1

编程简介

深入探讨 Python 编程之前，先大致了解一下 Python 是什么及其可用于编写哪些类型的程序，这大有裨益。本章还将概述程序员所做的工作。最后将介绍如何安装 Python 及运行其自带的 IDLE 编辑器。

如果你是编程新手，本章可助你为学习 Python 编程语言做好准备。

如果你已掌握这些基本概念，可跳到有关如何安装 Python 和运行编辑器的章节。

本章内容

- Python 语言
- Python 适合用于做什么
- 程序员如何工作
- 安装 Python

1.1 Python 语言

那么 Python 是什么呢？简单地说，它是一种计算机编程语言及一组配套的软件工具和库。Python 最初由 Guido van Rossum 于 20 世纪 90 年代初开发，当前由世界各地的数十位程序员（包括 van Rossum）负责维护。

Python 易于理解和学习。相比于用其他大多数编程语言编写的程序，Python 程序更整洁：Python 几乎没有多余的符号，且使用的是简单易懂的英语名称。

Python 语言的效率极高。精通 Python 后，与使用其他大多数编程语言相比，使用 Python 可在更短的时间内完成更多的工作。Python 支持但不强制你使用面向对象编程（OOP）。

Python 自带了各种现成库，供你在自己的程序中使用。有些 Python 程序员喜欢这样说：Python“开箱即可使用”。

Python 的一个极其实用的特点是易于维护。鉴于 Python 程序理解和修改起来相对容易，程序员可轻松地确保它们紧跟潮流。在程序员所做的工作中，程序维护所占的比例很可能高达甚至超过 50%，因此在很多专业人士看来，Python 对维护的支持是个亮点。

最后，说说名称 Python 的由来。据 Python 之父 Guido van Rossum 说，Python 是以喜剧团体 Monty Python（巨蟒小组）的名字命名的。虽然这种起源充满喜庆色彩，但 Python 当前使用的标识确乎是两条缠在一起的蛇（可能是蟒蛇），其中一条为蓝色，另一条为黄色。

1.2 Python 适合用于做什么

虽然 Python 是一种通用语言，可用于编写任何类型的程序，但它最常用于编写下述应用程序。

- **脚本**。这些简短的程序自动执行常见的管理任务，如在系统中新增用户、将文件上传到网站、在不使用浏览器的情况下下载网页等。
- **网站开发**。作为快速创建动态网站的工具，Django（www.djangoproject.com）、Bottle（www.bottlepy.org）和 Zope（www.zope.org）等众多 Python 项目深受开发人员的欢迎。例如，深受欢迎的新闻网站 www.reddit.com 就是使用 Python 开发的。
- **文本处理**。Python 在字符串和文本文件处理方面提供了强大的支持，包括正则表达式和 Unicode。
- **科学计算**。网上有很多卓越的 Python 科学计算库，提供了用于统计、数学计算和绘图的函数。
- **教育**。鉴于 Python 简洁实用，越来越多的学校将其作为第一门编程教学语言。

当然，Python 并非对任何项目来说都是最佳选择，其速度通常比 Java、C#、C++ 等语言慢，因此开发新操作系统时不会使用 Python。

然而，需要最大限度地减少程序员花在项目上的时间时，Python 通常是最佳选择。

1.3 程序员如何工作

虽然对如何编写程序没有严格的规定，但大多数程序员都采用类似的流程。

该程序开发流程如下。

1. 确定程序要做什么，即搞清楚需求。

2. 编写源代码，这里是使用 Python 集成开发环境 IDLE 或其他文本编辑器编写 Python 代码。这一步通常最有趣也最具挑战性，要求你创造性地解决问题。Python 源代码文件使用扩展名 .py，如 web.py、urlexpand.py、clean.py 等。

3. 使用 Python 解释器将源代码转换为目标代码。Python 将目标代码存储在 .pyc 文件中，例如，如果源代码存储在文件 urlexpand.py 中，目标代码将存储在文件 urlexpand.pyc 中。

