内 容 简 介

全书系统介绍了 Python 程序设计语言和程序设计的基本方法，按照由浅入深、由易到难、由理论到实践的原则进行编写。主要内容包括 Python 概述、Python 语言基础、Python 函数和高级特征、Python 函数式编程和模块、Python 的面向对象编程、Python 应用开发、网络编程之爬虫应用、Python 数据分析。

本书可作为“Python 程序设计”课程的教材，也可供相关技术人员参考。

图书在版编目(CIP)数据

Python 程序设计基础 / 罗少甫，谢娜娜主编. -- 北京：北京邮电大学出版社，2018.11(2023.7 重印)
ISBN 978-7-5635-5633-5

Ⅰ.①P… Ⅱ.①罗… ②谢… Ⅲ.①软件工具—程序设计—高等职业教育—教材 Ⅳ.①TP311.561

中国版本图书馆 CIP 数据核字(2018)第 264904 号

策划编辑：刘 建 **责任编辑：**边丽新 **封面设计：**刘文东

出版发行：北京邮电大学出版社
社 址：北京市海淀区西土城路 10 号
邮政编码：100876
发 行 部：电话：010-62282185 传真：010-62283578
E-mail：publish@bupt.edu.cn
经 销：各地新华书店
印 刷：大厂回族自治县聚鑫印刷有限责任公司
开 本：787 mm×1 092 mm 1/16
印 张：15 插页 1
字 数：365 千字
版 次：2019 年 1 月第 1 版
印 次：2023 年 7 月第 5 次印刷

ISBN 978-7-5635-5633-5 **定 价：**45.00 元

Preface 前 言

Python是一门易学且功能强大的编程语言，它拥有高效的高级数据结构和简单而有效的面向对象程序设计方法。Python优雅的语法和动态类型，结合它的解释执行特性，使它在大多数计算机平台和许多应用领域中成为脚本编写与快速应用开发的理想编程语言。

本书共分8个模块，推荐学时安排见下表。

模　块	内　容	学　时
1	Python概述	4
2	Python语言基础	10
3	Python函数和高级特征	10
4	Python函数式编程和模块	8
5	Python的面向对象编程	8
6	Python应用开发	8
7	网络编程之爬虫应用	6
8	Python数据分析	6
总计		60

本书主要特色如下。

(1)本书语言简练，内容由浅入深，在讲解上遵循人们对事物的认知习惯。另外，本书更加强调应用，除了讲解Python程序设计的基础知识外，还加强了Python应用方面的内容讲解，使学生在理解理论知识的同时，充分掌握其应用方法。

(2)将Python的基础知识和应用融入8个模块中，每个模块都通过问题引导的方式来加深读者对Python编程的了解。

(3)在每个模块中都有针对性地设置了实训练习，使读者摆脱枯燥的纯理论学习，适应教学发展需要，提升实际操作能力。

本书由罗少甫和谢娜娜任主编，由董明、赵波、郑剑、牟鑫任副主编。具体编写分工

如下:模块 1 和模块 5 由赵波编写,模块 2 由董明编写,模块 3 和模块 4 由谢娜娜编写,模块 6 由郑剑编写,模块 7 由牟鑫编写,模块 8 由罗少甫编写。全书由罗少甫统稿。本书在编写过程中得到了我校诸多同仁的大力支持和帮助,尤其得到了黄诚、杨光的鼎力帮助,在此一并表示衷心的感谢。

由于编者水平有限,书中难免存在不足之处,恳请广大读者批评指正。

编　者

Contents 目 录

模块 1

Python 概述

学习要点

- 什么是 Python?
- 如何搭建 Python 的简单开发环境?
- 了解 Python 的基本语法。

1.1 初识 Python

1.1.1 什么是 Python

Python 是一种面向对象的解释型计算机程序设计语言,由荷兰人 Guido van Rossum 于 1989 年发明,第一个公开发行版发行于 1991 年。Python 是纯粹的自由软件,源代码和解释器 CPython 遵循 GPL(GNU General Public License)协议。Python 语法简洁清晰,特色之一是强制用空白符(white space)作为语句缩进。Python 具有丰富和强大的库。它常被称为“胶水语言”,能够把用其他语言(尤其是 C/C++)制作的各种模块很轻松地连接在一起。常见的一种应用情形是,使用 Python 快速生成程序的原型(有时甚至是程序的最终界面),然后对其中有特别要求的部分用更合适的语言改写,如 3D 游戏中的图形渲染模块,性能要求特别高,就可以用 C/C++重写,而后封装为 Python 可以调用的扩展类库。需要注意的是,在使用扩展类库时可能需要考虑平台问题,某些可能不提供跨平台的实现。

1.1.2 Python 的特点

Python 具有如下一些特点。

(1)简单。Python 是一种代表简单思想的语言。

(2)易学。Python 有极其简单的语法。

(3)免费、开源。Python 是 FLOSS(自由/开放源码软件)之一。

(4)高层语言。使用 Python 编写程序时无须考虑如何管理程序使用的内存一类的底层细节。

(5)可移植性。Python 已被移植到很多平台,这些平台包括 Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE 甚至还有 Pocket PC。

(6)解释性。可以直接从源代码运行。在计算机内部,Python 解释器把源代码转换为字节码的中间形式,然后把它翻译成计算机使用的机器语言。

(7)面向对象。Python 既支持面向过程编程,也支持面向对象编程。

(8)可扩展性。部分程序可以使用其他语言编写,如 C/C++。

(9)可嵌入性。可以把 Python 嵌入 C/C++程序中,从而提供脚本功能。

(10)丰富的库。Python 标准库很庞大。它可以帮助用户处理各种工作,包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI(图形用户界面)、Tk 和其他与系统有关的操作。

