式有以下几种，不同浏览器略有不同。源代码显示后的视图如图 3-1，左侧为原始网页，右侧为源代码显示页。

图 3-1 显示源代码视图

第一，在网页空白处点击鼠标右键，在弹出的对话框中点击“显示网页源代码”或“检查”条目。

第二，点击浏览器菜单栏“视图”，选择“开发者”，随后进入“显示源代码”条目。

第三，点击浏览器隐藏菜单按钮，选择“更多工具”下的“开发者工具”条目。

显示代码后，可根据需要调整源代码区域的布局方式。本章随后的小节中，我们将调

① 陈宇、巩晓波、高杨等：《给产品经理讲技术》，电子工业出版社 2019 年版，第 16—17 页。

用 Python 语言中的 Selenium 库抓取网络数据，此时需要多个浏览器窗口并列同步操作，如此可以观察到代码对浏览器的操作执行过程。

小练习

请登录你所在学校的网站，查阅学校网站主页 logo 的源代码。

二、探索源代码

调阅成功后，便可开始探索网页源代码。点击源代码显示区域左上角的 按钮，图标变为蓝色后，可进行响应式调阅，快速锁定数据对应的代码（见图 3-2）。此时挪动鼠标，网页左侧鼠标悬停区域变为阴影，相应的源代码则显示在右侧，并以阴影标示（见图 3-3）。

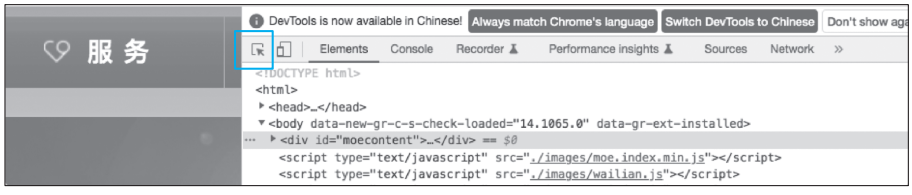


图 3-2　响应式调阅按钮

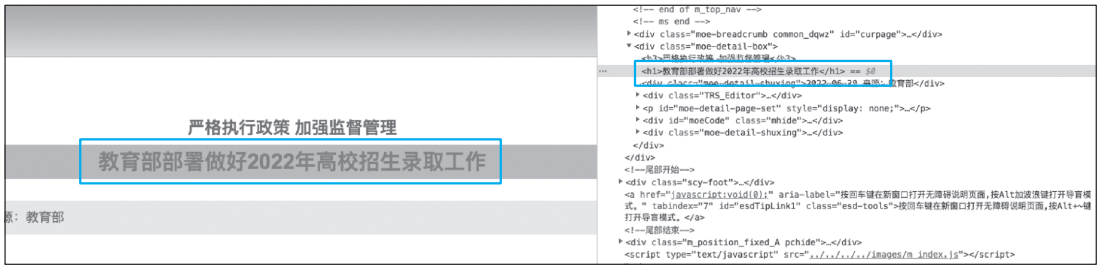


图 3-3　响应式调阅视图

三、认识源代码

通过探索源代码进行响应式调阅后，可锁定需要抓取的内容，确定所用代码。还需要理解这些源代码的含义，才能对其提出请求以获取数据。

这些源代码由 HTML 标记语言构建。简要地说，HTML 标记语言包含标签与元素两部分。标签由尖括号（< >）标识，总是成对出现，分为起始标签和结束标签。通常情况下，起始标签与结束标签内容一致，但需注意结束标签以反斜杠（/）标识。例如，<body> 为起始标签，</body> 为结束标签，如忽略反斜杠则会影响网页兼容性。HTML 语言对大小写不敏感，<BODY> 等同于 <body>，但推荐使用小写。元素指起始标签和结束标签之间的全部代码。比如，<body> 这是一段文字 </body>，两个标签中间的“这是一段文字”部分即为元素。浏览器能够读取 HTML 文档，并以网页的形式呈现。

表 3-1 列出了 HTML 语言常用的标签和元素，理解这些标签和元素的含义有助于认识网页结构，这也是获取网络数据的基础。

表 3-1 HTML 语言常用标签及含义

开始标签	元素内容 / 描述	结束标签
<!DOCTYPE>	声明文档类型，以便浏览器识别	无
<html>	定义 HTML 文档	</html>
<head>	定义文档头部	</head>
<title>	定义文档标题	</title>
<body>	定义文档主体内容	</body>
<h1><h2><h3> <h4><h5><h6>	定义 6 个层级的标题	</h1></h2></h3> </h4></h5></h6>
	定义段落	
<!--…-->	插入注释	无
	向网页中嵌入图像网址	无
	定义超链接	
<link>	定义文档与外部资源的链接	无
	定义无序列表	
	定义有序列表	
	定义列表项目	
<table>	定义 HTML 表格	</table>
<td>	定义单元格内容	</td>
<tr>	定义表格行	</tr>
<th>	定义表头	</th>
	定义文档中一个分隔区域	
	组合文档中的行内元素，以便进行样式设定	
<script>	定义客户端脚本	</script>

学习 HTML 语言最好的方式是实践，一些网站或编辑器可以编辑代码并实时生成网页（见图 3-4，左边为代码，右边为根据代码生成的网页），读者可自行探索。①

图 3-4 HTML 语言在线学习视图

① 示例网站为 w3school，这是一个在线学习编程语言的网站。

了解了 HTML 语言，接下来我们便可以开始学习网络数据的获取了。

第二节　网络高级搜索

本节我们聚焦于网络数据采集的第一种方式——网络高级搜索。如本书第二章介绍，使用搜索引擎查找是获取数据来源的重要方式。但搜索引擎常返回海量的搜索结果，去粗取精耗时劳神，使用网络高级搜索的方法，添加搜索限定条件则可优化搜索结果，快速锁定数据，极大节省时间。

一、理解网址结构

在了解网页结构的基础上，还需要进一步了解网址的结构，由此才能提出搜索限定条件。网页地址的英文是 Uniform Resource Locator（URL），直译为统一资源定位符，简称网址，是因特网上标准资源的地址，如同网络上的地址牌。一个标准的 URL 网址的构成如下：

[协议]://[服务器地址]:[端口]/[文件路径][? 查询]

我们以下面一段网址为例做详细讲解。

https://movie.douban.com/celebrity/1047973/movies?sortby=time&format=pic

这段网址中开头的部分——“https://”为协议，它表示只能通过 HTTPS 这套规则访问页面资源。每一个 URL 地址都标识了一个资源，资源存放于服务器，协议规定了访问资源的规则。

示例网址的第二部分“movie.douban.com”是服务器地址，也可称作主机名。在高级搜索中，我们可将搜索范围限定于某个服务器。

端口可以理解为网络服务器的门。一般来说 HTTP 协议的端口为 80，HTTPS 协议的端口为 443。示例中没有显示端口，即默认为 443。可尝试将端口号补充于网址上，实际指向同一个网页（https://movie.douban.com:443/celebrity/1047973/movies?sortby=time&format=pic）。

其后到问号之前的部分，也就是示例网址中的“celebrity/1047973/movies”，指示的是最终文件所在的路径和文件名。

示例网址中问号到结尾的部分称作参数，也叫查询，即“?sortby=time&format=pic”部分，是向服务器传递信息的部分。

下面我们再来重新标注网址的结构（见图 3-5）：

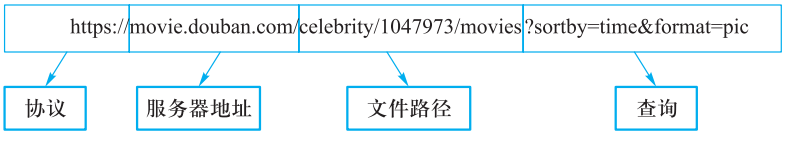


图 3-5　网址结构

二、高级搜索命令

高级搜索是使用命令限定条件进行搜索的方式，需使用英文输入法下的标点符号键入命令。下面介绍常见的高级搜索命令，这些命令适用于不同的搜索引擎，本书以“必应”为例进行讲解。

需要注意的是，高级搜索命令的执行结果受到搜索引擎排名规则（如商业逻辑、排列顺序规则等）的影响。部分命令执行后返回较多广告网页，因此本书未提供相关搜索结果示例图，有的示例图遮挡了广告部分。

1. 加号（+）

表示逻辑“与”或“并”，可连接两个及以上搜索条件。

示例：搜索同时含有“北京”和“上海”这两个关键词的网页。

命令：上海 + 北京

说明：加号前后是否空格不影响搜索结果。

返回结果见图 3-6。

图 3-6 使用“+”命令搜索结果

2. 减号（-）

表示逻辑“非”，排除特定搜索条件，即不含特定搜索结果。

示例：搜索含有关键词“番茄”但不含“鸡蛋”的网页。

命令：番茄 -鸡蛋

说明：在表示逻辑“非”时，减号前需空一格，减号后不含空格，否则会被视为连接符。

返回结果如图 3-7。

图 3-7　使用“-”命令搜索结果

3. 标题（intitle）：对网页的标题栏进行搜索

使用浏览器打开网页时，标题栏内容一般显示为标签，网页标题即源代码中 <title></title> 之间的部分，如图 3-8 标签上显示“上海市人民政府”，其所对应的代码为“<title> 上海市人民政府 </title>”。