4. 运行或执行程序。就 Python 而言，通常紧接着第 2 步自动完成这一步。实际上，Python 程序员很少直接与目标代码（.pyc 文件）交互。

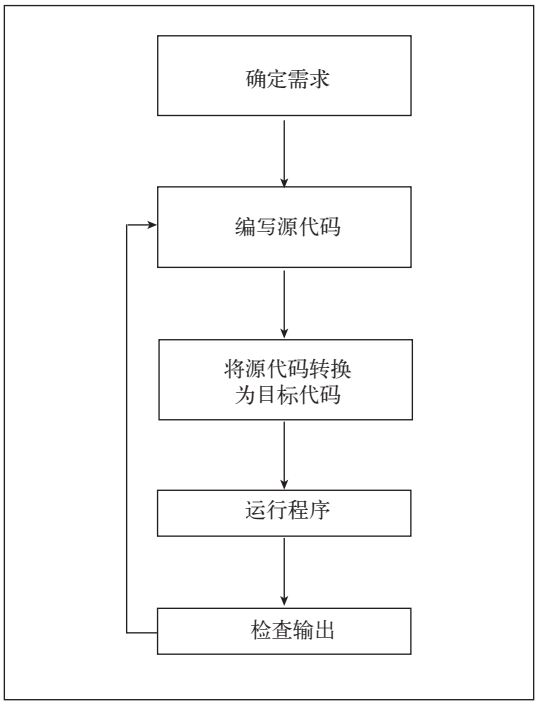

图 1-1　基本计算机程序编写步骤。检查程序输出时，通常会发现错误。为修复错误，必须回到步骤“编写源代码”

术语说明

我们通常将 .py 文件的内容称为程序、源代码或代码。

目标代码有时也称为可执行代码、可执行文件或软件。

5. 最后，检查程序的输出。如果发现错误，回到第 2 步并尽力修复错误。修复错误的过程称为调试。开发庞大或复杂的程序时，可能大部分时间都用在调试上，因此经验丰富的程序员设计程序时，会尽力采用可最大限度地减少调试时间的方式。

如图 1-1 所示，这是个循环往复的过程：编写程序，测试，修复错误，再测试……直到程序正确运行。

1.4 安装 Python

Python 是一种实践性语言，下面来看看如何在计算机上安装它。

1.4.1 在 Windows 系统上安装 Python

步骤如下。

1. 访问 Python 下载页面 www.python.org/download。

2. 选择最新的 Python 3 版本（其名称类似于 Python 3.x，其中 x 是一个较小的数字），这将打开相应的下载页面，其中说明了如何下载用于不同计算机系统的 Python。

3. 根据计算机使用的操作系统，单击相应的安装程序链接。例如，如果是 Windows 操作系统，单击 Windows x86 MSI Installer (3.x)。

4. 下载完毕后，双击安装程序以运行它。

5. 安装完成后（需要几分钟），通过测试看看是否正确安装了 Python。为此，打开“开始”菜单并选择“所有程序”，将看到一个与 Python 3.0 相关的选项（其背景通常为黄色）。选择其中的选项 IDLE (Python GUI)，一段时间后程序 IDLE 将启动，如图 1-2 所示。

6. 输入 24 * 7 并按回车，应出现数字 168。

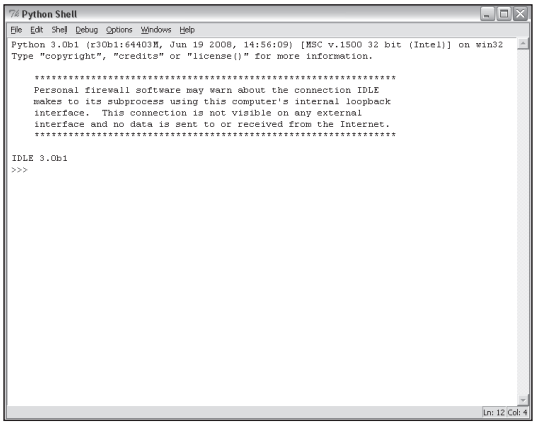

图 1-2 IDLE 编辑器的起始屏幕。第一行指出了当前使用的是哪个 Python 版本（这里为 3.0b1）

1.4.2 在 Mac 系统上安装 Python

OS X 自带并安装了一个 Python 版本，但该版本没有 IDLE 编辑器，通常也不是最新版本。要安装更新的 Python 版本，可按 www.python.org/download/mac/ 给出的说明做，也可从 www.pythonmac.org/packages/ 下载一个安装程序并运行它。下载安装程序时，务必选择正确的 Python 版本（3.0 或更高版本），并确保 Mac OS 版本号与你的操作系统版本号一致。