1.1.3 Python 的运行过程

1. 过程概述

Python 先把代码(".py"文件)编译成字节码,交给字节码虚拟机,然后虚拟机一条一条执行字节码指令,从而完成程序的执行。

2. 字节码

字节码在 Python 虚拟机程序中对应的是 PyCodeObject 对象,".pyc"文件是字节码在磁盘上的表现形式。

1.2 进入 Python 的世界

1.2.1 下载 Python

Python 语言解释器是一个轻量级的小尺寸软件,可以在 Python 语言主网站上下载(文件大小为 25～30 MB),下载网址为:https://www.Python.org/downloads/。对于初学者来说,建议采用 3.5 或之后的版本。

注意: Python 提供了不同操作系统版本,包括 Windows 操作系统版本、Linux 操作系统版本、Mac 操作系统版本及其他操作系统版本,本书以 Windows 操作系统版本为例进行介绍。

1.2.2　安装 Python

这里以 3.6.5(64 位)版本为例进行介绍,具体安装过程如下。

(1)双击所下载的 Python 安装程序,弹出图 1-1 所示的窗口,在该对话框中,选中“Add Python 3.6 to PATH”复选框。

图 1-1　安装启动窗口

(2)在图 1-1 所示的窗口中有两个安装选项,一个是根据软件默认安装设置进行安装,另一个是根据自定义设置安装。这里单击“Customize installation”自定义安装,随即进入“Optional Features”窗口,如图 1-2 所示。

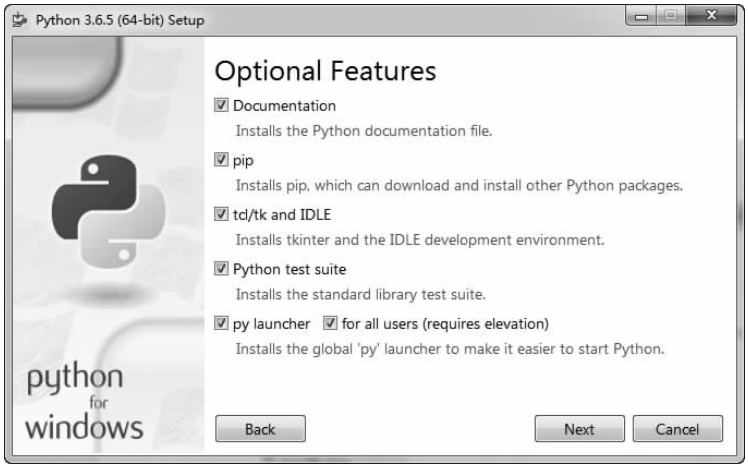

图 1-2　“Optional Features”窗口

(3)在“Optional Features”窗口中选中要安装的选项对应的复选框,这里选中所有的复选框,然后单击 Next 按钮,弹出“Advanced Options”对话框,如图 1-3 所示。