网页设计时通常言简意赅地将网页主要内容写在标题栏中，由此只需查询标题栏便可找到与内容相关的网页。

示例：搜索网页标题栏中有关键词“上海”的网页。

命令：intitle: 上海

说明：命令与搜索词间是否存在空格不影响搜索结果。

4. 网站（site）

在使用数据时，需要辨别数据的可信程度。一般而言，从原始出处获得的数据较为可信，这就需要判断数据归属哪里，从原始出处入手寻找。比如研究教育政策时，出现在教育部官方网站上的数据会比出现在新闻网站或其他社会网站上的数据更可信，这就需要使用“site”命令，将搜索范围限定于教育部官方网站。

图 3-8 网页标题栏的显示及对应代码

示例：搜索教育部网站上有关开学日期的信息（开学日期最权威的发布方是教育部门，将搜索限定在教育部网站，可以较快地找到源数据，见图 3-9）。

命令：开学日期 site:www.moe.gov.cn

图 3-9 使用“site”命令搜索结果

5. 网址（inurl）

很多网站会把相同文件类型的资源汇总在同一个目录下，并体现在网址的文件路径部分中（参见前文网址结构介绍）。比如这段网址：“https://mcchou.com/mp3/16323.html”，网址中含有“mp3”，指示了文件路径，说明网站将 mp3 类型的文件存储在一起。搜索时可以先找到同样类型文件资源的存储路径，接着按这一路径做进一步搜索。形象地说，inurl 提供了专题内容搜索，将搜索范围限定在 url 链接中，通过使用 inurl 命令先锁定某类属性的内容，进而在这类内容中做进一步搜索。

示例：搜索 mp3 格式的《明天会更好》。

命令：inurl:mp3 明天会更好

说明：这则命令只在部分搜索引擎上适用。

6. 文件类型（filetype）

一般来说，文件扩展名标示了文件的运行程序，通过搜索特定的文件扩展名可以找到特定类型的文件。

示例：搜索有关高等教育的 word 文档文件

命令：filetype:doc 高等教育

返回结果见图 3-10（结果受到搜索引擎商业逻辑与排序规则影响）。

图 3-10　使用“filetype”命令搜索结果

7. 精确匹配（“ ”）

在搜索时经常出现搜索词组分开出现在网页中的情况，如果希望词组组合在一起出现，就可使用引号标注。为搜索关键词加上双引号意为精确匹配，即搜索关键词完整地出现在网页中。比如输入搜索词“明天会更好”，返回结果中 5 个字会连在一起出现。

示例：搜索精确匹配数据新闻的网页

命令：“数据新闻”

小练习

请使用搜索引擎查找 2022 年中国高等教育毛入学率具体数字。

在必应搜索引擎中输入“2022 年中国高等教育毛入学率”，共返回约数十万个结果，不少新闻网页都提到了毛入学率达到了 59.6%，但新闻生产必须找到数据原始出处或使用权威数据。这个数据的权威披露方应为教育部，使用高级搜索命令，将搜索网站限定为教育部官网。

命令：“高等教育毛入学率” site: moe.gov.cn inurl:2022

- 加双引号意即这段文字完整地出现在网页中。
- 教育部是发布高等教育毛入学率的官方机构，所以限定在教育部网站搜索。
- 教育部网站网址命名包含时间要素，可引入 inurl 命令，将时间作为限定条件。同时本年度通常发布上一年度的统计数据，由此搜索网址中含有 2022 的网页，而非 2022。

返回结果如图 3-11：

图 3-11 小练习示例搜索

第三节　使用工具软件抓取网络数据

使用工具软件抓取网络数据简单易行，无须编程即可实现操作，适合初学者上手。但工具软件抓取较多应用于结构化网页，抓取结构复杂或存在较多交互设计的网页时就显得捉襟见肘，因此无法适用于多种多样的数据采集场景。本节简要介绍使用工具软件抓取数据的方法，但鼓励读者学习编程方法采集数据（第四节），以获得更大选择自由度。表 3-2 列出了主要的网络数据抓取软件，这些抓取工具功能大同小异，掌握一种便可触类旁通。

表 3-2　网络数据抓取工具介绍

工具名称	简介	系统适用情况
八爪鱼采集器	可以使用采集模板，也可以自定义采集任务。在一定程度上可应对防采集措施与复杂的网页结构，同时可实现云采集。	可在 Windows 系统和 Mac 系统中使用。
后羿采集器	操作简单，流程可视化，无须编程。采集和导出无数量限制，完全免费。可一次性批量抓取网页，后台运行，速度实时显示。	安装简便，可在 Windows、Mac、Linux 系统中使用。
火车采集器	可做抓取工具，也可以用于数据清洗、分析、挖掘、可视化等工作。可抓取网页上的文本、图片、压缩文件、视频等内容。	目前仅可在 Windows 系统下使用，Mac 系统无法使用。

下面以“后羿采集器”为例简介获取网络数据的过程，它只需四步：输入网址、选择模式、实施采集和导出数据。

第一步，打开后羿采集器页面，输入需要采集的网址，随后操作区将加载网页。

第二步，选择采集模式。后羿采集器提供了两种采集模式，一是适用于初阶操作的智能采集模式，这种模式无须采集者设定条件，就可以智能分析网页内容和分页，自动按字段进行采集；二是流程图模式，即自行设定采集规则。对较为规则的网页而言，采用智能模式即可。智能采集模式中采集器自动分析网页内容后，会将网页数据列在页面下方的表格中（见图 3-12）。

第三步，如需增加采集字段，在点击新的字段后，跟随光标选择需要采集的内容（见图 3-13），点击完成。

第四步，点击开始采集，随后便可导出数据。

如需进行网页翻页采集，则可以点击“设置采集范围”按钮。在弹出的对话框中，设置起始页和结束页（见图 3-14）。

接下来，页面会转至连续采集状态，结束后导出数据即可。

后羿采集器也可实现社交媒体平台数据的采集。下面我们以微博“保研超话”为例，展示后羿采集器采集社交媒体平台数据的过程。将“保研超话”的网址输入地址栏，后羿

采集器便开始进行网页内容解析。采用智能采集模式可将“超话”下的跟帖内容悉数采集（见图 3-15），包含发帖者 ID、发帖时间、发帖内容及其点赞量等。

图 3-12 采集操作页面

图 3-13 增加采集字段

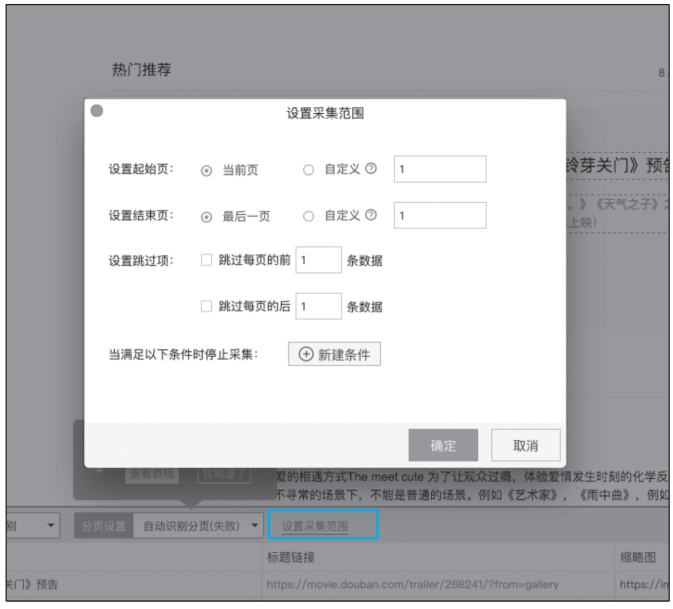


图 3-14　设置采集范围

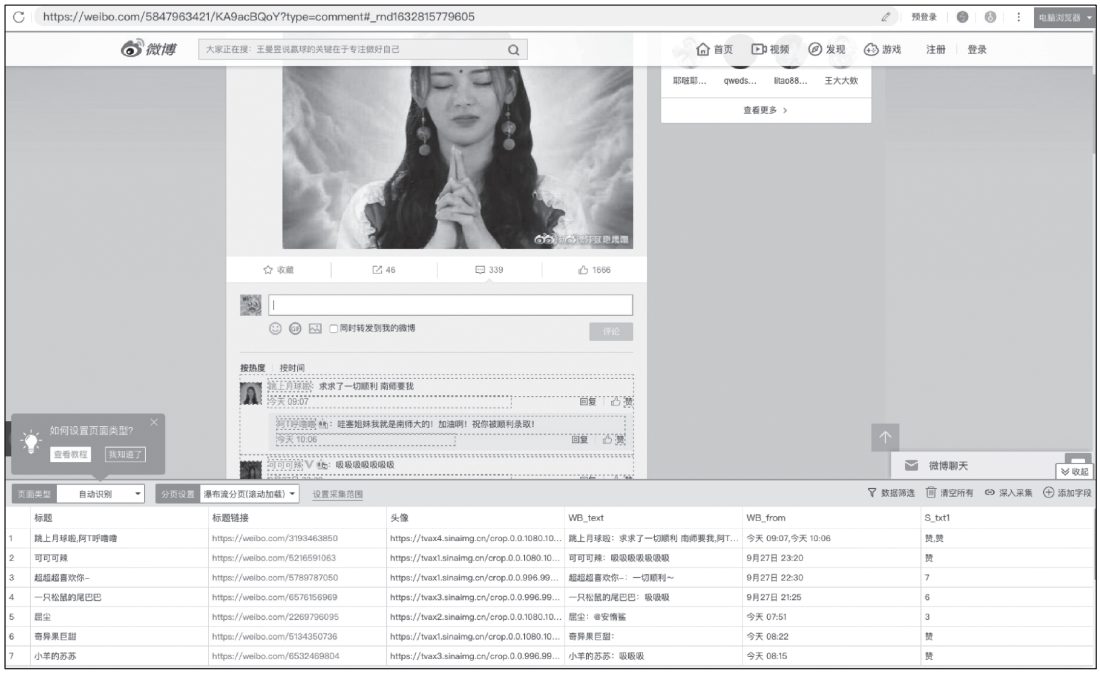


图 3-15　社交媒体平台数据采集

如需提取链接中的数据，即采集详情页数据，可使用后羿采集器的“深入采集”功能。将网址载入后，后羿采集器会自动分析网页并提取出网页中的链接，如未能自动识别，可手动添加新字段，抓取链接地址，右键点击标签栏将其属性设置为“提取链接地址”，接着点选“深入采集”，就可以采集详情页数据了（见图 3-16）。