1.4.3 在 Linux 系统上安装 Python

如果你使用的是 Linux，很可能已经安装了 Python。要确认这一点，可打开命令行窗口并输入 python，如果输出与图 1-2 类似，说明 Python 运行正常。

务必检查版本号。本书介绍的是 Python 3，如果当前安装的是 Python 2.x 或更早的版本，就应安装 Python 3。

具体如何安装因 Linux 系统而异。例如，在 Ubuntu Linux 系统上，需要在 Synaptic Package Manager 中搜索 Python。你也可以访问 www.python.org/download，了解如何在 Linux 系统上安装 Python。

3

编写程序

当目前为止，我们编写的都是单行Python语句，并通过交互式命令行运行它们。这对于学习Python函数虽然很有用，但当需要编写大量Python代码行时，就很烦琐了。

因此，现在开始转而编写程序（也叫脚本）。程序不过是文本文件，但包含一系列Python命令。当你运行（或执行）程序时，Python依次执行文件中的每条语句。

在本章中，你将学习如何在IDLE中编写程序，以及如何从IDLE和命令行运行程序。你将明白如何获取用户通过键盘提供的输入以及如何将字符串打印到屏幕上。

你应该尽力亲手输入代码，因为这是熟悉各种Python编程规则的最佳方式。对于较大的程序，可从本书的配套网站下载代码，网址为http://pythonintro.googlecode.com。

本章内容

- 使用IDLE的编辑器
- 编译源代码
- 从键盘读取字符串
- 在屏幕上打印字符串
- 源代码注释
- 程序的组织

3.1 使用 IDLE 的编辑器

IDLE 自带了一个明白 Python 的文本编辑器。要学习这个编辑器，最佳的方式是编写一个简单程序。

3.1.1 在 IDLE 中编写程序

在 IDLE 中编写程序的步骤如下。

1. 启动 IDLE。

2. 选择菜单 File > New Window。

3. 输入下面的代码：

```
print('Welcome to Python!')
```

4. 选择菜单 File > Save 将程序存盘。将其存储在你的 Python 程序文件夹中，并命名为 welcome.py；末尾的 .py 表明这是一个 Python 文件。

5. 选择菜单 Run > Run Module 运行程序。

将出现一个 Python shell，其中显示了“Welcome to Python!”。

熟悉 IDLE 编辑后，你可能想使用表 3-1 列出的一些快捷键，它们确实能提高编辑速度。

表 3-1 一些实用的 IDLE 快捷键

命 令	作 用
Ctrl-N	打开一个新的编辑器窗口
Ctrl-O	打开一个文件进行编辑
Ctrl-S	保存当前程序
F5	运行当前程序
Ctrl-Z	撤销最后一次操作
Shift-Ctrl-Z	重做最后一次操作

提示 在计算机桌面上创建一个特殊文件夹，并将其命名为 python，用于存储你的所有 Python 程序。不要将它们存储到 Python 安装目录中，否则将面临无意间覆盖 Python 核心文件的风险。

提示 你必须按原样输入 Python 程序，一个字符都不差。哪怕错一个字符（多一个空格、将数字 1 错误地输入为字母 l），都可能导致错误。

提示 如果你运行程序时看到错误消息，请返回到编辑器窗口，并逐字符地仔细核对程序，确保输入正确。

其他编辑器

对初学者来说，IDLE 是一款杰出的编辑器，连一些专业人员都经常使用它。然而，如果你不喜欢 IDLE，可在网上使用 programming editors(编程编辑器)进行搜索，你将得到大量的建议。例如，在 Windows 系统上，Notepad++ 是一款深受欢迎的编程编辑器，而且是免费的。另一款深受欢迎的编辑器是 Sublime Text，它有 Windows、Mac 和 Linux 版本，但不是免费的。

要获悉众多其他建议，请参阅 http://wiki.python.org/moin/PythonEditors。

3.1.2 从命令行运行程序

另一种运行 Python 程序的常见方式是，从命令行运行。例如，要运行 welcome.py，可以打开命令行窗口，并执行下述命令：

```
C:\> python welcome.py
Welcome to Python!
```