(4)在“Advanced Options”对话框中根据需要设置安装路径和相关选项,然后单击 Install 按钮程序将开始安装,如图 1-4 所示。

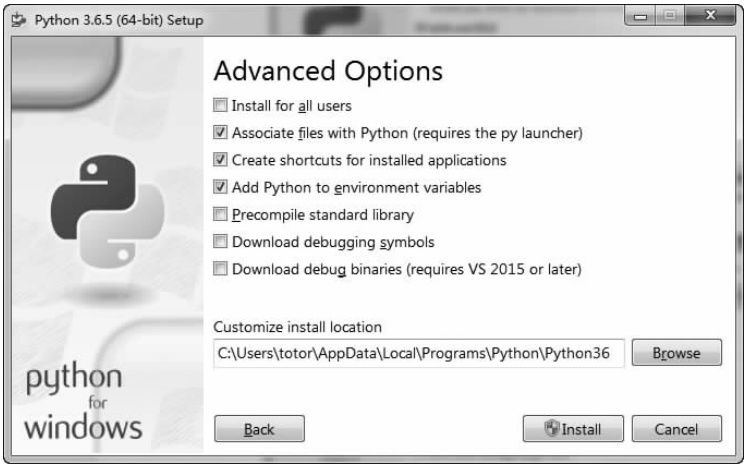

图 1-3 “Advanced Options”对话框

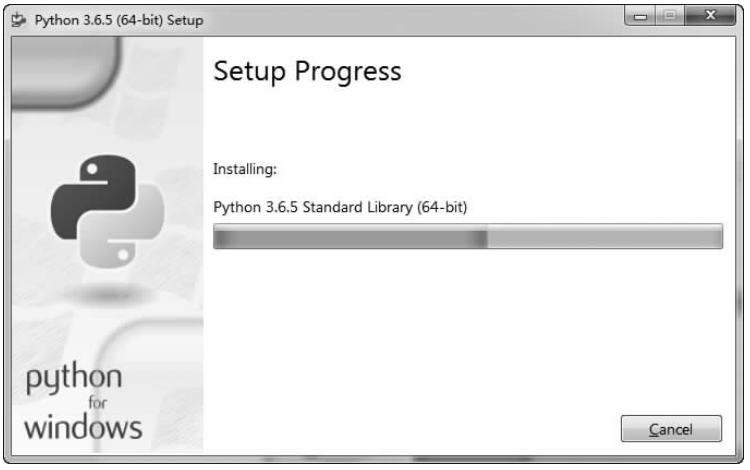

图 1-4 “Setup Progress”对话框

(5)安装完成,弹出“Setup was successful”对话框,如图 1-5 所示,单击 Close 按钮完成安装。

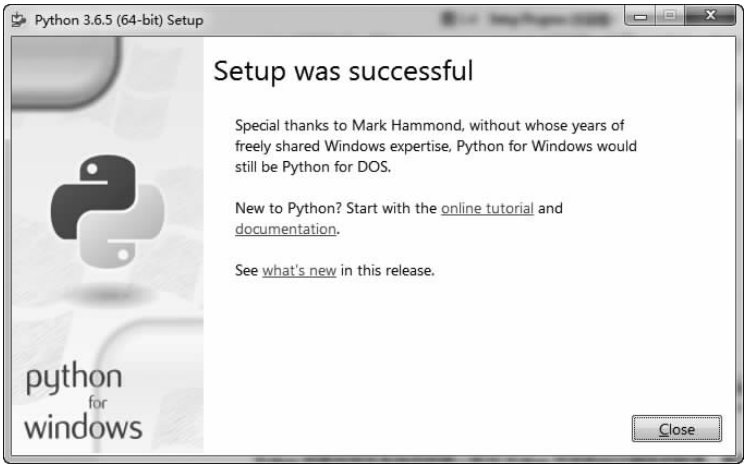

图 1-5 “Setup was successful”对话框

Python 安装包将在系统中安装一批与 Python 开发和运行相关的程序，其中最重要的两个是 Python 命令行和 Python 集成开发环境(IDLE)。

1.2.3　执行 Python 脚本文件

运行 Python 程序有两种方式：交互式和文件式。交互式指 Python 解释器即时响应用户输入的每条代码，给出输出结果。文件式也称批量式，指用户将 Python 程序写在一个或多个文件中，然后启动 Python 解释器批量执行文件中的代码。下面分别以这两种方式来编写并运行 Hello World 程序。

1. 交互式启动和运行方法

交互式有两种启动和运行方法。

第一种方法，启动 Windows 操作系统命令行工具(＜Windows 系统安装目录＞\system32\cmd.exe)，在控制台中输入 python 并按 Enter 键，然后在命令提示符＞＞＞后输入如下程序代码：

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

按 Enter 键后显示输出结果“Hello World”，如图 1-6 所示。在＞＞＞提示符后输入 exit()或者 quit()并按 Enter 键可以退出 Python 运行环境。

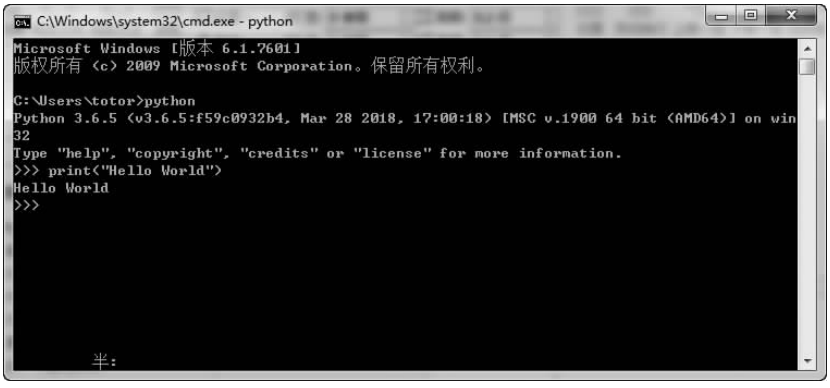

图 1-6　通过命令行启动交互式 Python 运行环境

第二种方法，通过调用安装的 IDLE 来启动 Python 运行环境。IDLE 是 Python 软件包自带的集成开发环境，可以在 Windows“开始”菜单中搜索关键词“IDLE”找到 IDLE 的快捷方式。如图 1-7 所示，展示了 IDLE 中运行 Hello World 程序的效果。

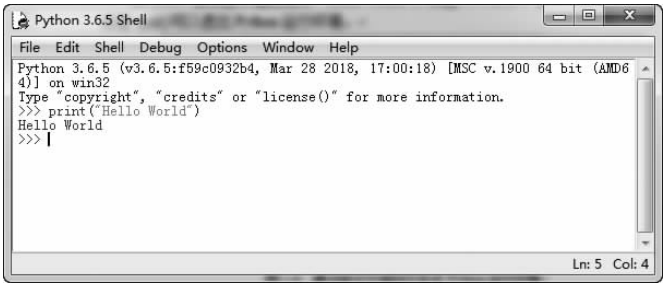

图 1-7　通过 IDLE 启动交互式 Python 运行环境

2. 文件式启动和运行方法

与交互式相对应，文件式也有两种运行方法。

第一种方法，按照 Python 的语法格式编写代码，并保存为“. py”形式的文件（以“Hello World”程序为例，将代码保存成文件 hello. py）。Python 代码可以在任意编辑器中编写，对于百行以内的代码，建议使用 Python 安装包中的 IDLE 编辑器或第三方开源记事本增强工具 Notepad＋＋。然后，打开 Windows 的命令行窗口（cmd. exe），使用 cd 命令进入 hello. py 文件所在目录，运行 Python 程序文件获得输出。