图 3-16 进入“深入采集”

点击“深入采集”后，会默认打开第一条链接对应的详情页（见图 3-17）。也可以点

图 3-17 深入采集详情页

击链接字段中的任意一条链接，都能够打开详情页。接着在新打开的页面中通过自动识别或“添加字段”设置采集规则，其他页面会自动依据规则进行采集。

第四节　使用 Python 编程语言抓取网络数据

使用编程语言编写程序抓取网络数据具有更广泛的适用性。不规则网页、网页弹出窗口或是具有较多交互设计的网页可能无法简单使用工具软件进行数据抓取，需要通过编程语言来实现抓取。此外，当下的信息环境日益受到不断涌现的新技术的冲击和影响，身处其中的新闻传播工作者需要具备跨学科的视野，掌握大数据时代所要求的复合传播技能，编程能力即是从事数据新闻工作的必备能力之一。

当然，对于从未接触过编程语言的新手来说，从零开始学习任何一门编程语言都是不易的。新闻传播专业的学生可以采取拾级而上的学习策略。第一步，通过搭建编程语言运行环境和学习语法规则，做到看懂和理解编程语言；第二步，通过查阅相关代码示例，自己动手写一段简单的代码或者改写一段代码（在学习代码初期，拷贝与改写代码是重要和有效的学习策略）；第三步，在达到一定熟练程度之后，可以通过实际案例学习自行编写代码。本书鼓励初学者按照编程学习的阶梯循序渐进。万事开头难，要坚信熟能生巧、百炼成钢。

多款编程语言都可实现网络数据抓取，比如 Java 语言、PHP 语言及 Python 语言等，本章推荐使用 Python 语言[①]。首先，Python 语言功能强大，它有很多包和库，帮助使用者解决各种问题，适用范围较广，比如抓取数据的 Selenium 库和 Beautiful Soup 库，用来做数据分析的 Pandas 库，以及制作可视化的 Matplotlib 库等；其次，它是一款容易学习的语言，代码简洁明了、语法规则明晰；最后，Python 的集成开发环境有代码报错功能，如果代码编写有误，Python 会非常明确地提示错误位置和错误类型，初学者可拷贝错误提示进行网络搜索，寻找解决方案。

一、Python 语言基础简介[②]

1. Python 运行环境

Python 既可以在操作系统中直接运行，也可以安装集成开发环境运行。本书选择一种对初学者来说比较友好，且可以直接看到代码运行效果的方法，在 Python 开发环境——Anaconda 上运行。安装成功后，可使用 Anaconda 平台上集成的 Jupyter Notebook 运行 Python。

---

① Python 是一款面向对象的、解释型程序设计语言。计算机程序语言一般可分为解释型语言和编译型语言，前者可一句一句直接运行代码，后者需要经过编译器编译为机器代码后再运行。Python 由出生于荷兰的工程师吉多·范·罗苏姆（Guido van Rossum）设计创造，第一版发布于 1991 年。它简便易用，是一款开源的程序设计语言，可在任何操作系统中运行。

② 更多 Python 语言在线学习资源可参见 RUNOOB 网。

Anaconda 的安装

1. 官网安装

第一步，进入 Anaconda 官方网站。

第二步，点击 Download 按钮。

注意选择最新发布的版本，并依据操作系统选择对应的安装包（Windows 系统有 32 位和 64 位之分）。

2. 清华大学开源软件镜像站下载安装（国内 IP 速度更快）

打开网址，选择最新发布的版本，选择相对应的安装包，MacOS 用户使用 .pkg 格式的安装包。

Jupyter Notebook 是一款开源网络应用程序，可用于创建和共享包含实时代码公式、可视化效果和文本的文档。使用它进行编程，可以直接通过浏览器运行代码，同时在代码块下方展示运行结果。

使用 Jupyter Notebook 运行 Python

第一步，打开“我的应用”，找到 Anaconda Navigator 图标（见图 3-18），双击运行。

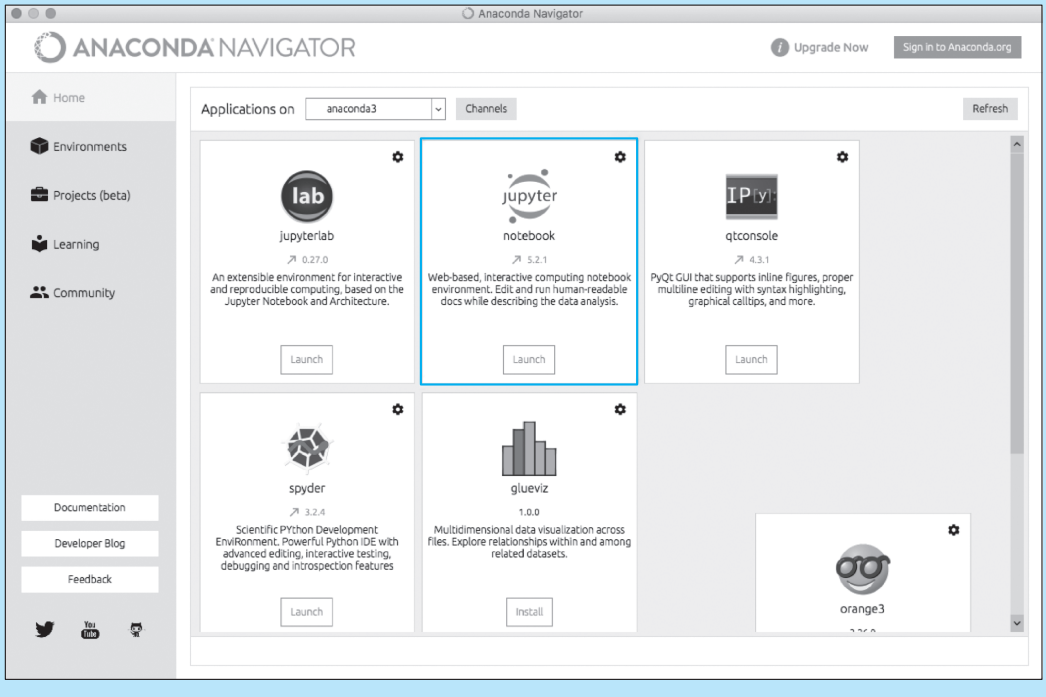


图 3-18 Notebook 选择界面

第二步，在打开的 Anaconda 操作界面中找到 Jupyter Notebook，点击 Launch 按钮，此时电脑默认浏览器会弹出新窗口。

第三步，如果未能弹出新窗口，则会弹出一个新的终端界面（见图 3-19），打开其中的镜像网址（阴影标示部分）即可。

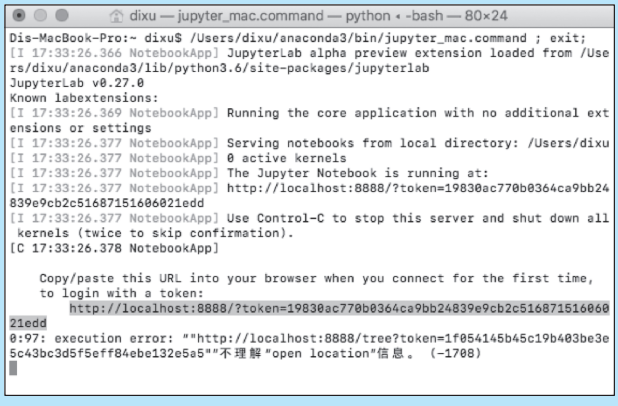


图 3-19　终端界面

2. Python 书写规范

书写规范规定了代码的编写格式，编写格式正确才能顺利运行。我们在 Jupyter Notebook 中右上角找到“New”按钮，点击新建一个页面以学习书写规范。Jupyter Notebook 的操作页面见图 3-20。