也可以只调用 Python，而不指定程序，这将打开交互式解释器的缩简版本，但依然很有用。

3.1.3 从命令行调用 Python

执行如下命令：

```
C:\> python
Python 3.0b2 (r30b2:65106, Jul 18 2008,
→ 18:44:17) [MSC v.1500 32 bit (Intel)]
→ on win32
Type "help", "copyright", "credits" or
→ "license" for more information.
>>>
```

将 Python 脚本作为其他程序的一部分进行运行时，通常从命令行调用 Python。

提 示 在 Windows 系统中，打开命令行窗口的最简单方式是，单击“开始”按钮，在“运行”对话框中输入 cmd 并按回车键。这将打开一个命令行窗口。

提 示 在 Mac 和 Linux 系统中，从命令行运行 Python 程序的方式相类似：打开命令 shell（具体如何打开随系统而异，但你可通过桌面上的菜单浏览可用的程序），再输入 python 以及要运行的程序的名称。

提 示 从命令行运行 Python 时，令人讨厌的一点是通常需要配置环境变量（具体地说是系统的路径变量），让系统知道到计算机的什么地方去寻找 Python。关于具体如何配置的介绍过于烦琐且随系统而异，这超出了本书的范围。然而，如果你要配置，很容易在网上找到详细的说明。例如，只需在喜欢的搜索引擎中输入 set windows path 即可。修改环境变量时务必小心：如果你拿不准该如何做，很可能破坏系统，导致程序再不能正确运行。在这种情况下，最佳的选择是从头再来并重新安装 Python。

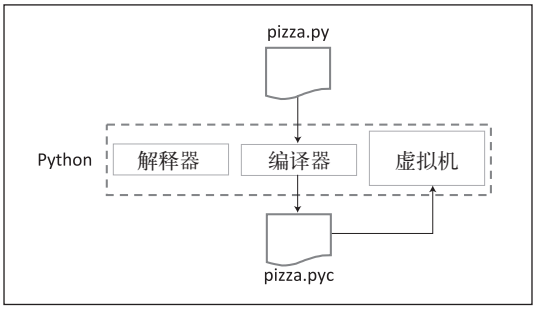

图 3-1　Python 由 3 个主要部分组成：运行语句的解释器、将 .py 文件转换为 .pyc 文件的编译器以及运行 .pyc 文件的虚拟机。请注意，严格地说 IDLE 并非 Python 的一部分，它是一个位于 Python 之上的独立应用程序，旨在让 Python 使用起来更容易

3.2　编译源代码

我们经常将 Python 程序中的语句称为源代码，并将程序文件称为源代码文件。根据约定，所有 Python 源代码文件都使用扩展名 .py。这让人和程序一眼就能明白文件包含 Python 源代码。

目标代码

当运行 .py 文件时，Python 会自动创建相应的 .pyc 文件，如图 3-1 所示。.pyc 文件包含目标代码（编译后的代码）。目标代码基本上是一种 Python 专用的语言，以计算机能够高效运行的方式表示 Python 源代码。这种代码并不是供人类阅读的，因此在大多数情况下你都应对 .pyc 文件置之不理。

Python 程序是使用名为虚拟机的特殊软件运行的。这个软件模拟计算机，是专为运行在 Python 上而设计的，这让很多 .pyc 文件无需做任何修改就能在不同的计算机系统上运行。

提示　你几乎不用关心 .pyc 文件。Python 在需要时会自动创建它们，并在你修改了相应的 .py 文件时自动更新它们。千万不要删除、重命名或修改 .pyc 文件！

提示　鉴于 .pyc 文件仅供计算机阅读，因此它们不是文本文件。如果你试图在文本编辑器中查看 .pyc 文件，看到的将是一堆乱码。

3.3 从键盘读取字符串

从键盘读取字符串是从用户那里获取信息的一种最基本的方式。例如，请看下面这个简单程序：

```
# name.py
name = input('What is your first name? ')
print('Hello ' + name.capitalize()+ '!')
```

