



Python应用实战

爬虫、文本分析与可视化

▶▶▶▶ 张丽 张鹏 彭笛 编著



教学资料

 中国工信出版集团

 电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>



Python应用实战

爬虫、文本分析与可视化

张丽 张鹏 彭笛 编著

电子工业出版社
Publishing House of Electronics Industry
北京 • BEIJING



内 容 简 介

欢迎来到 Python 的世界。本书介绍了 Python 的语法、数据结构等基础知识，以及经典的 Python 爬虫、网页文本分析及可视化。在本书中，读者不仅可以与 Python “结识”，还会遇到新“朋友”——浏览器的开发者工具，通过它来了解 HTML 编写网页的语言，并进行结构化的网页分析和所需数据的提取。

拿来主义特别适合来类比 Python 语言中的库，Python 将与 re、requests、lxml 等经典的库组合在一起，自动抓取网页数据的爬虫。Pandas 这个工具会对抓取的数据进行文本分析，并实现将枯燥的数据进行漂亮的可视化呈现。

千里之行，始于足下，欢迎进入本书的奇妙之旅。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Python 应用实战：爬虫、文本分析与可视化 / 张丽，张鹏，彭笛编著. —北京：电子工业出版社，2020.3

ISBN 978-7-121-38013-6

I. ① P… II. ① 张… ② 张… ③ 彭… III. ① 软件工具—程序设计—高等学校—教材 IV. ① TP311.561

中国版本图书馆 CIP 数据核字（2019）第 263832 号

策划编辑：章海涛

责任编辑：底 波

印 刷：北京京师印务有限公司

装 订：北京京师印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×1092 1/16 印张：11 字数：280 千字

版 次：2020 年 3 月第 1 版

印 次：2020 年 3 月第 1 次印刷

定 价：42.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：192910558（QQ 群）。

前言

打开本书，请记下今天的日期，同时记住 30 天内要完成本书内容的学习。那么 Python 是什么？为什么要学习 Python？我作为一名受益者，亲历了从一个编程小白到学会使用 Python 进行自动化的工具开发，以及编写自己工作中需要代码的过程。

大学学习 C 语言编程的时候，老师上课讲的知识我都听懂了，但我就是不会编写程序。这个问题困扰了我很久。现在回头想想学习编程有两种境界：痴迷和崩溃。我应该就是学到崩溃了吧。参加工作以后，我因为工作需要自学了 Python，看着编写的代码运行起来，心情也跟着放飞了，即使过程中遇到各种错误和异常状况，我也会专注地投入去解决问题。

初学编程语言很容易陷入复杂的逻辑中而一筹莫展，久而久之就会逐步放弃，所以选择一门好的编程语言就是成功的一半。使用 Python 编程不用很费劲就能实现想要的代码功能，编写的代码也清晰易懂，并且能够保持自己的编码风格。

本书围绕学会编程并能使用编程语言进行程序设计、围绕数据进行处理的主题，介绍编程和相关的知识。

学以致用是一条很幸福的学习道路。本书面向那些希望学习一门编程语言并想对数据进行处理的读者。如果你是 Python 语言的小白，那么请从第 1 章开始学习；如果你已经具备 Python 编程的基础，那么请跳过第 1 章，从第 2 章开始学习。仅凭几行代码搞定复杂的文本处理任务，是不是很酷？快速进行项目实战，就不会失去学习的兴奋点。

保持幽默是 Python 语言和社区的传统，你在学习的过程中，输入 `import this` 后就会体验到了。那么就按照 `import this` 输出的箴言前进吧。好了，祝你编程愉快，30 天后见。

本书特色

本书能让你在 30 天内将 Python、HTML、爬虫、数据抓取、文本分析和数据可视化等技术，从应用流程上将各个知识点串联起来。30 天后，你将有如下收获。

1. 获得 Python 语言的基础技能，学会用程序员的思维来处理问题。
2. 了解 HTML，学会使用强大的网页分析工具，轻松获取网页中的数据。在面对大量数据时，会借助 Python，学习怎样自动抓取。
3. 当抓取的数据（文本）杂乱无章时，文本分析的方法可对数据进行清洗，你将了解到正则表达式的强大。
4. 当干净的数据被导入后，你会学习分析这些数据，将枯燥的数据转化为可视化的、生动的图片。