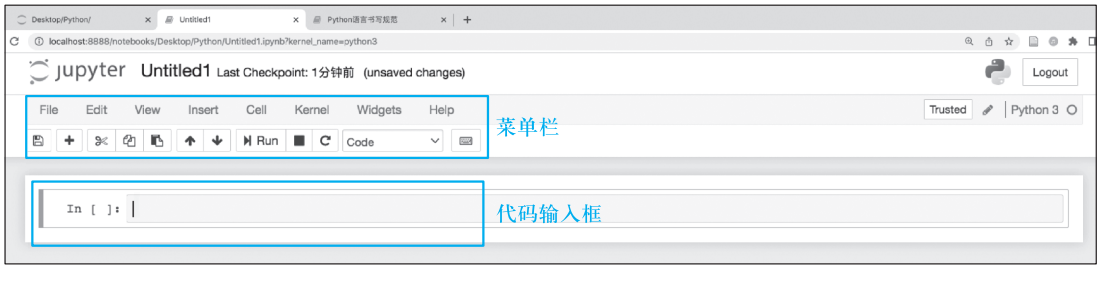


图 3-20　Notebook 操作页面

编写程序时经常要添加注释，对代码或输入内容作出解释，在 Notebook 中输入注释时，需要增加标注以与代码相区别，Notebook 提供了两种标注方式：使用井字符号“#”或是使用菜单选择栏中的标记。请注意代码中必须键入英文标点符号。键入内容后，可以选择标签栏上的“运行”（Run）按钮运行[①]，也可使用快捷键 Shift+Enter 运行。可尝试如下代码（本书所用的 Python 版本为 Python3）。

① 如未加或错加标注符号直接运行则会报错。

```
In [1]①: #"输入 # 字符号，说明这是一条注释，字体相应变为绿色斜体。"
In [2]    "hello wolrd!"
# 也可在代码后面增加注释，只需键入 # 字符号。
Out [3]   'hello wolrd!'
```

3. Python 语法规则

学习编程语言，最重要的是学习它的语法规则，这是理解和读懂代码的基础。Python 语言的语法规则主要包括基本概念、数据类型以及函数、模块和库。本书还将介绍条件命令与循环命令，为数据抓取做铺垫。

（1）基本概念

① 输出

编写程序时，需要计算机对变量或常量进行输出，Python 使用内置的 print 函数实现输出——print（字符串 / 变量）。

```
print('Hello')
# 输出字符串 Hello
```

结果为：Hello

② 变量

在程序中总是完整地书写字符串和数值相对烦琐，为数据命名可以更方便地调用它们，这个名称叫作变量。变量有三个命名原则：首字符须为字母或下划线，其余标识符可以是字母、下划线或数字，大小写敏感。

```
T=(3)
# 设定变量 T（大写），赋值为数值 3。
print(T)
# 输出变量 T 的值。
```

结果为：3

```
t=(4)
# 设定变量 t（小写），赋值为数值 4。
print(t)
# 输出变量 t 的值。
print(T,t)
# 同时返回变量 T（大写）和 t（小写）的值。
```

结果为：3 4

③ 表达式

表达式是由运算符将不同类型的数据连接起来组成的。

---

① 代码前的英文单词 In 表示输入命令，Out 表示输出结果，后文将不再特地指明，# 字符号后是对代码含义的解释。

```
2+2
# 算术运算符连接的表达式。
```

结果为：4

```
2*2
# 算术运算符连接的表达式。
```

结果为：4

```
2>1
# 比较运算符连接的表达式。
```

结果为：True

④ 语句

语句是指完整地执行一项任务的代码。语句有多种类型，有赋值语句、逻辑运算语句等。语句与表达式的区别在于前者是要完成一项任务，而后者则是任务中的一个具体组成部分。

```
X = 3
# 为变量 X 赋值为 3。
Y = 2
# 为变量 Y 赋值为 2。
print(X + Y)
# 输出变量 X 与 Y 相加的结果。
```

结果为：5

语句的显示方式需要格外注意。Python 中的语句要正常显示需注意分行和缩排。通常以新起一行作为语句的结束标志。多条语句也可以在同一行输入，但需增加分隔符——分号（;）。一条语句也可分多行显示，需使用反斜杠作为标识（\）。

```
X = 3
Y = 2
print(X)
print(Y)
# 新起一行区分出两条语句。
```

结果为：3
　　2

```
print(X);print(Y)
# 同一行中写两条语句则需要加分隔符（;）。
```

结果为：3
　　2

```
total = X + \
        Y + \
        5
# 使用反斜杠（\）将一条语句分为多行显示。
```

```
print(total)
```

结果为：10

相较于其他编程语言，Python 语言的独特之处在于它使用缩进来控制代码模块，即不同级别的代码块之间通过缩进来分隔。编写代码时可用空格键或 Tab 键实施缩进，二者不可混用，在同一代码块中使用两种缩进方式将会报错。使用空格键进行缩进时，空格的数量可变，两个或四个空格均可，但同一代码块中空格数量必须一致。

```
X = 3
# 将变量 X 赋值为 3。
if X == 3:
# 条件语句，意为如果 X 是 3，返回 True，否则返回 False。
    print(True)
else:
    print(False)
```

结果为：True

上面这段代码是一个条件语句，两个等号连写“==”，意为如果条件成立。if 语句与 print 语句为同一个代码块，print 前面缩进了四个空格或一个制表符位置。同理，else 语句与 print 语句也是同一个代码块，也要缩进四个空格或一个制表符位置。需要注意的是，如果第一个缩进使用了空格键作为缩进方式，则第二个也要使用同样方式。图 3-21 中列出了因缩进产生的错误，并对错误提示给出解释。

```
In [72]: if X == 10:
             print(X)
           print(10)    #缩进格式不统一
           File "<tokenize>", line 3
             print(10)
             ^
         IndentationError: unindent does not match any outer indentation level

In [73]: if X == 10:
         print(X)       #应有缩进
           File "<ipython-input-73-7af9273710a9>", line 2
             print(X)
                 ^
         IndentationError: expected an indented block

In [74]: print(X)
           print(10)    #不应有缩进
           File "<ipython-input-74-9f083ee991e2>", line 2
             print(10)
             ^
         IndentationError: unexpected indent
```

图 3-21 缩进错误类型

（2）数据类型

Python 中有多种数据类型服务于数据抓取，我们主要介绍五种数据类型：数值与布尔型、字符串、元组、列表和字典。抓取数据时，需要理解和灵活使用这些数据类型。

① 数值与布尔型

Python 可以进行多种数学运算，它提供了多种可用的数值类型，主要有整型（即整数）、浮点型和复数，可以使用 type 函数查阅不同数值的类型。

布尔型是一种特殊的数据类型，通常用于比较和判断，只有 True 和 False 两个值，注意首字母必须大写。

```
type(1.0)
# 返回 1.0 的数据类型，1.0 带有小数点，float 意为浮点型。
```

结果为：float

```
type(1111)
# 返回 1111 的数据类型，1111 为整数，int 是 integer 的缩写，意为整型。
```

结果为：int

```
type(True)
# 返回 True 的数据类型，bool 为布尔型。
```

结果为：bool

```
3 == 3
# 判断数值 3 等于 3，返回结果为 True，即正确。
```

结果为：True

```
3 == 5
# 判断数值 3 等于 5，返回结果为 False，即错误。
```

结果为：False

② 字符串

字符串是由任意字符构成的数据类型。输入字符串时要以引号开始并以引号结束，在 Jupyter Notebook 中字符串显示颜色会发生改变。使用单引号（‘）、双引号（“）或三引号（”’）均可，单引号较为简便，但需注意的是，Python 中要使用英文输入法键入命令。

```
'You can do it!'
```

结果为：'You can do it!'

```
"You can do it!"
```

结果为：'You can do it!'

```
'''You can do it!'''
```

结果为：'You can do it!'

可以看出，当使用不同的引号时，Python 都将其识别为字符串并返回同样的结果。有时也可以使用引号嵌套来陈述一段引语。

```
'I said, "You can do it!"'
# 单引号嵌套双引号，双引号中是一句引语。
```

结果为：'I said, "You can do it!" '

```
"I said, 'You can do it!'"
# 双引号嵌套单引号。
```

结果为："I said, 'You can do it!' "

但在以下示例中，引号带来了麻烦（见图 3-22）。

```
'I'm working on it.'
#I'm 是英文缩写，Python 无法识别缩略符号（'），会将其识别为字符串结束的标志。
```

```
'I'm working on it'
  File "<ipython-input-1-f224ba98b841>", line 1
    'I'm working on it'
       ^
SyntaxError: invalid syntax
```

图 3-22 字符串输入错误提示

上述错误提示表明 Python 无法识别这个句子，因此需要加入转义符号——反斜杠（\），将缩写符号转为字符串。

```
'I\'m working on it.'
# 加入转义符号（\）后，可以正确识别缩略符号，将其识别为字符。
```

结果为：'I'm working on it.'

使用三引号可以输出多行文本，它适用于为字符串换行，也可以使用转义符号（\n）分行输出文本。

```
print('''You can do it!
Python is not that difficult.
You just need to be more confident.''')
# 使用三引号，实现换行输出。
```

结果为：

You can do it!

Python is not that difficult.

You just need to be more confident.

```
print('You can do it!\n Python is not that difficult.\n You just need to be more confident.')
# 使用转义符号（\n）实现换行输出。
```

结果为：

You can do it!

Python is not that difficult.

You just need to be more confident.