要在 IDLE 中运行它，请在 IDLE 窗口中打开 name.py，再按 F5（或选择菜单 Run > Run Module）。此时将出现一个窗口：

```
What is your first name? jack
Hello Jack!
```

你（用户）必须输入名字（这里为 jack）。

3.3.1 跟踪程序

下面来仔细研究这个程序的每一行。第 1 行是源代码注释（简称为注释）。注释不过是给程序员阅读的说明，Python 对其置之不理。Python 注释总是以符号 # 打头，并延续到行尾。这里的注释指出，这个程序存储在文件 name.py 中。

第 2 行调用函数 input，这是用于从键盘读取字符串的标准内置函数。这行代码执行时，将在输出窗口中显示 What is your name? 和闪烁的光标。程序等待用户输入一个字符串并按回车。

函数 input 返回用户输入的字符串，让变量 name 最终指向用户输入的字符串。

该程序的第 3 行（也是最后一行）显示一句问候语。函数 name.capitalize() 确保字符串的第一个字符为大写，而其他字符为小写。这样，如果用户输入的名字没有采用正确的首字母大写方式，Python 将更正。

提示 要获悉字符串包含哪些函数，可在 IDLE 的交互式命令行输入 dir('')。

提示 如果你多次运行程序 name.py，且每次都输入不同的字符串，将很快发现当你输入类似于 'Jack Aubrey' 的姓名时，姓的首字母将被转换为小写：'Hello Jack aubrey!'。这是因为函数 capitalize 的头脑非常简单，根本没有单词和空格的概念。

提示 从键盘读取字符串时，另一种常见的实用技巧是，使用函数 strip() 将开头和末尾的空白字符删除，如下所示：

```
>>> ' oven '.strip()
'oven'
```

因为经常需要删除不需要的空白，所以我们常常像下面这样调用函数 input：

```
name = input('Enter age: ').strip()
```

3.3.2 从键盘读取数字

函数 input 只是返回字符串，因此如果你需要的是数字（如用于算术运算），就必须使用 Python 的数值转换函数之一。例如，请看下面的程序：

```
# age.py
age = input('How old are you today? ')
age10 = int(age) + 10
print('In 10 years you will be ' +
→ str(age10) + ' years old.')
```

假设运行该程序时用户输入 22，变量 age 将指向字符串 '22'，因为 Python 不会自动将看起来像数字的字符串转换为整数或浮点数。如果你要将字符串用于算术运算，必须先将其转换为数字。为此，可使用函数 int(s)（如果你需要的是整数）或 float(s)（如果你需要的是浮点数）。

这里要指出的最后一个技巧是，在 print 语句中，必须将变量 age10（它指向一个整数）转换为字符串，这样才能打印它。如果你忘记这样做，Python 将显示错误消息，指出不能将数字与字符串相加。

不同类型的数字

一开始，各种不同的数值类型令你迷惑。请看下面 4 个不同的值：5、5.0、'5' 和 '5.0'。虽然它们看起来相似，但内部表示截然不同。

5 是一个整数，可直接用于算术运算。

5.0 是一个浮点数，也可用于算术运算，但包含小数部分。

'5' 和 '5.0' 都是字符串，分别包含 1 个和 3 个字符。字符串可显示到屏幕上或用于基于字符的操作（如删除空白或计算字符数）。字符串不能用于算术运算。当然，字符串可用于拼接，虽然结果可能让人觉得有点不合情理。例如：

```
>>> 3 * '5'
'555'
>>> 3 * '5.0'
'5.05.05.0'
```

术语说明

程序员常常使用术语标准输出（stdout）来表示文本被打印到的窗口。通常，标准输出是简单的文本窗口，几乎只显示字符串，不显示任何类型的图形。

同样，标准输入（stdin）是函数 input 从中读取字符串的地方，通常是标准输出对应的窗口，但必要时可更改标准输入和标准输出。

你有时还会看到输出标准错误（stderr），它指的是显示错误消息的地方。默认情况下，错误消息通常在标准输出中显示。

3.4 在屏幕上打印字符串

print 语句是用于将字符串打印到屏幕的标准内置函数。正如你将看到的，它非常灵活，有很多功能可用于正确地设置字符串和数字的格式。

你可将任意数量的字符串传递给 print：

```
>>> print('jack', 'ate', 'no', 'fat')
jack ate no fat
```