通过上述学习之后，你能够对 Python 工具生态圈有一个完整的认识，了解自己在这个生态圈中的定位，决定自己后续的升级（学习）方向。

作者

本书配套教学资源



全书彩色图片



全书源代码

目 录

第1章 初识 Python	1	2.8 以 URL 结束	52
1.1 使用 IDLE	1	2.9 本章总结	55
1.2 从字符串着手	4	第3章 数据抓取	56
1.3 复杂数据的福音——列表	7	3.1 工具准备	56
1.3.1 创建列表	7	3.2 XPath 和 lxml.html	58
1.3.2 列表的操作	7	3.2.1 网页分析利器——lxml	58
1.4 处理数据——条件判断	9	3.2.2 XPath	59
1.5 处理数据——循环	11	3.2.3 XPath 使用实例	60
1.6 处理数据进阶——嵌套语句	12	3.2.4 XPath 演示	61
1.7 函数	14	3.3 关于 robots.txt	62
1.8 拿来就用——模块	16	3.4 小试牛刀	64
1.9 文件	17	3.4.1 过程分析	64
1.10 处理异常	18	3.4.2 动手敲代码	67
第2章 网页	20	3.4.3 小结	68
2.1 工具准备	20	3.4.4 扩展	68
2.2 从 URL 开始	21	3.5 获取电影数据（上）	69
2.2.1 简单获取 URL	22	3.5.1 过程分析	70
2.2.2 链接与 URL	24	3.5.2 动手敲代码	73
2.3 编写网页的语言——HTML	25	3.5.3 小结	74
2.3.1 创建自己的第一个网页	26	3.6 获取电影数据（下）	75
2.3.2 标签——创建网页的方块	27	3.6.1 过程分析	76
2.3.3 标签属性	30	3.6.2 动手敲代码	76
2.4 CSS 与 class	31	3.6.3 考虑加强代码的健壮性	78
2.5 JavaScript 和 id	33	3.6.4 小结	80
2.6 网页分析工具	36	3.7 另类的网页抓取	80
2.6.1 谷歌开发者工具	36	3.7.1 过程分析	81
2.6.2 查看网页结构	38	3.7.2 动手敲代码	84
2.6.3 定位指定的元素	39	3.7.3 小结	85
2.6.4 筛选不同的资源	41	3.8 爬虫与网络机器人	85
2.7 网页的快递——HTTP	44	3.9 本章总结	86
2.7.1 HTTP 请求	45	第4章 文本处理	87
2.7.2 HTTP 响应	46	4.1 正则表达式	87
2.7.3 HTTP 的应用——Cookie 和 Session	47	4.1.1 怎样进行匹配	87
2.7.4 实战——HTTP 的交互过程	49	4.1.2 常用的元字符	88

4.2 更强的文本工具——Python 的 re 库	89	5.2.3 对 Series 进行一些操作	124
4.2.1 匹配对象怎么用	91	5.2.4 方法串联	128
4.2.2 使用 regex 来搜索	91	5.2.5 操作 Series 中的字符串数据	129
4.2.3 使用 regex 来替换	93	5.2.6 小结一下 Series	130
4.2.4 更方便查找	95	5.3 DATAFRAME	131
4.2.5 re 库中的控制标志	95	5.3.1 创建 DataFrame	132
4.2.6 replace()和 re.sub()	98	5.3.2 对齐	133
4.2.7 实现更高级的 strip()方法	99	5.3.3 了解 DataFrame	134
4.2.8 新的拆分方法 re.split()	100	5.3.4 常用 DataFrame 操作	137
4.2.9 怎样提取中文	101	5.3.5 数据的导入与导出	141
4.3 电影数据的处理	102	5.4 简单数据分析	145
4.3.1 提取之前的观察	104	5.4.1 电影评分分布	145
4.3.2 需要获取哪些数据	104	5.4.2 电影产量趋势	146
4.3.3 多样化的方法	111	5.4.3 评论人数最多的电影	147
4.3.4 格式化的数据	112	5.4.4 发行电影最多的国家	148
4.4 本章总结	115	5.5 看得见的数​​据	153
第 5 章 数据分析	116	5.5.1 线图	153
5.1 工具准备	116	5.5.2 柱状图	155
5.1.1 配置 Jupyter Notebook	116	5.5.3 饼图	157
5.1.2 数据生成帮手——Numpy	116	5.6 matplotlib	158
5.1.3 Pandas 中的数据结构	118	5.6.1 绘图方法	158
5.2 像一维数组的 SERIES	118	5.6.2 子图形及布局	160
5.2.1 获取 Series 信息	120	5.6.3 图形大小、颜色和样式	163
5.2.2 Series 进行数学运算	123	5.7 画一张图来结尾	165
		5.8 本章总结	167