③ 元组（tuple）

Python 中内置了多种数据类型，包括序列类型、集合类型和映射类型等。序列是最

基本的数据结构，序列中包含元素，每个元素对应一个位置或索引，第一个位置或索引是0，第二个是 1，依此类推。常见的序列类型有元组和列表。

元组是有序的、不可更改的数据序列，使用小括号“()”创建。元组中的元素使用逗号区隔，既可以是数值也可以是字符串等。元素的值不可更改，元组中元素的位置从 0 开始。

```
tuple = (1,'Emma', 2,'Mike')
# 创建了一个元组，包含四个元素，使用逗号隔开。
print(tuple)
# 输出元组中的值。
```

结果为：(1, 'Emma', 2, 'Mike')

```
tuple[1]
# 返回元组中第二个元素的值，此处要注意，元组中元素的位置从 0 开始。
```

结果为：'Emma'

```
tuple[0] ='2'
# 将元组中第一个元素的值改为 2，返回错误提示，因为 Python 中元组对象不支持赋值操作。
```

④ 列表

列表是有序的、可更改的数据序列，使用方括号“[]”创建，列表中的元素同样使用逗号区隔。列表可以通过从 0 开始的整数进行索引，使用索引可以查找特定的元素，也可以修改更新现有元素。应用于数据抓取时我们可以通过索引来访问特定数据。

```
list1 = ['Emma', 1, 'Mike', 2, 'Cobi', 3]
# 创建 list1，包含六个元素，用逗号隔开
print(list1)
# 输出列表中的值
```

结果为：['Emma', 1, 'Mike', 2, 'Cobi', 3]

```
list1[0]
# 输出列表中第一个元素的值，0 表示列表中第一个元素
```

结果为：'Emma'

```
list1[1:]
# 从列表中第二个元素开始输出直至列表结束
```

结果为：[1, 'Mike', 2, 'Cobi', 3]

如果要为列表添加新元素，可使用 append 函数或是 extend 函数。前者表示将新元素看作一个对象整体打包添加到列表中；后者则是将新元素看作一个序列，将其与列表中的序列合并，放在后面。

```
list1 = ['Emma', 1, 'Mike', 2, 'Cobi', 3]
list2 = ['Sophie', 4, 'Lily', 5]
list1.append(list2)
list1
```

```
# 创建列表 list1 和 list2，使用 append 函数将 list2 打包作为整体加入 list1，list2 成为 list1 中的一个元素。
```

结果为：['Emma', 1, 'Mike', 2, 'Cobi', 3, ['Sophie', 4, 'Lily', 5]]

如果要直接增加多个元素则要使用 extend( ) 函数。

```
list1 = ['Emma', 1, 'Mike', 2, 'Cobi', 3]
list2 = ['Sophie', 4, 'Lily', 5]
list1.extend(list2)
list1
# 将 list2 作为元素添加至 list1 末尾。
```

结果为：['Emma', 1, 'Mike', 2, 'Cobi', 3, 'Sophie', 4, 'Lily', 5]

⑤ 字典

字典是映射类型的数据，使用花括号“{}”创建，包含一系列键，每个键对应一个值，键和对应的值之间用冒号“:”连接。字典中键一般是唯一的，值不需要唯一。字典是可变的、无序的数据组合，不可以进行按位置截取等操作，但可以添加或移除数据项。

```
dict = {1:'Emma', 2:'Cobi', 3:6, 4:'Monna'}
# 创建一个字典变量 dict，每个键对应一个值，键和对应的值用冒号连接。值既可以是数字也可以是字符串。
print(dict.keys())
# 输出字典变量 dict 的全部键。
```

结果为：dict_keys([1, 2, 3, 4])

```
dict.values()
# 输出字典变量 dict 的全部值。
```

结果为：dict_values(['Emma', 'Cobi', 6, 'Monna'])

（3）函数、模块和库

编程过程中，我们会经常调用相同或者类似的操作，这些操作由同一段代码完成。函数可以帮助我们避免重复编写这段代码，它是指可以重复使用的、用来实现一定功能的代码块。Python 中内置了一些函数，比如前面使用到的 print 函数以及 type 函数，使用者也可自行创建函数。

模块将代码依据一定的逻辑组织在一起，模块内包含函数、变量等。而多个具有相关功能的模块集合在一起便称作库或是包。库或包，包含着模块，模块包含着代码。

Python 的强大之处就在于它内置了多个标准库，还有大量第三方库以及自定义模块，比如 Pandas 和 Selenium 都是第三方库，前者可用于数据分析，后者可用于抓取网络数据。调用库之后，便可使用库中的函数来实现不同操作。导入模块或者库使用 import 语句，示例如下。

```
import pandas as pd
# 导入第三方库 pandas，并将其简写为 pd（简写方便调用，无须每次写全称）。
pd.read_csv('1')
```

```
# 导入后使用 pandas 库中的函数 read_csv 读取名为 1 的 .csv 文件。pd 是 pandas 的简写，表明调用的是 pandas 库中的函数。
```

（4）条件命令

条件命令指通过一条或多条语句来控制程序执行的代码块，常见的条件命令语句为 if 语句。执行过程如图 3-23 所示，先输入判断条件，如符合条件则执行代码，返回代码结果。如不符合判断条件，则直接返回结果。

在条件命令书写格式中，判断条件即要执行的条件命令的内容。执行语句与判断语句为同一个代码块，需缩进。书写格式如下：

```
if 判断条件：
    执行语句
else:
    执行语句
```

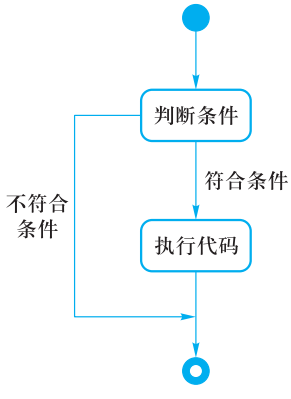


图 3-23　条件命令图示

下面是一则简单的条件命令，分别设置了是和否两个选项。

```
X = 10
if X == 10:
    print(X)
else:
    print('否')
```

结果为：10

这段代码意为为变量 X 赋值 10，如果 X 等于 10，则返回 X。下面是不符合执行条件的案例。

```
X = 9
if X == 10:
    print(X)
else:
    print('否')
```

结果为：否

10 不等于 9，不符合条件判断，所以执行 else 语句，返回字符串“否”。

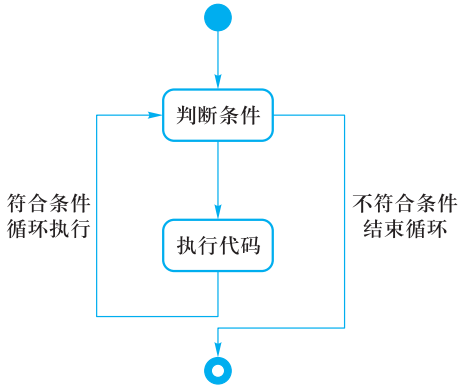


图 3-24　循环命令执行图示

（5）循环命令

循环命令允许多次执行一条或多条语句，执行过程如图 3-24 所示。当符合条件时，执行循环命令，不符合条件即返回其他结果。

常见的循环命令有 for 语句和 while 语句。

① for 语句

使用 for 语句可以遍历序列中全部元素。

for 语句的语法格式如下：

```
for 重复出现的变量 in 序列数据：
    执行命令
```

下面是一则实例。

```
X = [2, 4, 6, 8, 10]
for number in (X):
    print (number)
```

结果为：

2

4

6

8

10

上面代码表示创建一个列表 X，列表为序列数据。对于序列来说，数字是重复出现的变量，使用 for 语句遍历列表中的全部数字。

for 语句常与 else 语句连用，表示在结束循环后执行 else 语句。

```
X = [2, 4, 6, 8, 10]
for number in (X):
# 遍历列表 X 中的全部数据。
    if number < 5:
        print(number, '小于 5')
    else:
        print(number, '大于或等于 5')
# 如果数据小于 5 则执行循环，如果大于或等于 5，则执行 else 语句。
```

结果为：

2 小于 5

4 小于 5

6 大于或等于 5

8 大于或等于 5

10 大于或等于 5

② while 语句

while 语句用于执行循环程序，处理需要重复执行的任务。while 语句的格式如下：

```
while 判断条件：
    执行语句
else:
    执行语句
```

while 语句和 for 语句等循环语句可以用来实现网页的循环抓取，例如当网页需要翻页时，就需要使用循环语句。下面是 while 语句的实例：

```
X = 0
# 赋值语句。
    while X < 9:
# 判断条件为 X 小于 9
    print(X, '小于 9')
# 符合条件则执行输出语句。
    X = X+1
# 循环命令。
else:
# 不符合条件则结束循环。
    print(X, '大于或等于 9')
# 如果变量 X 小于 9 就输出 X 小于 9，否则输出大于或等于 9。
```

结果为：

0 小于 9
1 小于 9
2 小于 9
3 小于 9
4 小于 9
5 小于 9
6 小于 9
7 小于 9
8 小于 9
9 大于或等于 9

二、使用 Selenium 库抓取数据


Python 抓取数据教学视频

Python 语言中有多个库可以实现网络数据抓取，我们推荐非计算机专业的初学者使用 Selenium 库，它通过控制浏览器执行操作，每一步代码执行的动作都可以实时呈现，这种所见即所抓的方式能够帮助初学者理解代码的意义。但相较于其他库，Selenium 亦有不便之处，比如运行速度略慢、安装烦琐等。