第二种方法，打开 IDLE，按 Ctrl＋N 快捷键打开一个新窗口，或在菜单中选择“File”→“New File”选项。这个新窗口不是交互模式，它是一个具备 Python 语法高亮辅助的编辑器，可以进行代码编辑。在其中输入 Python 代码。例如，输入“Hello World”程序并保存为 hello. py 文件。按 F5 键或在菜单中选择“Run”→“Run Module”选项运行该文件。

3. 启动和运行方法推荐

标识符共有四种 Python 程序运行方法，其中，最常用且最重要的是采用 IDLE 的文件式方法。

IDLE 是一个简单且有效的集成开发环境，无论交互式或文件式，都有助于快速编写和调试代码，它是小规模 Python 软件项目的主要编写工具。本书所有程序都可以通过 IDLE 编写并运行。行文方面，对于单行代码或通过观察输出结果讲解少量代码的情况，本书采用 IDLE 交互式（由＞＞＞开头）进行描述；对于讲解整段的情况，采用 IDLE 文件式。

1.3 Python 基本语法

1.3.1 Python 标识符

标识符是开发人员在程序中自定义的一些符号和名称，如变量名、函数名等。标识符规则：由字母、数值和下划线组成且不能以数字开头，Python 中的标识符是区分大小写的。命名规则：见名知意。起一个有意义的名字，尽量做到看一眼就知道是什么意思（提高代码可读性）。例如，名字就定义为 name，学生就定义为 student。

1.3.2 Python 保留字符

Python 3.6.5 中的保留字符为 33 个（Python 3.7.0 版本中的保留字符增加到了 35 个），如表 1-1 所示。自定义的标识符要避免和保留字符相同。

表 1-1 Python 中的保留字符及其说明

保留字符	说　明
False	Python 内置常量
None	Python 内置常量

续表

保留字符	说　明
True	Python 内置常量
and	用于表达式运算，逻辑与操作
as	用于类型转换
assert	断言，用于判断变量或条件表达式的值是否为真
async	Python 3.7.0 中新增保留字符，只能用于 Python 3.5 及后续版本
await	Python 3.7.0 中新增保留字符，只能用于 Python 3.5 及后续版本，且 await 语法只能出现在通过 async 修饰的函数中，否则会报 SyntaxError 错误
break	中断循环语句的执行
class	用于定义类
continue	继续执行下一次循环
def	用于定义函数或方法
del	删除变量或序列的值
elif	条件语句，与 if、else 结合使用
else	条件语句，与 if、elif 结合使用，也可用于异常和循环语句
except	except 包含捕获异常后的操作代码块，与 try、finally 结合使用
for	for 循环语句
finally	Python 3.7.0 中新增保留字符，用于异常语句，出现异常后，始终要执行 finally 包含的代码块，与 try、except 结合使用
from	用于导入模块，与 import 结合使用
global	定义全局变量
if	条件语句，与 else、elif 结合使用
import	用于导入模块，与 from 结合使用
in	判断变量是否在序列中
is	判断变量是否为某个类的实例
lambda	定义匿名函数
nonlocal	用于声明，修改嵌套作用域(enclosing 作用域，外层非全局作用域)中的变量
not	用于表达式运算，逻辑非操作
or	用于表达式运算，逻辑或操作
pass	空的类、方法或函数的占位符
raise	异常操作输出
return	用于从函数返回计算结果
try	try 包含可能会出现异常的语句，与 except、finally 结合使用

续表

保留字符	说　明
while	while 循环语句
with	简化 Python 的语句
yield	用于从函数依次返回值

Python 的保留字符可以用如下的命令查看。

```
>>>import keyword
>>>keyword.kwlist
```

结果如图 1-8 所示。

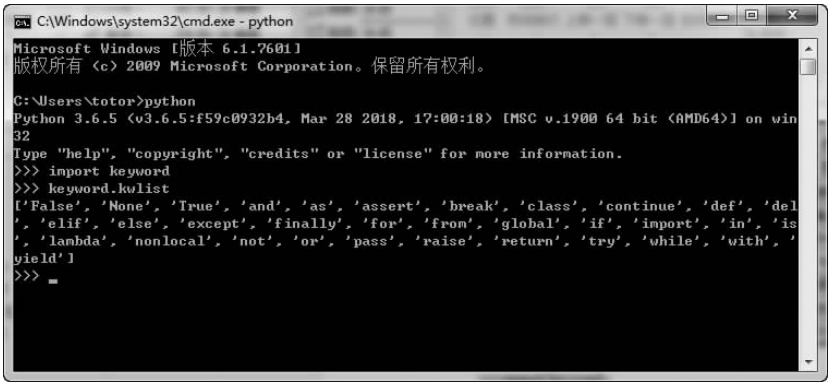

图 1-8　查看 Python 的保留字符

1.3.3　Python 行和缩进

在 Python 中，逻辑行行首的空白是有规定的，逻辑行行首的空白不对，就会导致程序执行出错。这是与其他语言的一个很重要的不同点。

缩进的空白是有要求的，下面是一些缩进的方法。

(1)一般情况下逻辑行行首不应该出现空白。

(2)if 语句的缩进方法。

(3)while 循环的缩进方法。

具体的缩进方法如下所示。

```
#一般情况下,行首不应该出现空白
import sys
#缩进的方法有两种,可以按空格,也可以按 Tab 键
#if 语句的缩进方法
a=7
if a>0:
    print "hello"  #前面的空格是按 Tab 键
```

```
#while 语句的缩进方法
a=0
while a<7:
    print a     #前面的空格是按 Tab 键
    a+=1        #前面的空格是按 Tab 键
```