第1章 初识 Python

什么是 Python?

很多人学习一门语言，大多会问这种语言有什么与众不同？也许读者拿起这本书也会问为什么要学习 Python 呢？简单地回答，编程能控制计算机实现自己的一些想法，是一种美好的体验。首先 Python 作为一门编程语言，与其他编程语言一样，具有数据类型、操作符、语句、函数、模块、类等通用的内容。除此之外，简单易学的特点使 Python 非常适合作为进入编程世界的第一门语言。结合 Python 丰富的库资源等，尽可能让计算机来执行工作任务，工作效率会高很多。目前，Python 在科学计算、机器学习、人工智能等领域得到了广泛的应用。当然，本书的目标就是让读者在案例中学习和领会 Python 的过人之处。其图标如图 1-1 所示。

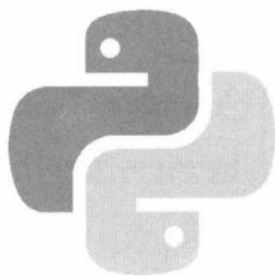


图 1-1



笔记栏

www.python.org 是 Python 的官方网站，可在此获得 Python 的官方文档以及解释器。

Python 是解释性的语言，不是编译性的语言。因此执行 Python 代码时，代码直接运行而未加密。

1.1 使用 IDLE

“工欲善其事，必先利其器。”在正式编程之前先安装 Python 软件。Windows、Linux、macOS 是目前最受欢迎的三种操作系统。其中，Windows 系统的用户比例是最大的。对于初学者来说，在 Windows 系统中安装 Python 及一些科学包是很容易出现问题的，这将使得初次接触 Python 就挫败感十足。这里，我们介绍一种集成的开发环境（Integrated Development Environment, IDE）——Anaconda。其图标如图 1-2 所示。IDE 具有语法高亮显示、自动补全、错误突出显示和检查等功能，使用 IDE 开发代码的效率会更高。Python 是具有代码块缩进规则的语言，刚开始使用 Python 时，可能很难适应这点，而 IDE 能很清楚地记住 Python 缩进的语法，根据需要自动缩进。



图 1-2



问题来了

问：为什么不使用 PyCharm 作为本次学习的 IDE?

答：PyCharm 也是 Python 编程中常用的 IDE，在编译、运行代码时可以体现其优势。Anaconda 是开源的、专注于数据科学与分析的 Python 发行版本。Anaconda 包含了大量的科学