学有余力的读者，可进一步学习使用 Beautiful Soup 库抓取数据。在本节实际操作环节，我们首先学习使用 Selenium 库编写程序抓取网络数据，同时提供使用 Beautiful Soup 库抓取同样数据的全部代码，方便读者比较和学习。

1. 安装 Selenium 库

安装 Selenium 库过程略为烦琐，安装过程可视作编程学习中的一个挑战，完成这个挑战可以增进对代码的理解。

安装 Selenium 库

第一步，导入 Selenium 库。

Mac 系统：新建一个 Jupyter Notebook 文件，键入安装命令“!conda install -y -c conda-forge selenium”，完成后显示如图 3-25 所示：

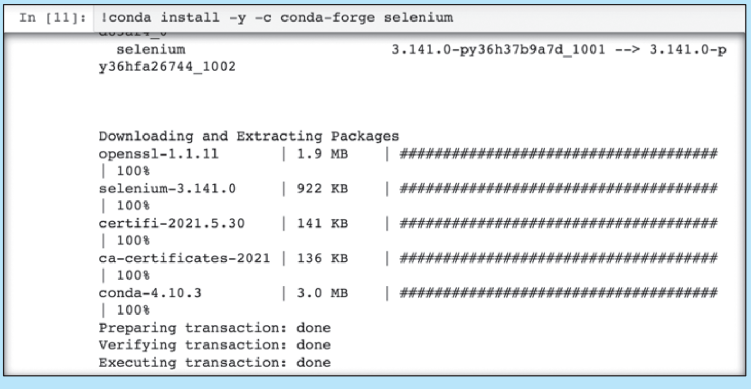


图 3-25 Mac 系统 Selenium 库安装示意图

Windows 系统：在应用程序中打开 Anaconda Prompt。

在新弹出的窗口中键入安装命令“conda install -y -c conda-forge selenium”，回车，如图 3-26 所示。

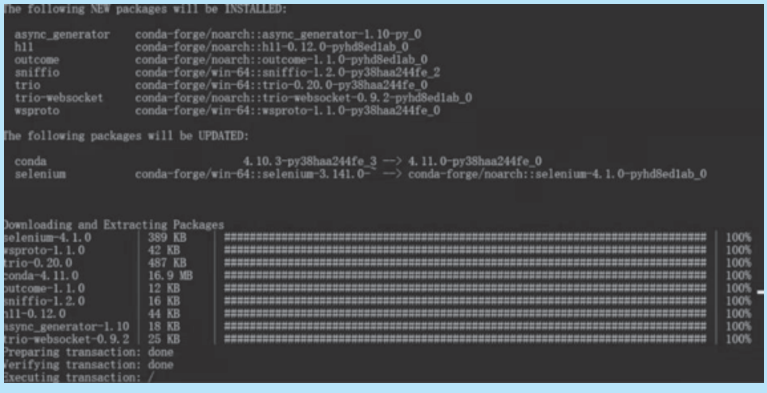


图 3-26 Windows 系统 Selenium 库安装图示意

第二步，声明调用的浏览器。在此步骤中，Mac 系统和 Windows 系统操作方法相同。

Selenium 支持调用多个浏览器，它通过向浏览器发出命令来抓取数据，需要声明所调用的浏览器。我们以 Chrome 浏览器作为默认浏览器，也可用同样命令调用其他浏览器。

键入如下命令，意为引入网页驱动，将 Chrome 设置为受控制的浏览器：

```
from selenium import webdriver
driver = webdriver.Chrome()
```

执行成功后会自动弹出新的浏览器窗口，并显示为 Chrome 正受到自动测试软件的控制，如图 3-27 所示。

图 3-27　Selenium 库安装成功示意图

第三步，安装浏览器驱动。如无法成功弹出窗口，通常需安装 Chrome 浏览器驱动。

首先查询 Chrome 版本，点击 Chrome 浏览器右上角的三个圆点，找到“关于”。

接着找到对应的驱动版本，下载与 Chrome 版本对应的驱动。

Mac 系统：下载后将驱动解压至 usr/bin 文件夹，不同电脑路径可能不同。

Windows 系统：下载后将驱动解压至 Anaconda 文件夹中 Scripts 目录（不同电脑路径可能不同）。

安装成功后，回到 Jupyter Notebook，再次键入调用浏览器命令，实现控制。

电脑一直报错，怎么办?

执行了上述步骤，仍然无法运行代码，怎么办?

在学习数据抓取的过程中，一定会遭遇许多“崩溃时刻”——无论如何尝试，电脑始终无法正常显示。无须着急，每台电脑各有不同，运行环境也不同，当无法在书本上找到解决方案时，不妨求助网络。将报错信息贴至搜索引擎，看看有没有“过来人”提供解决方案，每一次解决问题都意味着技能的增长。

2. 访问网页

成功安装后，我们可以通过 driver.get(“ 网址 ”) 代码控制浏览器访问页面。代码运行后，受控制的浏览器会登录该网址，编写者可以看到代码运行对浏览器产生的具体结果。

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("http://movie.douban.com/top250")
```

小练习

请运用 Selenium 编写一行代码，控制浏览器访问你的学校主页。

3. 查找元素

在实施抓取之前，首先要解析网页源代码，确定数据所对应的代码，随后通过查找元素命令锁定这些数据进行抓取。在新版 Selenium 中需要引入 By 模块来定位元素，引入的代码为“from selenium.webdriver.common.by import By”，名称、标识符和属性名称等都可以用来锁定数据。命令中 element 在使用时需要区分单复数，单数时只查找一个数据，复数则查找全部数据。

以豆瓣电影 Top250 网页为例，查阅源代码可知电影的豆瓣评分即是通过属性名称（class）来标识的，要获取评分数据就需使用“driver.find_elements(By.CLASS_NAME)”命令。

4. 抓取示例

接下来我们使用 Selenium 库编写程序抓取豆瓣电影 Top250 网址中全部电影名称及其豆瓣评分，可分解为四步：解析网页源代码、查找元素、抓取首页内容和实现循环抓取。

（1）解析网页源代码

确定抓取字段后，解析网页源代码，确定数据字段所对应的代码。调阅源代码发现，每条电影信息都是一个列表，表示为“
_”；电影名称对应的源代码为“class="title"”，评分则对应“class="rating_num"”（见图 3-28）。

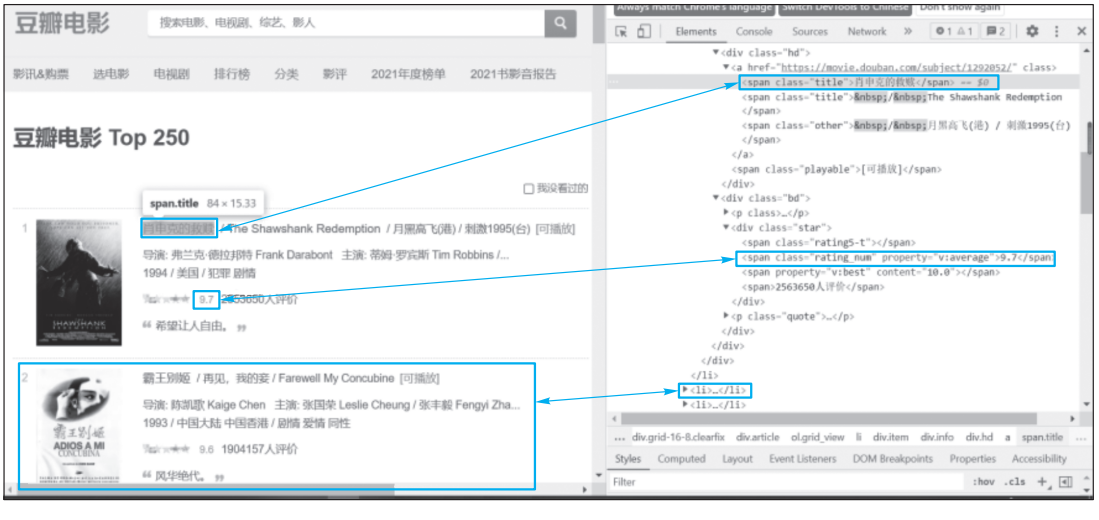


图 3-28 解析源代码

（2）查找元素

电影名称与评分都是依据属性名称来标识的，抓取时使用查找元素命令按属性找到元素，提取 title 和 rating_num 标识的数据。

```
titles =driver.find_elements(By.CLASS_NAME, "title")
for title in titles:
```

```
    print(title.text)
```

 # 解析网页中电影名称的源代码，电影名称代码为“class="title"”。设置变量 titles，控制浏览器以属性名称为标识查找元素，将查找到的全部元素存放在变量 titles 中。遍历变量 titles 中的所有元素，输出每个元素所对应的电影名称数据，即文本（text）。

运行代码后可看到浏览器由上到下滚动操作，返回了 Top250 电影名称的数据。这些数据由 title 所标识，可以看到其中含有不规则符号和电影的外文名称，这也是网络数据抓取的特点——带有噪音，即不需要的数据内容或不规则的符号，随后需通过 Excel 或 Open Refine 等工具进行数据清洗（参见第四章）。

接下来，可尝试修改代码，获取电影评分的数据：

