第5章 程序思维——程序设计基础(Python)

"Everybody in this country should learn how to program a computer… because it teaches you how to think."

——Steve Jobs(史蒂夫·乔布斯)

Life is short, you need Python.

——Bruce Eckel(MindView 公司总裁,
ANSI/ISO C++ 标准委员会拥有表决权的成员之一)

5.1 Python 起步

人类有了语言和文字后才有了蓬勃的文明发展;同样计算机也有了计算机语言后才有了与计算机沟通的程序。计算机语言、算法和程序是三位一体的,计算机语言是工具,算法是解题思路,是程序设计的灵魂,程序是用某种计算机语言来实现算法的技术。

一个算法若用计算机语言来书写,则它就是一个程序。计算机程序设计是为计算机规划、安排解题步骤的过程,一般来说由以下 4 个步骤组成:

(1) 分析问题:在着手解决问题之前,要通过分析来充分理解问题,明确原始数据、解题要求、需要输出的数据及形式等。

(2) 设计算法:首先进行算法的总体规划,然后逐层降低问题的抽象性,逐步充实其细节,直到最终把抽象的问题具体转化为算法。

(3) 编码:用计算机语言表示算法的过程称为编码。程序是用计算机语言编码的解题算法。

(4) 调试与测试程序:调试的任务是排除编码错误,保证程序稳定的运行,并对程序的局部功能和性能进行检查。测试的任务是排除逻辑错误和系统设计错误,对程序进行系统全面的检查,保证程序整体的功能和性能。

Python 具有简单易学,免费、开源,可扩展性、可嵌入性和可移植性强,代码规范、代码具有较好可读性,Python 标准库很丰富,支持命令式编程、面向对象程序设计、函数式编程、泛型编程等多种编程范式等特点,因此,可以说 Python 的设计哲学是"优雅""明确"和"简单"。

5.1.1 Python 的版本与环境搭建

1. Python 2.x 与 3.x 版本

Python 的版本，目前主要分为两大类：Python 2.x 版本的，被称为 Python2，例如 Python 2.7.3；Python 3.x 版本的，称为 Python3，例如 Python 3.5.2。现在大多数第三方库都相容 Python 3.x 版本。因此建议使用 3.x 版本，本书使用的是 Python 3.5.2。

2. Python 的环境搭建

Python 可应用于包括 Linux 和 Mac OS X 多平台，这些系统已经自带 Python 支持，不需要再配置安装了。可以通过终端窗口输入 python 命令来查看本地是否已经安装 Python 以及 Python 的安装版本，然后根据需要升级或安装。

(1) Python 下载。打开 Python 官网主页 https://www.python.org/后选择适合自己的版本下载并安装即可。本书所有示例均在 Windows 8 上使用 Python 3.5.2 进行开发和演示。

(2) Windows 平台上安装 Python。在 Window 平台上安装 Python 的简单步骤如下：

- 打开 Web 浏览器访问 https://www.python.org/download/。
- 在下载列表中选择 Window 平台安装包，包格式为 python-XYZ.msi 文件，其中 XYZ 为版本号；
- 下载后，双击下载包，进入 Python 安装向导，首先建议选择第二项自定义安装(Custom installation)，并选择最下面的 Add Python 3.5 to PATH 选项，如图 5.1 所示。此选项的功能是把 Python 的安装路径添加到系统路径下面，选中这个选项的话，安装好后就不用设置路径了，以后在命令行输入命令 python 就可调用 python.exe；否则会报错。

图 5.1 Python 安装界面

- 选择第二项自定义安装(Custom installation)后,就可以自行设置安装位置,可以设置简单一点的路径,例如,这里设置为 C:\Python35。

安装好 Python 后,在“开始”菜单选择 IDLE 命令,即可启动 Python 解释器并可以看到当前安装的 Python 版本号,如图 5.2 所示。

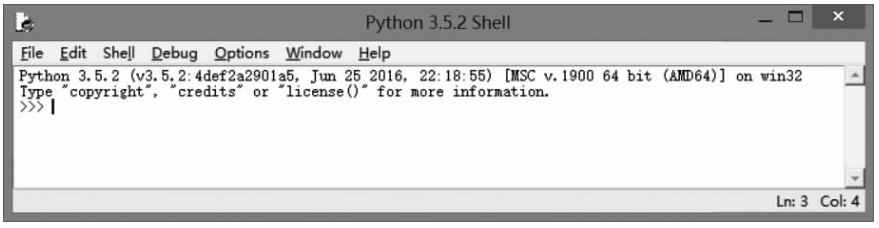

图 5.2 Python 3.5.2 主界面

注意:>>>是提示符,表示可以在它后面输入要执行的语句。在本书的示例中,带有符号>>>的代码是指在 IDLE 交互环境下运行的代码;不带此提示符的表示是以脚本程序的方式运行的。