1.3.4　Python 多行语句

Python 中有逻辑行与物理行之分，其区别如下所述。

```
#以下是 3 个物理行
print("abc")
print("789")
print("777")
#以下是 1 个物理行，3 个逻辑行
print("abc") ; print("789"); print("777")
#以下是 1 个逻辑行，3 个物理行
print("我其实\
是一个\
逻辑行")
```

在 Python 中，一个物理行一般可以包含多个逻辑行，在一个物理行中可以编写多个逻辑行时，逻辑行与逻辑行之间用分号隔开。每个逻辑行的后面必须有一个分号，但是在编写程序时，如果一个逻辑行占了一个物理行的最后，这个逻辑行可以省略分号。将一个逻辑行分别写在多个物理行中，行连接的方法是在行的最后加上一个“\”。

1.3.5　Python 引号

Python 中的引号有如下两个作用。

(1)表示多行注释。一对三个单引号或双引号表示多行注释。

(2)用于定义字符串。

①单引号字符串：'abc'。

②双引号字符串："abc"。

②三引号字符串：'''abc'''(三单引号)，"""abc"""(三双引号)。

总结：

(1)单引号内可以使用双引号，中间的内容会当作字符串打印。

(2)双引号内可以使用单引号，中间的内容会当作字符串打印。

(3)三单引号和三双引号中间的字符串在输出时会保持原来的格式。

(4)引号无论单双都是成对出现的，当字符串需要加入引号时，可采用单引号与双引号互相嵌套使用。

(5)Python 支持单引号，因为在某些场景下需要用到单引号，要么用“\”转义符转义(如果代码中有一大堆转义符，肯定会很难看，Python 很好地解决了这个问题)，要么外加一对

双引号，如果是双引号，则外加一对单引号。

1.3.6 Python 输入与输出

这里只介绍标准 Python 输入与输出。

1. 打印到屏幕

输出的最简单方法是使用 print 语句，可以用逗号分隔零个或多个表达式，如下面的代码所示。

```
print("Python is really a great language,","isn't it?")
```

这将产生以下结果。

```
Python is really a great language,isn't it?
```

2. 读取键盘输入

Python 2 中有两个内置的函数可从标准输入（默认来自键盘）读取数据，这两个函数分别是 input()和 raw_input()。但在 Python 3 中，raw_input()函数已被弃用。此外，input()函数从键盘读取的数据是作为字符串来处理的，不论是否使用引号('' 或"")。

示例：

```
x=input("请输入 x=")
请输入 x=111

y=input("请输入 y=")
请输入 y=222

z=x+y
print("x+y="+z)
```

运行结果如下。

```
x+y=111222
```

可以看到 input 的返回值永远是字符串，当需要返回 int 型时需要使用 int(input())的形式。例如：

```
x=int(input("请输入 x="))
请输入 x=111

y=int(input("请输入 y="))
请输入 y=222

z=x+y
print("x+y=",z)
```

运行结果如下。

```
x+y=333
```

3. 格式化输出

一般来说，我们希望更多地控制输出格式，而不是简单地以空格分割。这里有两种方式。

第一种是由用户自己控制。使用字符串切片、连接操作及字符串包含的一些有用的操作。

示例：

```
>>>for x in range(1,11):
        print(str(x).rjust(2),str(x*x).rjust(3),end='')
            print(str(x*x*x).rjust(4))
```

输入完成后，按两次 Enter 键，输出结果如下。

```
1   1    1
2   4    8
3   9   27
4  16   64
5  25  125
6  36  216
7  49  343
8  64  512
9  81  729
10 100 1000
```

在第一种方式中，字符串对象的 str.rjust()方法的作用是将字符串靠右，并默认在左边填充空格，所占长度由参数指定，类似的方法还有 str.ljust()和 str.center()。这些方法并不会写任何东西，它们仅仅返回新的字符串，如果输入很长，它们并不会截断字符串。

第二种是使用 str.format()方法。

用法：通过{}和:来代替传统%方式。

(1)使用位置参数。

要点：从以下例子可以看出位置参数不受顺序约束，且可以为{}，只要 format 里有相对应的参数值即可，参数索引从 0 开始，传入位置参数列表可用 * 列表的形式。

```
>>> li = ['hoho',18]
>>> 'my name is {},age {}'.format('hoho',18)
   'my name is hoho,age 18'
>>> 'my name is {1},age {0}'.format(10,'hoho')
   'my name is hoho,age 10'
>>> 'my name is {1},age {0} {1}'.format(10,'hoho')
```

```
'my name is hoho,age 10 hoho'
>>> 'my name is {},age {}'.format(*li)
   'my name is hoho,age 18'
```

(2)使用关键字参数。

要点:关键字参数值要对得上,可用字典当关键字参数传入值,字典前加**即可。

```
>>> hash={'name':'hoho','age':18}
>>> 'my name is {name},age is {age}'.format(name='hoho',age=19)
   'my name is hoho,age is 19'
>>> 'my name is {name},age is {age}'.format(**hash)
   'my name is hoho,age is 18'
```

(3)填充与格式化。

格式:{0:[填充字符][对齐方式 <^>][宽度]}.format()。

```
>>> '{0:*>10}'.format(20)  ##右对齐
   '********20'
>>> '{0:*<10}'.format(20)  ##左对齐
   '20********'
>>> '{0:*^10}'.format(20)  ##居中对齐
   '****20****'
```