```
ratings =driver.find_elements(By.CLASS_NAME, "rating_num")
for rating in ratings:
    print(rating.text)
```

（3）抓取首页内容

获取电影名称和评分数据之后，需要把两个数据对应起来，这时就要用到字典这种数据类型。

```
elements = driver.find_elements(By.TAG_NAME,"Li")
movies = []
for element in elements:
    try:
        title = element.find_element(By.CLASS_NAME, "title").text
        rating = element.find_element(By.CLASS_NAME, "rating_num").text
        movie = {"title": title, "rating": rating}
        movies.append (movie)
    except:
        pass
```

 # 设置变量 elements（变量名称也可自行设定），控制浏览器以标签（tag）为标识查找元素，分析网页源代码可见每一部电影被标识为一个列表（li），需在列表内抓取数据。设置空列表 movies，尝试抓取电影名称文本和评分文本，存入变量 title 和 rating 中。创建 movie 字典数据，花括号内包含键以及对应的值。键和值构成了一个项，title 和 rating 为键，分别对应 title 和 rating 值。随后将 movie 字典数据添加至 movies 空列表后。

在这段代码中，我们引入了 Python 处理异常的 try/except 语句，这个语句适合初学者使用。try 语句后连接一段执行代码，如果代码有误发生异常，则执行 except 语句后的代码。有时 except 语句后还会连接 else 语句，表示如果没有异常则执行 else 语句后的命令。上面一段代码中结尾为 pass 语句，它没有实际含义，仅作占位使用。执行完毕后可通过输出命令检查抓取结果 print(movies)。

```
try:
    执行代码
```

```
except:
    发生异常时执行代码
else:
    没有异常时执行代码
```

（4）实现循环抓取

成功抓取首页内容后，便可引入循环条件实现循环抓取。简单的循环条件设置方式如下：

```
i = 0
while i<10:
    代码块
i = i+1
```

以豆瓣电影 Top250 网页为例，每页有 25 条数据，共有 10 页，如果手动翻页，需要翻 9 次。所以设置循环次数为 9 次，也就是 i<10，i=i+1 即为循环条件，从 0 开启循环。代码如下：

```
movies = []
i = 0
while i<10:
    movies_list = driver.find_elements(By.TAG_NAME, "li")
    for element in movies_list:
        try:
            title = element.find_element(By.CLASS_NAME, "title").text
            rating = element.find_element(By.CLASS_NAME, "rating_num").text
            movie = {"title": title, "rating": rating}
            movies.append(movie)
        except:
            pass
    driver.find_element(By.CLASS_NAME, "next").click()
    driver.switch_to.window(driver.window_handles[0])
    i=i+1
else:
    print(movies)
```

仔细观察上面代码，对比抓取首页的代码，可发现除了循环条件，还多了两行代码：

```
driver.find_element(By.CLASS_NAME, "next").click()
driver.switch_to.window(driver.window_handles[0])
```

这两行代码恰好体现了用 Selenium 库抓取的特点——所见即所爬，运行代码后会看到受控制的浏览器不断进行翻页操作。试想使用鼠标翻页，需要点击网页上的“后页”按钮，而要用代码控制，就需要找到“后页”按钮的源代码。解析网页发现“后页”代码为“class="next"”，使用查找元素命令 find_element(By.CLASS_NAME, "next")，接着控制

浏览器进行点击操作“click( )”。这句代码 driver.find_element(By.CLASS_NAME, "next").click( ) 意为控制浏览器进行翻页操作。

所见即所爬，顾名思义首先要“见到”，受控浏览器进行点击“后页”操作后打开了新的页面，这时需要转到新的页面进行控制，这句代码 driver.switch_to.window(driver.window_handles[0]) 表示转到新的页面进行控制。

实现循环抓取的方法不止一种，同样使用 Selenium 库，无须进行翻页操作，通过巧妙设置网址也可实现循环抓取。

以豆瓣电影 Top250 网页为例，观察每一页的网址结构可以发现，只有“start=”后面的数字发生变化，其他内容均未变，数字部分以 25 为倍数增长，查看网页内容可知每个页面记录了 25 条电影数据。网址变化颇有规则，可通过引入 str( ) 函数来登录网址。str( ) 函数意为提取字符串，将网址中变化的部分用 str(i*25) 函数替代得到下面一行代码：

```
https://movie.douban.com/top250?start="+str(i*25)
```

同样使用循环命令，加入引用了函数的网址，代码变得更为简洁。

```
movies = []
    i = 0
    while i<10:
        driver.get("https://movie.douban.com/top250?start="+str(i*25))
        movies_list = driver.find_elements(By.TAG_NAME, "li")
        for element in movies_list:
            try:
                title = element.find_element(By.CLASS_NAME, "title").text
                rating = element.find_element(By.CLASS_NAME, "rating_num").text
                movie = {"title": title, "rating": rating}
                movies.append(movie)
            except:
                pass
        i=i+1
else:
    print(movies)
```

三、使用 Beautiful Soup 库进行抓取

使用 Selenium 抓取数据可以看到代码的实时运行效果，但正如前文所说，Selenium 安装烦琐，运转速度略慢。掌握了 Selenium 抓取方法后，我们进一步学习一种一键安装、抓取效率更高、速度更快的方法，即配合使用 Request 和 Beautiful Soup 库。这种方法无法做到所见即所爬，要求编写者具备一定代码基础，一气呵成写就代码进行抓取。我们以豆瓣读书 Top250 网页为例进行说明。

Request 模块的工作原理是向目标站点发起请求，也就是发送一个 Request，请求通常包含额外的题头（header）等信息，等待服务器响应。如果服务器能正常响应，发起者

会得到一个回应（response），回应的内容便是所要获取的页面内容，类型可以是 HTML、json 字符串、二进制数据（图片或者视频）等。得到响应之后，使用 Beautiful Soup 库对网页源代码进行解析，解析之后便可以通过标签名称、标识符和属性名称来查找元素并抓取。具体操作如下：

1. 安装和导入

使用 pip 函数安装这两个模块。

```
! pip install requests
! pip install beautifulsoup4
```

成功安装后使用 import 语句将其导入 Python 代码中。

```
import requests
from bs4 import BeautifulSoup
```

2. 发起请求，获取回应

Request 模块最重要的函数就是 get 函数，即向目标站点发出请求。其中的参数包括目标站点的网址、请求题头和超时时间（timeout），即本次访问在规定时间内未收到回应则终止。

```
res = requests.get('https://book.douban.com/top250', headers=headers, timeout=10)
```

其中比较难理解的是题头参数，header 是 HTTP 协议规定的一块数据区域，用来存储请求访问的客户端的基础信息。在爬虫当中，header 中的关键信息就是 User-Agent 参数和 Host 参数，Host 参数表示目标站点的域名。一般来说，上网时向服务器请求数据都是通过浏览器完成的，User-Agent 参数就表示浏览器的版本信息，当服务器收到浏览器的请求后会经过一系列处理，向浏览器返回一个数据包。但是，当我们用 Python 而不是浏览器请求信息时，网站很可能会因为请求者不是浏览器而拒绝发送数据，这就需要我们“装扮身份”，模拟成浏览器请求数据，由此需要改变 User-Agent 参数。这段参数因每个电脑设置而有所不同。

```
headers = {
    'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/80.0.3987.122 Safari/537.36',
    'Host': 'book.douban.com'
}
```

如何获取 User-Agent 参数呢？如果使用的是 Chrome 浏览器，可以尝试以下操作：

首先，右键点击网页进入“检查”。接着，在检查窗口顶部导航栏找到 Network 并点击。随后，任意点击网页，可以看到 Network 界面发生了变化，与服务器通信的数据包出现，显示在 Name 这一栏内。继而任意打开一个数据包，查看导航栏 Headers 下的信息，鼠标滚动下拉找到“Request Headers”栏下的“User-Agent”，这里就是当前浏览器的信息（见图 3-29）。

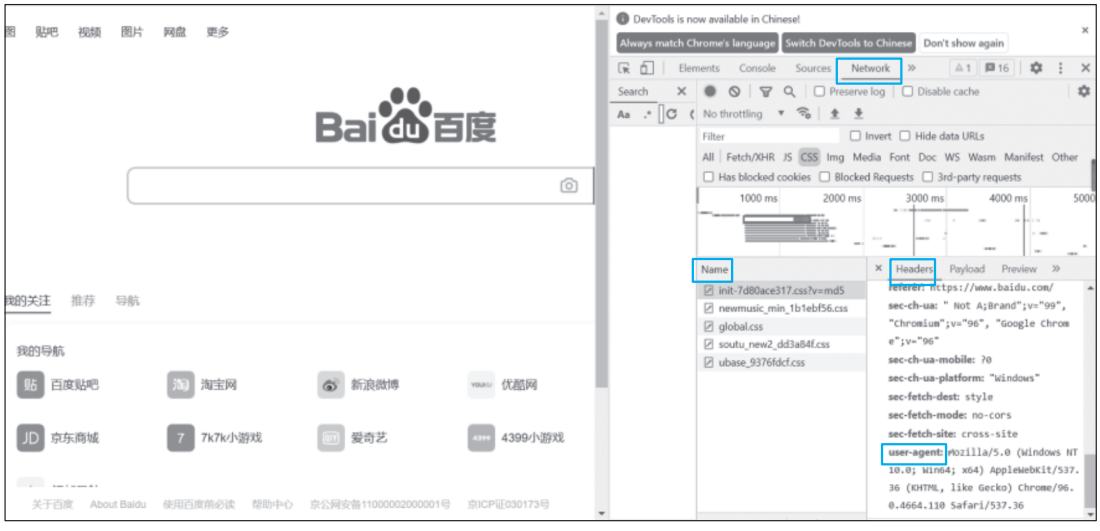


图 3-29　查找 User-Agent 参数

收到服务器返回的回应之后，使用 Beautiful Soup 进行解码，具体使用的是 lxml 解析器，将解析后的数据存储到变量 Soup 中。