包或者模块，初学者可以很容易地安装所需要的科学包或者模块，不会被困在环境的准备上。选择一种好的工具，学习的道路就不会那么曲折了。

根据计算机的操作系统和系统位数，可通过网址 <https://www.anaconda.com/download> 下载 Python 3 的匹配安装文件。

在安装 Anaconda 时根据提示，依次单击“Next”按钮即可完成安装。这里说明两个安装中的注意事项。

① 选择目标文件夹，即选择希望安装到哪个文件夹下（选择的文件夹需为空文件夹），如图 1-3 所示。

② 在安装前，需要勾选如图 1-4 所示的选项。

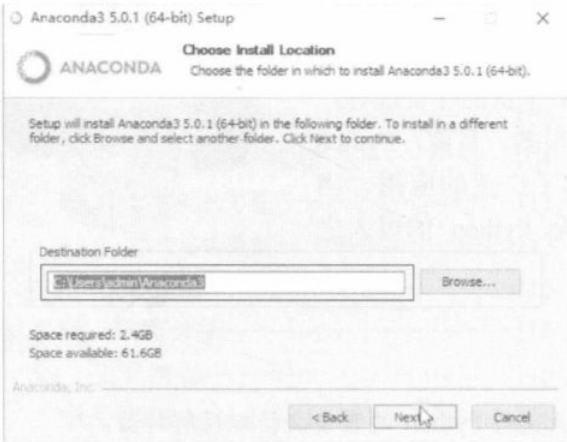


图 1-3

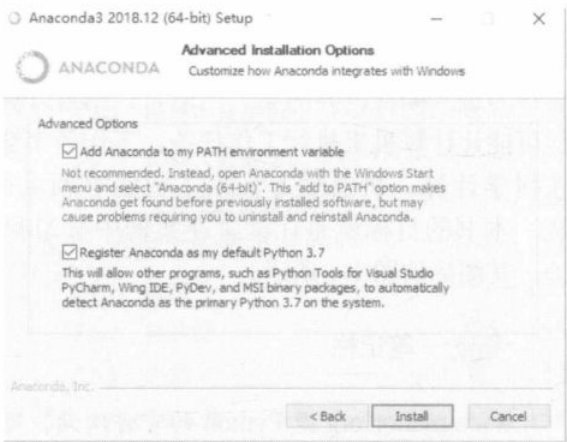


图 1-4

安装完成后，在 cmd 命令行中输入“Anaconda”，即可打开 Anaconda，如图 1-5 所示。双击 Jupyter 图标，Jupyter 会在默认的网络浏览器中打开，地址是 <http://localhost:8888/tree>。Jupyter 是交互式笔记本，能提供强大的交互能力。按照如图 1-6～图 1-9 所示的步骤学习 Jupyter 的使用，创建第一个编程文件吧。



图 1-5



图 1-6



图 1-7



图 1-8

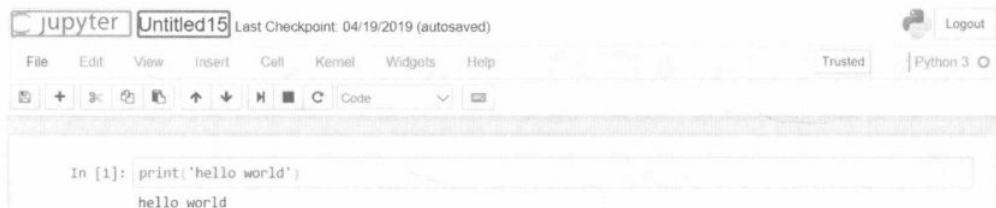


图 1-9

按照上面的提示，在代码编辑框中编写自己的第一行 Python 代码并运行查看结果吧。

随着编写的代码越来越多，在 Jupyter 下保存代码的方式是必须知道的。单击图 1-9 中的右框区域，即可按照提示修改保存。单击图 1-9 中的左框区域，即可查看保存的文件，我们会看到保存的文件是扩展名为.ipynb 的代码文件。