(3) 环境变量配置。程序和可执行文件可以在许多目录,而这些路径很可能不在操作系统提供可执行文件的搜索路径中。path(路径)存储在环境变量中,这是由操作系统维护的一个命名的字符串。这些变量包含可用的命令行解释器和其他程序的信息。

在环境变量中添加 Python 目录的方法有以下两种。

方法一:

- 打开控制面板中的“系统”对话框。
- 选择“高级系统设置”。
- 在“系统变量”对话框中单击右下角的“环境变量”按钮。
- 然后在 Path 行,添加 Python 安装路径即可。注意路径直接用分号“;”隔开。这里设置为 F:\Python35,如图 5.3 所示。
- 最终设置成功以后,在 cmd 命令行下,输入命令 python,就可以有相关显示,如图 5.4 所示。

方法二:

在命令提示框中输入命令:

```
path=%path%;C:\Python35
```

注意:这里 C:\Python35 是 Python 的安装目录。

5.1.2 Python 的开发环境

Python 的开发环境有很多,主要有以下 3 种方法。

1. 使用 Python 的交互式解释器

从“开始”菜单→“所有程序”→Python 3.5→Python 3.5(64-bit),即可启动 Python

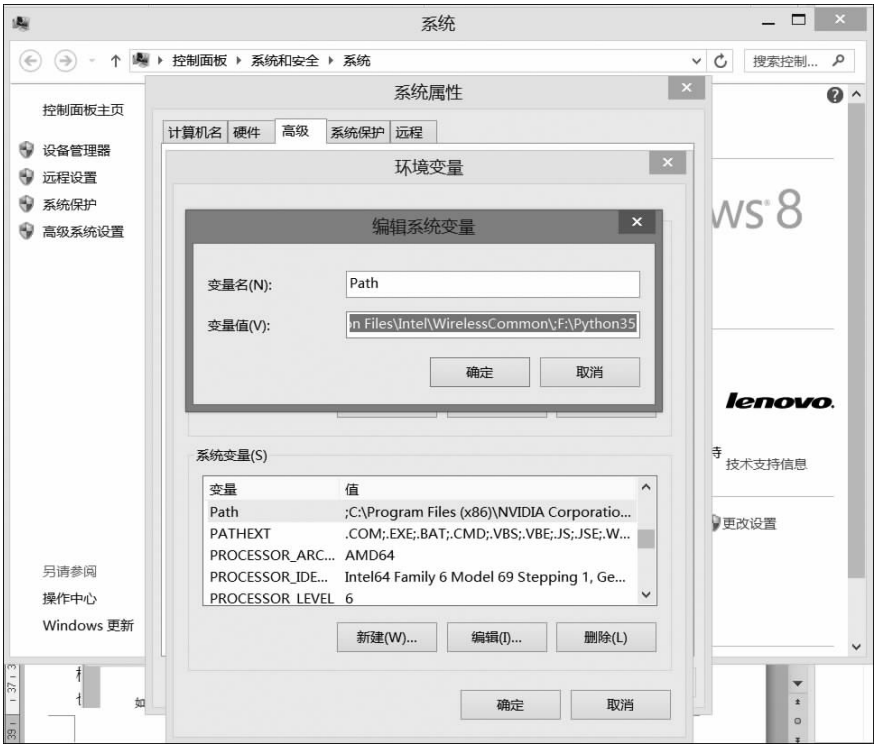

图 5.3　Python 环境变量设置

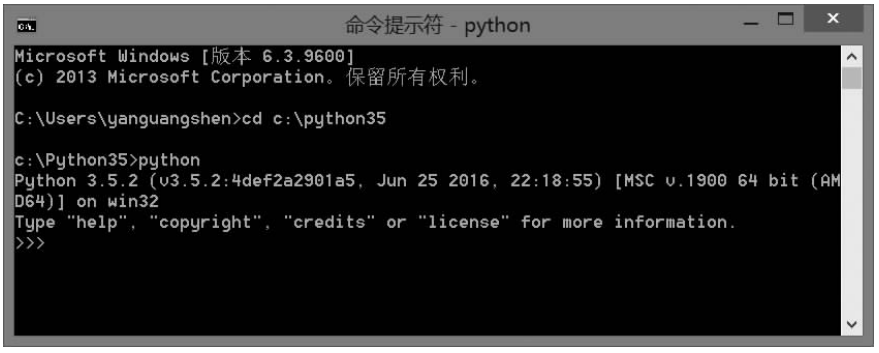

图 5.4　在命令提示符环境中输入命令

的交互式解释器窗口。在提示符>>>下输入 Python 语句，回车后，系统就会立即执行这条语句。

例如，输入

```
1+2
```

将会在下一行输出 3。很简单，任何有效的数学计算都可以算出来。

例如，输入

```
print("Hello World")
```

将会在下一行输出 Hello World，如图 5.5 所示。如果要让 Python 打印出指定的文字，可以用 print()函数，然后把希望打印的文字用单引号或者双引号括起来，但不能混用单引号和双引号。这种用单引号或者双引号括起来的文本在程序中称为字符串，今后我们还会经常遇到。

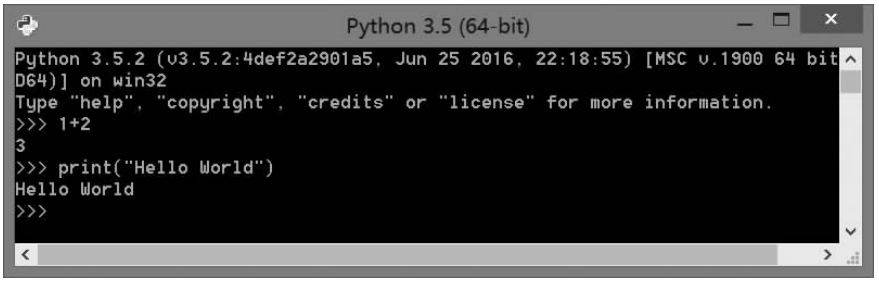