(4)精度与进制。

```
>>> '{0:.2f}'.format(1/3)
   '0.33'
>>> '{0:b}'.format(10)    #二进制
   '1010'
>>> '{0:o}'.format(10)    #八进制
   '12'
>>> '{0:x}'.format(10)    #十六进制
   'a'
>>> '{:,}'.format(12369132698)  #千分位格式化
   '12,369,132,698'
```

(5)使用索引。

```
>>> li=['hoho',18]
>>> 'name is {0[0]} age is {0[1]}'.format(li)
   'name is hoho age is 18'
```

1.4　实训:配置 Python 开发环境

1.4.1　下载 Python 集成开发环境 PyCharm

PyCharm 提供免费的社区版与付费的专业版。专业版额外增加了一些功能,如项目模板、远程开发、数据库支持等。个人学习 Python 使用免费的社区版已足够。

PyCharm 社区版下载地址:http://www.jetbrains.com/pycharm/download/。

1.4.2　安装 PyCharm Community Edition

下载 Python 集成开发环境 PyCharm 后,可按照如下步骤进行安装。

(1)按照 1.2 中所示的内容,下载并安装好 Python。

(2)双击 PyCharm 的安装程序 pycharm-community-2016.2.3.exe,进入安装界面,如图 1-9 所示。

图 1-9　PyCharm 的安装界面

(3)单击 Next 按钮,在弹出的“Choose Install Location”对话框中选择安装 PyCharm 的路径,如图 1-10 所示。

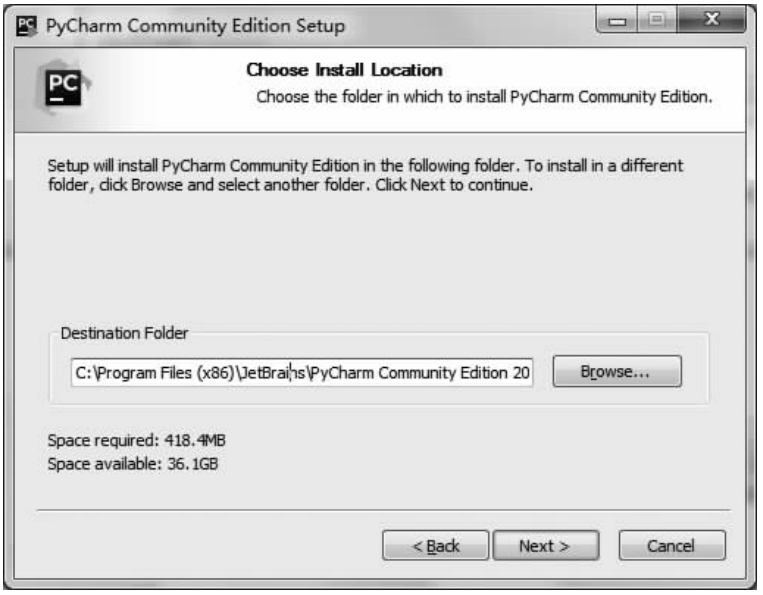

图 1-10　选择安装路径

(4)单击 Next 按钮,进入图 1-11 所示的“Installation Options”对话框,根据需要选中相应复选框,在“Create associations”选项组中可选中“.py”复选框,则以后打开“.py”文件就会用 PyCharm 打开。