知识库

ipynb 文件全称为 ipython notebook document，以.ipynb 结尾的代码文件可用 Jupyter notebook

打开并运行，计算机通过这样的扩展名能识别出该文件是专有的文件，而不是普通的文本文件。



问题来了

问：为什么有时候按照上述步骤双击 Jupyter 图标后，无法打开 Jupyter？

答：双击 Jupyter 图标后，在图 1-10 所示的终端/命令行中会生成一个 URL，为带有令牌密钥提示。需要将包含这个令牌密钥在内的整个 URL（如图 1-10 中框内所示示意内容）都复制并粘贴到你的浏览器地址栏中，然后才能打开一个 Jupyter Notebook。此步骤执行一次即可，后续将无须再次执行。

```
I 20:03:25.418 NotebookApp] JupyterLab alpha preview extension loaded from C:\Users\admin\Anaconda3\lib\site-packages\jupyterlab
JupyterLab v0.27.0
Known labextensions:
I 20:03:25.429 NotebookApp] Running the core application with no additional extensions or settings
I 20:03:25.629 NotebookApp] Serving notebooks from local directory: C:\Users\admin
I 20:03:25.629 NotebookApp] 0 active kernels
I 20:03:25.630 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=0872529cf8678c794190898bc18a016ffied22a2268e72b1
I 20:03:25.630 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
C 20:03:25.641 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=0872529cf8678c794190898bc18a016ffied22a2268e72b1
I 20:03:26.191 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

图 1-10

1.2 从字符串着手

先从一本书名入手——The Kite Runner（追风筝的人）。Python 理解的这本书的方式为：

In [1]: books='The Kite Runner'

创建字符串，需要以下 2 个步骤。

- ① 数据两边需加上成对的英文模式单引号 (') 或者双引号 (")，将书名转换为字符串。
- ② 通过等号赋值操作符 (=) 将字符串赋值为变量标识符。



知识库

字符串 (String) 是由字母、数字、文字、下画线等字符组成的一串字符。一个词、一句话是字符串，一段文本也是字符串，甚至一篇文档也可以看成一个字符串。当然，单个的字符是最小的字符串。

在处理文本数据和解析格式时，是可以对字符串进行大量操作的，常见的操作有以下几种。

(1) 索引和切片

使用中括号偏移量的方法来访问字符串中的数据项，Python 中的偏移量是从 0 开始计数的。

动手敲代码

① 使用 print() 内建函数，显示字符串，并使用 len() 内建函数得出字符串含有多少个数据项。

```
In [1]: books='The Kite Runner'
```

```
In [2]: print(books)
The Kite Runner
```

```
In [3]: print(len(books))
15
```

② 索引：通过编号访问数据。访问第一个数据项：

```
In [4]: print(books[0])
T
```

③ 切片：访问一部分数据。访问的方法：

```
In [5]: print(books[1:5])
He K
```

```
In [6]: print(books[-3:-1])
ne
```

这里访问提取的字符串从索引开始直到结束，但提取不包含结束的索引上的字符。

笔记栏

字符串中的每个字符都有特定的位置，称为索引。Python 中具有两种索引方式：正向索引和反向索引。正向索引从 0 开始，每个字符每次加 1；反向索引从-1 开始，每个字符每次减 1。访问字符串的一部分，称为切片。

Python 中有大量的内建函数（BIF），常见的内建函数见表 1-1。如代码练习中使用到了 2 个内建函数 `print()`、`len()`，我们就可以直接使用这些 BIF，让编程过程尽可能简单。通过 `dir(__builtins__)` 方法可以查看当前 Python 版本中具体的 BIF。在没有熟悉这些 BIF 之前，看到这么多的 BIF 是一种负担，先从如下的 BIF 练习吧。

表 1-1

内建函数	BIF 功能
<code>range()</code>	生成指定范围的数字，for 循环中经常会被使用到
<code>str()</code>	返回字符串类型的对象，如 <code>str(1)</code> 返回值为字符串类型 '1'
<code>input()</code>	接收任意输入，将所有输入默认按照字符串处理，并返回字符串类型
<code>list()</code>	将元组或者字符串转换成列表
<code>max()</code>	返回给定参数的最大值

笔记栏

内建函数为编程语言预先定义的函数，多为实现数学运算、类型转换、序列操作、对象操作、反射操作、变量操作等方面的功能。

(2) 方法

Python 中提供庞大的内置方法用于执行字符串的转换、控制和操作。

动手敲代码

- ① 使用 `upper()` 方法，将字符串中所有的字符都变为大写字符。

```
In [10]: books.upper()
Out[10]: 'THE KITE RUNNER'
```

- ② 使用 `replace()` 方法，替换字符串。

```
In [12]: books.replace('Runner', 'Walking')
Out[12]: 'The Kite Walking'
```

- ③ 使用 `split()` 方法，切分字符串。