图 5.5　Python 的交互式解释器

使用 Python 的交互式解释器的优点是能马上看到每一行程序的结果。但是这样问答式的方式对于长程序就有些啰嗦了，所以这种方式通常用于短程序、进行简单计算、验证个别语句的语法等。

2. 使用记事本程序编辑 Python 程序

【例 5-1】　输出 Hello World。

以上面输出 Hello World 的例子为例，步骤如下：

(1) 使用记事本，输入以下内容(每行顶格写)：

```
def main():
    print('Hello World')
main()
```

(2) 将其保存到已有的文件夹 F：\eg 中，文件名为 01.py。

注意：保存时，选择保存类型为“所有文件(*.*)”，编码选择“UTF-8”。

(3) 选择开始菜单的“命令提示符”，在命令提示符环境下，使用 cd 命令进入 Python 安装所在文件夹目录下(这里是 cd c：\python35)，执行命令：

```
python ***.py
```

其中，***指的是要运行的程序所在的文件夹目录(这里是 python f：\eg\01.py)，如图 5.6 所示。

说明：在 Windows 资源管理器中双击.py 文件，也可以运行 Python 程序，只是这时有可能看到一个窗口很快一闪而过，这说明程序已经运行，只是输出速度太快，为了看到输出结果，可以在程序末尾加一个 input 函数，例如：

```
input("程序运行结束，按 Enter 键退出。")
```

图 5.6　在命令提示符环境中运行程序

3. 使用 IDLE 集成开发环境编写和执行 Python 程序

安装后默认使用 IDLE 为集成开发环境，下面重点介绍，本书均以 IDLE 为例。

(1) IDLE 的启动。IDLE 是与 Python 一起安装的，只要确保安装画面中选中了 Tcl/Tk 组件(默认时该组件是处于选中状态的)。

安装 Python 后，可以从“开始”菜单→“所有程序”→Python 3.5→IDLE 来启动 IDLE。

启动 IDLE 后，首先映入我们眼帘的是 Python Shell，通过它可以在 IDLE 内部执行 Python 命令。除此之外，IDLE 还带有编辑器、交互式解释器和调试器。IDLE 启动后的初始窗口如图 5.7 所示。