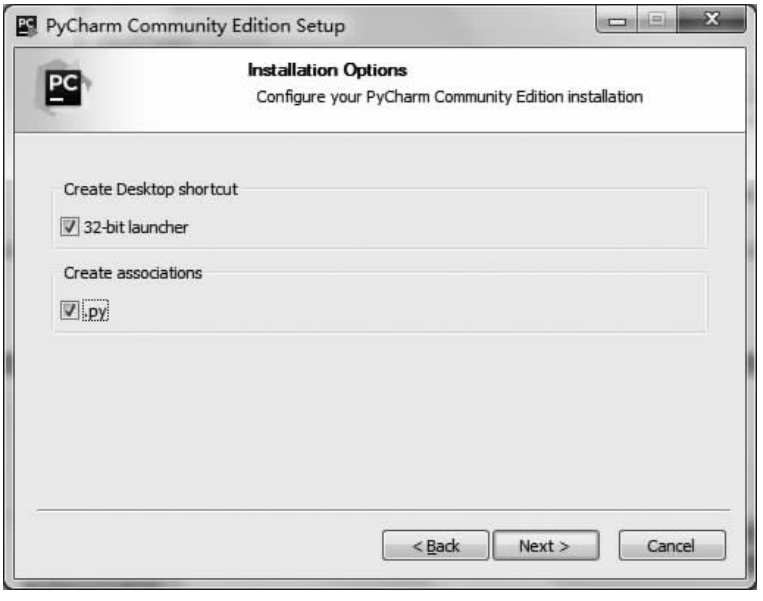

图 1-11　“Installation Options”对话框

(5)设置完成,单击 Next 按钮,根据安装向导保持默认设置,直到出现图 1-12 所示的对话框,单击 Finish 按钮完成安装。

图 1-12　安装完成

1.4.3　PyCharm 的使用步骤

(1)安装软件后，运行 PyCharm，选择“File”→“New Project”选项开始创建第一个项目，如图 1-13 所示。

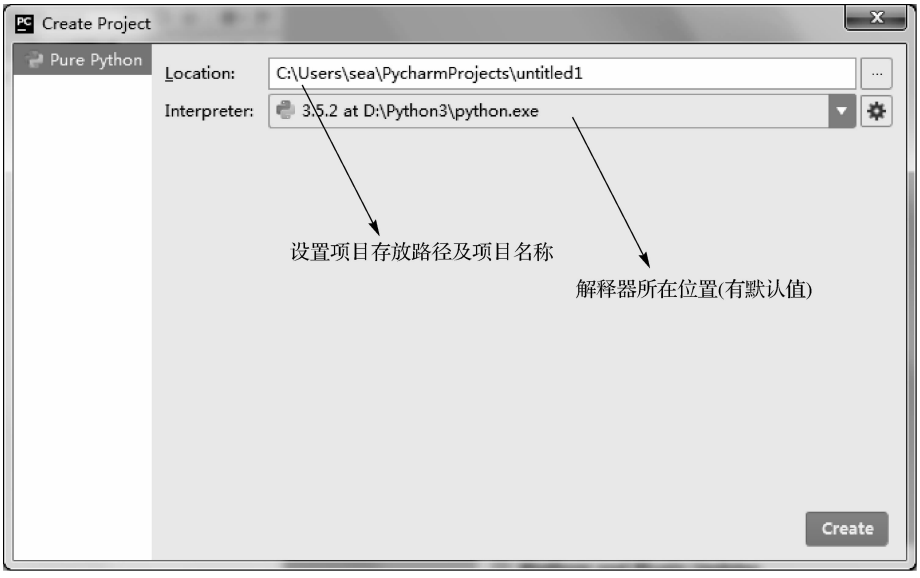

图 1-13　“Create Project”对话框

(2)在左侧导航栏中选择“Pure Python”选项,单击 Location 文本框右侧的按钮,在弹出的对话框中选择项目的路径,在 Interpreter 下拉列表框中选择 Python 版本,这里直接使用 Python 的安装路径即可。

(3)选择完成后,单击 Create 按钮,进入图 1-14 所示的窗口。这时就可以创建文件了,这里以刚刚创建的 untitled1 文件夹为例,右击 untitled1,在弹出的快捷菜单中选择“New”→“Python File”选项,如图 1-14 所示。

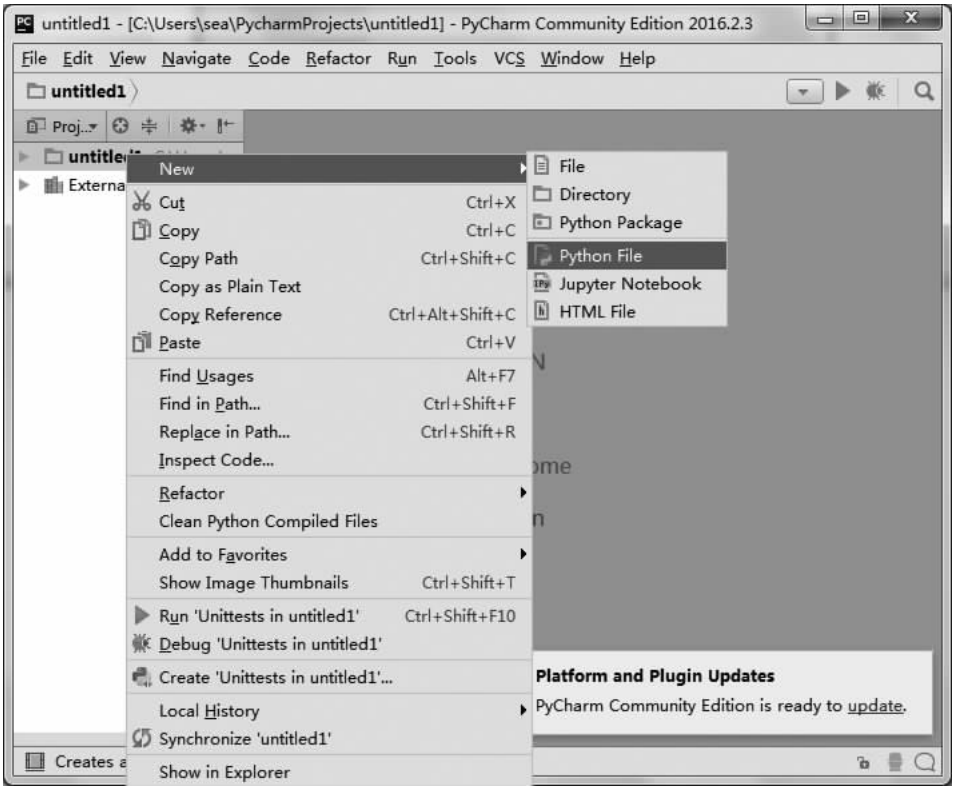

图 1-14　新建文件

(4)随即弹出图 1-15 所示的对话框。

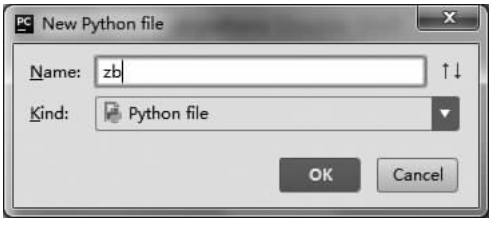

图 1-15　“New Python file”对话框

(5)在 Name 文本框中输入文件名,单击 OK 按钮之后写下“print('hello,world')”,然后在界面中右击,并在弹出的快捷菜单中选择“Run 'zb'”选项,如图 1-16 所示。