```
In [16]: books.split(' ')
Out[16]: ['The', 'Kite', 'Runner']
```

注意，上面示例的引号中预留了一个空格，即按照空格进行切分。计算机很容易识别这个空格，但目测是很难发现这个空格的。使用 `split()` 不带参数的写法 `books.split()` 即实现按照空格的方式进行字符串切分，并返回列表（后面会学习这种数据类型）。

- ④ 使用 `strip()` 方法，移除字符串头尾字符。

```
In [19]: books.strip('T')
Out[19]: 'he Kite Runner'
```

- ⑤ 使用 `replace()` 方法，把字符串中原有字符串替换为新字符串。

```
In [27]: books.replace('n', 'T')
Out[27]: 'The Kite RuTter'
```

上面的示例中将全部的 'n' 字符进行替换，如果对替换次数有要求，则可以指定第三个参数 `count`，替换不超过 `count` 次。例如：

```
In [28]: books.replace('n', 'T', 1)
Out[28]: 'The Kite RuTner'
```



笔记栏

读者在没有进行大量的代码编程前，学习每个方法将超出当前的认知范围。使用句点 (.) 进行方法的连接，是 Python 语法的一个特点。

(3) 格式化

字符串格式化的功能强大，可以替换字符串中特定的内容。在处理文本时，格式化是最常用的方式。格式化表达式和格式化方法函数是两种类型的格式化。

动手敲代码

- ① 格式化表达式：使用格式化运算符 % 进行字符串替换。

```
In [11]: 'The kite %s'%('Ruuner')
Out[11]: 'The kite Ruuner'
```

```
In [12]: 'The kite %d'%(5)
```

```
Out[12]: 'The kite 5'
```

其中，`%s` 和 `%d` 分别替换字符串和整数的占位符。

② 格式化方法函数：使用 `format()` 函数、通过运算符 `{}` 替代运算符 `%` 来实现字符串的格式化。

```
In [14]: 'The kite {}'.format(5)
```

```
Out[14]: 'The kite 5'
```

```
In [15]: 'The kite {}'.format('Runner', 5)
```

```
Out[15]: 'The kite Runner 5'
```

```
In [16]: 'The kite {1}{0}'.format('Runner', 5)
```

```
Out[16]: 'The kite 5 Runner'
```

1.3 复杂数据的福音——列表

字符串类型数据因为数据简单，是很容易直接进行数据处理的。实际上大部分数据都是比较复杂的，通常需要将数据通过某种方式组织在一起，按照一定的数据结构来处理，以降低数据的复杂性，列表是常用的数据类型。

1.3.1 创建列表

也是先从练习中熟悉的那本书入手吧。这本书的一些信息为：The Kite Runner（追风筝的人）、作者卡勒德·胡塞尼、长篇小说、226000 字。Python 按照列表的数据结构理解的这本书的信息，如图 1-11 所示。

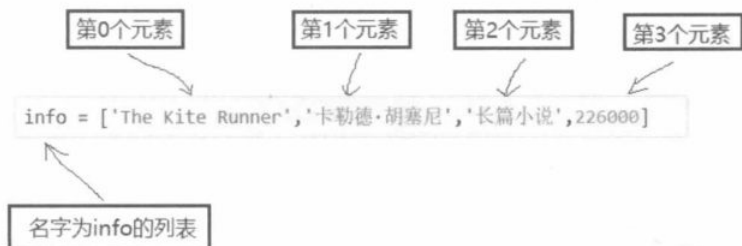


图 1-11

创建列表，需要以下 4 个步骤。

- ① 所有数据均放在方括号内，形成列表。
- ② 各个元素间使用逗号分隔。
- ③ 元素的表达遵循各自的语法规则，如字符串类型数据需加上引号，数字类型数据直接表示，列表则需在方括号内（列表中的元素可以是列表数据类型。）
- ④ 通过等号赋值操作符（`=`）将列表赋值为变量标识符。

1.3.2 列表的操作

列表是可以像字符串一样进行索引和切片的，本小节将讨论列表不同于字符串的地方，以

及列表的专有方法，以体现列表的强大之处。

动手敲代码

① 使用 `append()`方法，增加列表数据，并使用 `len()`内建函数得出字符串含有多少个数据项。