```
soup = BeautifulSoup(res.text, "lxml")
```

3. 数据抓取

接下来，要根据网页 DOM 结构进行数据抓取，也就是找到数据存储的具体标签。打开豆瓣读书 Top250 网页，可以看到所需抓取的每一条图书信息对应着一个属性名为“item”的“tr”标签（见图 3-30），也就是说一本书是表格中的一个单元格。

图 3-30　查看网页 DOM 结构

接着使用 find_all 函数，将 Soup 中所有属性名为“item”的“tr”找到，将所有结果以列表的形式存到变量“books_list”中。

```
books_list = soup.find_all('tr', class_='item')
```

接着键入变量名称——books_list，返回列表内容，如图 3-31 所示：

```
In [10]: books_list

Out[10]: [<tr class="item">
          <td valign="top" width="100">
          <a class="nbg" href="https://book.douban.com/subject/1007305/" onclick="moreurl(this,{i:'0'})">
          <img src="https://img1.doubanio.com/view/subject/s/public/s1070959.jpg" width="90"/>
          </a>
          </td>
          <td valign="top">
          <div class="pl2">
          <a href="https://book.douban.com/subject/1007305/" onclick="&quot;moreurl(this,{i:'0'})&quot;" title="红楼梦">
                        红楼梦


                      </a>


                            <img alt="可试读" src="https://img3.doubanio.com/pics/read.gif" title="可试读"/>
          </div>
          <p class="pl">[清] 曹雪芹 著 / 人民文学出版社 / 1996-12 / 59.70元</p>
```

图 3-31 得到“books_list”列表

上一步已抓取了每个单元格的内容，下一步是提取单元格中特定的字段，我们想要抓取的是书名和评分。分析网页源代码，可以看到书名对应的代码为 div class="pl2"，评分对应的代码为 span class="rating_nums"（见图 3-32）。

图 3-32 书名和评分对应的代码

遍历“books_list”的所有单元格，找到每个单元格内的书名和评分，将其存成“book”对象，将“book”对象使用 append 函数添加到“books”列表中。

```
books = []
for element in books_list:
    title = element.find('div', class_='pl2').a.text
    title = title.replace("\n","").replace(" ","")
rating = element.find('span', class_='rating_nums').text
book = {
        "title": title,
        "rating": rating
```

```
            }
    books.append(book)
```

上述代码中，replace 函数表示替换数据中的元素，它的书写逻辑为 replace（" 被替换元素 "，" 替换后的元素 "）。运行代码后，即可看到这一页的书名和评分信息都被抓取下来（见图 3-33）。

```
In [15]: books
Out[15]: [{'title': '红楼梦', 'rating': '9.6'},
          {'title': '活着', 'rating': '9.4'},
          {'title': '百年孤独', 'rating': '9.3'},
          {'title': '1984', 'rating': '9.4'},
          {'title': '飘', 'rating': '9.3'},
          {'title': '三体全集:地球往事三部曲', 'rating': '9.4'},
          {'title': '三国演义（全二册）', 'rating': '9.3'},
          {'title': '白夜行', 'rating': '9.1'},
          {'title': '房思琪的初恋乐园', 'rating': '9.2'},
          {'title': '福尔摩斯探案全集（上中下）', 'rating': '9.3'},
          {'title': '小王子', 'rating': '9.0'},
          {'title': '动物农场', 'rating': '9.3'},
          {'title': '撒哈拉的故事', 'rating': '9.2'},
          {'title': '天龙八部', 'rating': '9.1'},
          {'title': '安徒生童话故事集', 'rating': '9.2'},
          {'title': '平凡的世界（全三部）', 'rating': '9.0'},
          {'title': '围城', 'rating': '8.9'},
          {'title': '局外人', 'rating': '9.0'},
          {'title': '霍乱时期的爱情', 'rating': '9.0'},
```

图 3-33　抓取书名和评分信息

目前已经实现抓取第一页的书单信息，如果想要抓取后面几页，就需要进行翻页。用 Request 和 Beautiful Soup 的方式抓取数据与 Selenium 不同，不能通过 DOM 结构找到翻页按钮的标签，然后点击进行翻页。Request 模块抓取数据时要通过构建 url 的方式进行翻页。点击按钮翻页，本质上就是跳转了一个链接，只不过有的链接打开新页面，有的在同一页面展示新数据。观察豆瓣读书 Top250 不同页码地址可以发现，url 整体结构没有变化，只有“start=”后面的数字变了，第一页对应“start=0”，第二页对应“start=25”，第三页对应“start=50”，以此类推。因此我们只需要通过拼接的方式，构建出每一页的网址链接，再通过 request.get 函数发送请求至网址，即可完成数据抓取。具体 Python 代码如下：使用 for 语句循环遍历第 1 页到第 10 页，用“+”号拼接字符串形成网址，逐一对每一页的内容进行抓取，全部存到 books 列表中。最后展示 books 列表的长度，结果为 250，说明成功抓取了 Top250 的全部书单信息。

```
    books = []
    for i in range(0, 10):
        res = requests.get('https://book.douban.com/top250?start=' + str(i *
25), headers=headers, timeout=10)
        soup = BeautifulSoup(res.text, "lxml")
        movies_list = soup.find_all('tr', class_='item')
        for element in movies_list:
            title = element.find('div', class_='pl2').a.text
            title = title.replace("\n","").replace(" ","")
            rating = element.find('span', class_='rating_nums').text
        book = {
```

```
        "title": title,
        "rating": rating
            }
        books.append(book)

len(books)
```

结果为：250

上述代码中引入了两个 Python 自带的函数——range 和 len。函数 range 一般与 for 语句连用，表示创建一个整数列表，用于 for 循环中。上面例子中 range(0,10) 表示选取 0 到 10 之间的整数，但不包含 10，也就是选取“0、1、2、3、4、5、6、7、8、9”。函数 len 意为返回对象的长度或项目个数，示例中返回了 books 列表的项目个数。

四、向浏览器发送信息获取特定数据

在网络数据抓取时偶尔需要输入内容以获取特定数据，这就需要编写代码向浏览器发送数据。

复旦大学新闻学院“数据分析与信息可视化”课程中曾有学生做了一幅上海夏季小龙虾吃货地图，需要抓取大众点评网站上海地区小龙虾餐厅的信息。如果手动检索就需要在搜索框中输入“小龙虾”三个字，并点击搜索按钮；如果使用代码控制浏览器进行操作，只需找到搜索框的源代码，向该区域发送信息即可，具体代码如下：

```
driver.find_element（By.CLASS_NAME, 'J-search-input').send_keys('小龙虾')
```

分析网页源代码发现搜索框的源代码为 class="J-search-input"，通过查找元素锁定搜索框，随后向搜索框输入信息，使用“send_keys( )”代码。

随后控制浏览器进行点击操作，找到“搜索”按钮对应的代码“class="J-all-btn"”，使用“click( )”命令使浏览器进行搜索操作。

```
driver.find_element（By.CLASS_NAME, 'J-all-btn').click()
```

特别需要注意的是，当使用查找元素引用源代码时，必须与源代码完全一致，最好使用拷贝操作，如出现偏差则会导致代码无法正常运行。

思考与练习

1. 使用后羿采集器获取微博热搜页面的热搜议题。

2. 使用 Python 编程语言的 Selenium 库编写代码抓取豆瓣读书 Top250 网站中全部书籍的书名以及评分。

3. 使用 Python 编程语言 Beautiful Soup 库抓取豆瓣电影 Top250 网址中全部电影的名称以及评分。

4. 2017 年 5 月 2 日新华社发布如下信息，请依据文中信息找到起火点的地理位置数据。

内蒙古大兴安岭毕拉河北大河林场发生森林火灾

2017-05-02 17:41:27 来源：中国林业网

中国林业网 5 月 2 日 16 时 56 分讯 据国家森林防火指挥部办公室消息，接内蒙古自治区森防指办公室报告：

5 月 2 日 12 时 15 分，内蒙古大兴安岭毕拉河北大河林场发生森林火灾。经飞机观察，火场有五个火头，火势较强，面积约 500 公顷。目前，内蒙古大兴安岭森防指共调集 1385 名扑救人员（森警 455 人）赶赴火场，其中已有 100 名专业扑火队员进入火场开始扑救。国家森防指紧急调派 4 架直升机（M-26、K-32、EC-225、AS-350 各 1 架）参加扑救。

国家森防指副总指挥、国家林业局副局长李树铭，内蒙古自治区副主席张华，武警森林指挥部副司令员郭建雄等领导正赶赴内蒙古大兴安岭毕拉河林业局北大河林场火场，协调指导火灾扑救工作。

国家森防指、国家林业局要求内蒙古森防指高度重视，重兵投入，尽快扑灭火灾；加强值班调度和火情报送工作。（防火办）

参考答案