默认情况下，print 在标准输出窗口中打印每个字符串，并用空格分隔它们。修改字符串分隔符很容易，可以像下面这样做：

```
>>> print('jack', 'ate', 'no', 'fat',
→ sep = '.')
jack.ate.no.fat
```

默认情况下，print 打印完指定内容后添加一个换行符：\n。换行符导致光标移到下一行，因此默认情况下，调用 print 后不能在同一行打印任何内容：

```
# jack1.py
print('jack ate ')
print('no fat')
```

上述代码打印两行文本：

```
jack ate
no fat
```

要在同一行打印所有文本，可将第一行的结束字符指定为空字符串：

```
# jack2.py
print('jack ate ', end = '')
print('no fat')
```

提示 Python 2 和 Python 3 的主要差别之一就表现在函数 print 上。在 Python 2 中，print 从技术上说并非函数，而是语言的一部分。这带来的优点之一是，你可以不输入圆括号，例如，可以输入 print 'jack ate no fat'。然而，print 不是函数虽然带来了这种小小的便利，但导致修改默认分隔符和结束字符串非常困难，而在比较复杂的程序中常常需要这样做。

提示 Python 2 和 Python 3 的另一个不同之处在于，Python 3 函数 input 对应于 Python 2 函数 raw_input；Python 2 也有函数 input，但它对用户输入的字符串求值，这在有些情况下非常方便。在 Python 3 中，没有与 Python 2 函数 input 等价的函数，但可轻松地模拟这个函数——使用 eval(input(prompt))。例如：

```
>>> eval(input('? '))
? 4 + 5 * 6
34
```

3.5 源代码注释

前面使用了源代码注释来指出文件名，但注释还可用于在程序中添加各种说明，如文档、提示、解释或警告。Python 忽略所有注释，它们仅供你和其他可能阅读源代码的程序员阅读。

下面的示例程序演示了注释的其他一些用途：

```
# coins_short.py
# This program asks the user how many
# coins of various types they have,
# and then prints the total amount
# of money in pennies.
# get the number of nickels, dimes,
# and quarters from the user
n = int(input('Nickels? '))
d = int(input('Dimes? '))
q = int(input('Quarters? '))
# calculate the total amount of money
total = 5 * n + 10 * d + 25 * q
# print the results
print() # prints a blank line
print(str(total) + ' cents')
```

3.6 程序的组织

随着编写的程序越来越多，你将很快发现它们通常采用相同的结构。人们通常按图3-2所示的方式组织程序：包含输入部分、处理部分和输出部分。

在我们刚开始编写的小型程序中，这种结构通常显而易见，无需做太多的考虑。但随着程序越来越大、越来越复杂，很容易偏离这种总体结构，其结果常常是代码混乱、难以理解。

因此，应该养成良好的习惯——使用注释指出输入、处理和输出部分。这有助于阐明程序执行的不同任务。当你开始编写函数时，将发现这种结构提供了很好的指导，让你能够将程序合理地划分成多个函数。

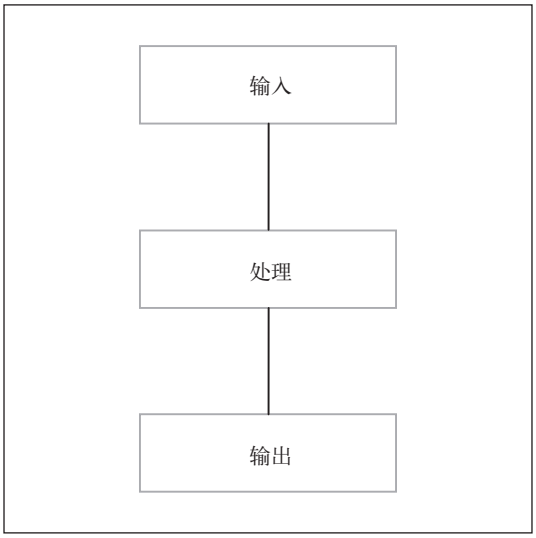

图 3-2　大多数程序都采用这里所示的结构：首先获取输入（例如，使用函数 input 从用户那里获取），然后对输入进行处理，最后向用户显示结果