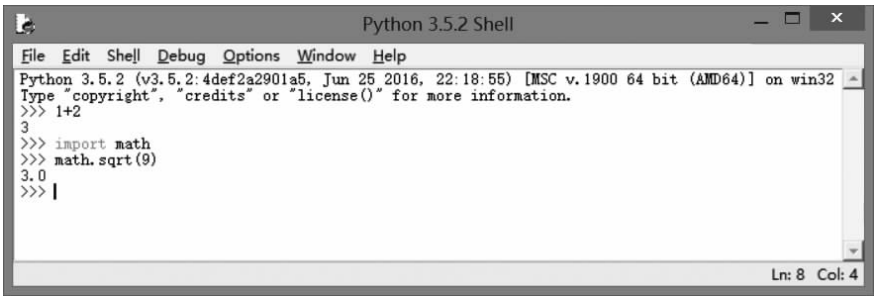

图 5.7　在命令提示符环境中输入命令

(2) IDLE 的交互编辑。打开 IDLE 后出现一个增强的交互命令行解释器窗口(具有比基本的交互命令提示符更好的剪切、粘贴等功能)。

在这个界面上即可进行简单交互程序的输入，如：

```
>>>1+2
3
>>>import math
>>>math.sqrt(9)
3.0
```

(3) IDLE 的文件编辑。如果需要编写大段的 Python 程序并复用，可以使用 IDLE 提供的文件编辑功能。

- 新建与编辑。按 Ctrl＋N 或在 IDLE 的 File 菜单中选择 New File，则会打开一个新的空白窗口，在此窗口中即可进行大段落编程，注意每行顶格写。

【例 5-2】 程序 02. py。

```
def main():
    print('Hello World')
main()
```

【例 5-3】 程序 03. py。

```
age=input("How old are you?")
height=input("How tall are you?")
weight=input("How much do you weigh?")
print("So,you're %r old,%r tall and %r heavy."%(age,height,weight))
```

- 保存和运行。当完成编辑后，请按 Ctrl＋S 或在 File 菜单中选择 Save 先保存文件。如果未保存直接运行将会出现提示，提醒用户请先保存。保存文件时，位置任意，但文件的扩展名必须为. py。

保存后，按 F5 键或选择 Run 菜单的 Run Module 进行运行。这时，如果程序无错误，即可在 IDLE 的交互编辑环境看到输出结果。由于此交互环境已经保存了刚运行的这个程序，所以可以继续在此交互环境中检查或者使用之前定义的变量、函数等信息，这对于调试程序非常有帮助。

(4) IDLE 的使用特性。IDLE 为开发人员提供了许多有用的特性，如自动缩进、语法高亮显示、单词自动完成以及命令历史等等，在这些功能的帮助下，能够有效地提高我们的开发效率。

- 缩进。当按 Enter 键之后，IDLE 自动进行了缩进。一般情况下，IDLE 将代码缩进一级，即 4 个空格。如果想改变这个默认的缩进量的话，可以从 Format 菜单选择 New indent width 项来进行修改。对初学者来说，需要注意的是尽管自动缩进功能非常方便，但是不能完全依赖它，因为有时候自动缩进未必完全合我们的心意，所以还需要仔细检查一下。
- 语法高亮显示。所谓语法高亮显示，就是给代码不同的元素使用不同的颜色进行显示，默认时，关键字显示为橘红色，注释显示为红色，字符串为绿色，定义和解释器的输出显示为蓝色，控制台输出显示为棕色。在输入代码时，会自动应用这些颜色突出显示。语法高亮显示的好处是，可以更容易区分不同的语法元素，从而提高可读性；与此同时，语法高亮显示还降低了出错的可能性。例如，如果输入的变量名显示为橘红色，那么就需要注意了，这说明该名称与预留的关键字冲突，所以必须给变量更换名称。
- 自动输入提示功能。自动输入提示是指当用户输入单词的一部分后，按 Alt＋/组合键或从 Edit 菜单选择 Expand word 项，IDLE 就能够根据语法或上下文来补全该单词。对于 Python 的关键字，例如函数，只需输入的开头一个或几个字母，然后在 Edit 菜单选择 Show completetions，IDLE 就会给出一些列表选项请用户选

择。IDLE 还可以显示语法提示，例如输入“print(”，IDLE 会弹出一个语法提示框，显示 print 函数的语法格式。

- 命令历史功能。命令历史可以记录会话期间在命令行中执行过的所有命令。在提示符下，可以按 Alt＋P 组合键找回这些命令，每按一次，IDLE 就会从最近的命令开始检索命令历史，按命令使用的顺序逐个显示。按 Alt＋N 组合键，则可以反方向遍历各个命令，即从最初的命令开始遍历。