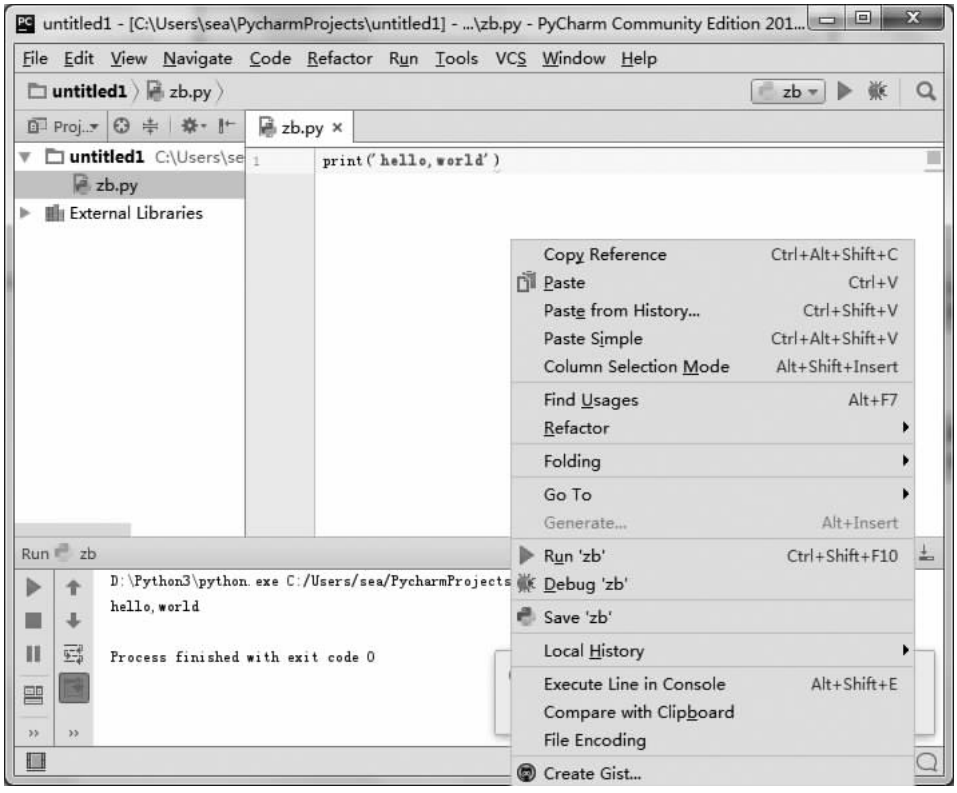

图 1-16　选择 Run 'zb' 选项

(6)随即出现图 1-17 所示的结果。

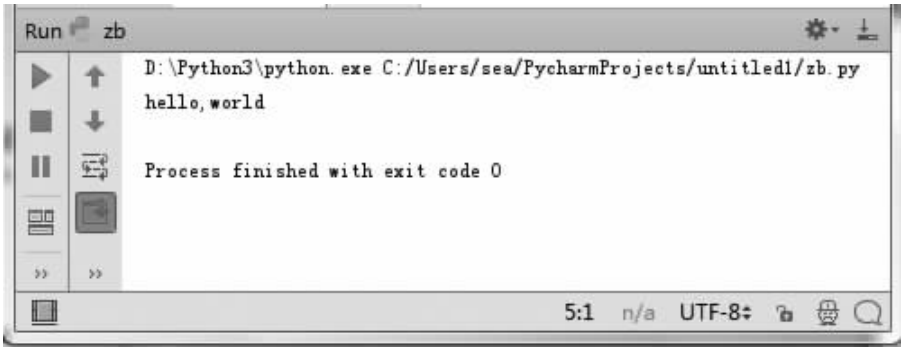

图 1-17　执行结果

(7)对于同一个脚本,第一次运行,可右击脚本名称并在弹出的快捷菜单中选择“Run×××”选项,以后运行脚本可以直接单击右上角或者左下角的三角按钮运行,如图 1-18 所示。

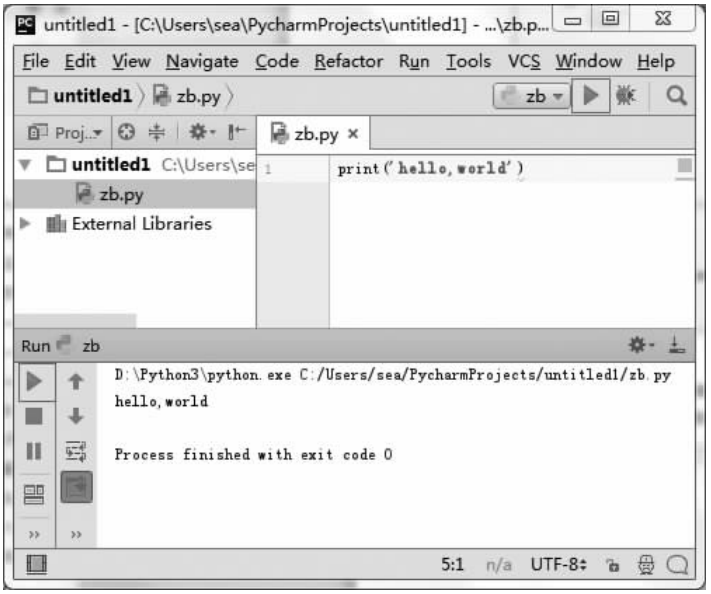

图 1-18　运行脚本

注意：更改文件后，单击三角按钮运行项目和启用快捷键运行项目时，其结果不会随着内容改变而自动更改，所以常会运行错误的文件而未发现。推荐第一次运行使用右键的方式，将脚本切换之后再使用单击三角按钮的方式运行。