```
In [17]: info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
In [18]: info.append('美籍阿富汗人')
In [19]: print(info)
['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000, '美籍阿富汗人']
```

② 使用 `insert()`方法，可在特定的位置前增加数据。

```
In [17]: info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
In [21]: info.insert(3, '上海人民出版社')
In [22]: print(info)
['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', '上海人民出版社', 226000, '美籍阿富汗人']
```

③ 使用 `remove()`方法，删除特定的数据。

```
In [17]: info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
In [23]: info.remove('长篇小说')
In [24]: print(info)
['The Kite Runner', '卡勒德·胡塞尼', '上海人民出版社', 226000, '美籍阿富汗人']
```


笔记栏

列表可使用的方法有很多，一时是很难全部记住的。通过输入“列表名.”使用 `Tab` 键的方式，可以查看到当前可用的方法。如上述示例中，输入 `info.`后，通过 `Tab` 键可查看 `info` 列表可用的方法，如图 1-12 所示。

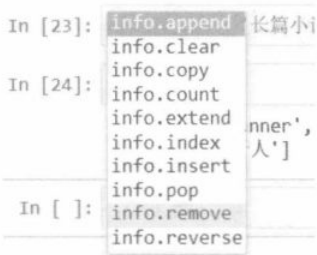


图 1-12

使用 `help()`函数，可查看各种方法的用法。在熟能生巧前，这是一种快速上手的办法。推荐的方法是从官方的文档中查询和学习方法及其语法和定义。例如：

```
In [25]: help(info.pop)
Help on built-in function pop:

Pop(...)method of builtins.list instance
L.pop([index])-> item -- remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.
```



阶段小练习

在下面列表中，插入出版年份“2003”。

```
info = ['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
```

添加你的代码：



小练习之答案

在下面列表中，插入出版年份“2003”。

```
info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
```

添加你的代码：

```
info.insert(2,2003)
```

```
info.append(2003)
```

1.4 处理数据——条件判断

列表是常用的数据，如果要对列表数据进行一些处理操作，则需要使用逻辑结构。Python 通过 `if...else...` 语法结构进行判断，这种结构称为条件判断。通常用于表示类似“如果……否则……”的逻辑。程序会根据输入来判断执行程序中不同的语句块。

关键字 `if` 作为条件判断的开始，冒号 (`:`) 作为条件判断的结尾。`if` 和冒号之间的内容是条件判断的语句。`if` 和判断语句之间需留有一个空格。冒号后的内容为语句块，需 4 个空格缩进，缩进结束处表示语句块的结束位置。



问题来了

问：什么是关键字？

答：Python 共有 33 个关键字，这些关键字是一种特殊的标识符，它们的名字已经被占用。自定义的变量名、函数名等不得与已有的关键字的名字相同。

```
In [7]: import keyword
print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

条件满足，即判断语句结果为真 (`True`) 时，执行 `True` 条件下的语句块；条件不满足，即判断语句为假 (`False`) 时，执行 `False` 条件下的语句块，从而不会执行 `True` 条件下

的语句块。

条件判断的格式如图 1-13 所示。



图 1-13

动手敲代码

在列表中查询指定的内容。

```
info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]  
if '长篇小说' in info:  
    print('这本书是长篇小说')  
else:  
    print('这本书非长篇小说')  
这本书是长篇小说
```

 笔记栏

- 1. True 和 False 在 Python 中表示两种状态——真与假，类似是与否。
- 2. 程序的缩进是 Python 代码编写的重要规则，用来告诉 Python 代码的起始位置，缩进结束处表示语句块的结束位置。

 阶段小练习

在下面列表中，插入出版年份信息“2003 年”。

```
info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]  
  
添加你的代码:
```

 小练习之答案

在下面列表中，插入出版年份信息“2003 年”。

```
info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]  
  
添加你的代码:
```

```
info=['The Kite Runner', '卡勒德·胡塞尼', '长篇小说', 226000]
```