(5) 使用 IDLE 的调试器。软件开发过程中，总免不了这样或那样的错误，其中有语法方面的，也有逻辑方面的。对于语法错误，Python 解释器能很容易地检测出来，这时它会停止程序的运行并给出错误提示。对于逻辑错误，解释器就鞭长莫及了，这时程序会一直执行下去，但是得到的运行结果却是错误的。所以，我们常常需要对程序进行调试。

- 最简单的调试方法是直接显示程序数据。例如，可以在某些关键位置用 print 语句显示出变量的值，从而确定有没有出错。但是这个办法比较麻烦，因为开发人员必须在所有可疑的地方都插入打印语句。等到程序调试完后，还必须将这些打印语句全部清除。
- 使用调试器来进行调试。利用调试器，可以分析被调试程序的数据，并监视程序的执行流程。调试器的功能包括暂停程序执行、检查和修改变量、调用方法而不更改程序代码等等。IDLE 也提供了一个调试器，帮助开发人员来查找逻辑错误。

下面简单介绍 IDLE 的调试器的使用方法。选择 Debug 菜单中的 Debugger 菜单项，就可以启动 IDLE 的交互式调试器。这时，IDLE 会打开 Debug Control 窗口，并在 Python Shell 窗口中输出[DEBUG ON]并后跟一个＞＞＞提示符。这样，就能像平时那样使用这个 Python Shell 窗口了。可以在 Debug Control 窗口查看局部变量和全局变量等有关内容。如果要退出调试器的话，可以再次单击 Debug 菜单中的 Debugger 菜单项，IDLE 会关闭 Debug Control 窗口，并在 Python Shell 窗口中输出[DEBUG OFF]。

在 IDLE 环境下，除了剪切(Ctrl＋X)、复制(Ctrl＋C)、粘贴(Ctrl＋V)、全选(Ctrl＋A)、撤消(Ctrl＋Z)等常规快捷键外，其他常用快捷键如下：

- Alt＋P：浏览历史命令(上一条)。
- Alt＋N：浏览历史命令(下一条)。
- Ctrl＋F6：重新启动 Shell，之前定义的对象和导入的模块全部失效。
- Alt＋/：自动单词完成，只要文中出现过，就可以帮你自动补齐。多按几次可以循环选择。
- Ctrl ＋]：缩进代码块。
- Ctrl ＋ [：取消缩进代码块。
- Alt＋3：注释代码行。
- Alt＋4：取消注释代码行。
- F1：打开 Python 帮助文档。

5.1.3 使用 pip 管理 Python 扩展库

pip 是一个安装和管理 Python 扩展库的工具。在 Windows 命令提示符环境下，可以使用 pip 来完成扩展库的安装、升级和卸载。在命令提示符输入：pip help，就可以看到所有 pip 命令及命令选项，如图 5.8 所示。

图 5.8　pip 命令

5.2 Python 编程基础

前面提到，传统程序的基本构成元素包括：常量、变量、运算符、内部函数、表达式、语句、自定义过程或函数等。现代程序增加了类、对象、消息、事件和方法等元素。高级语言体系和自然语言体系十分相似。计算机语言的学习过程一般是：基本符号及书写规则→常量、变量→运算符和表达式→语句→过程、函数→程序。

5.2.1 标识符和关键字

标识符是程序中用来表示变量、函数、类、模块和其他对象的名称。例如，在例 5-3 中 age、height、weight 都是标识符。

1. 标识符的命名规则

在 Python 中，标识符由字母、数字以及下画线组成，不能以数字开头，不能与 Python 中的关键字(保留字)相同。Python 中的标识符区分大小写，不限定长度。

注意：在 Python 3.x 中，标识符还可以使用 Unicode 编码的字母，即可以使用阿拉伯语、中文、日语或俄语字符或 Unicode 字符集支持的任意其他语言中的字符进行命名，但不建议这样使用。

2. Python 关键字

表 5.1 显示了在 Python 中的关键字。

注意：所有 Python 的关键字只包含小写字母。

表 5.1 Python 中的关键字

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	with
continue	except	global	lambda	raise	yield

在交互方式中输入 help()→keywords 可以进入帮助系统，查看所有的关键字列表，并可以根据提示查看某个关键字的说明信息。要退出帮助系统，使用 quit 命令。

3. 预定义的标识符

Python 中包含许多预定义的内置类、异常、函数等，如 float、input、print 等。用户应避免使用它们作为标识符。

在交互方式中输入 dir(_ _builtins_ _)可以查看所有的内置类、异常、函数名(注意