

普通高等教育公共基础课系列教材·信息技术类

Python 语言程序设计

胡 滨 石礼娟 万世明 © 主编

1.5
-6



科学出版社

普通高等教育公共基础课系列教材·信息技术类

Python 语言程序设计

胡滨 石礼娟 万世明 主编

汤海林 彭明霞 副主编



科学出版社

北京

内 容 简 介

本书主要介绍 Python 的运行环境、基本语法、程序基本结构、组合数据类型、函数、文件、文件异常、常用标准库和第三方库的相关知识等内容。书中列举了丰富的教学案例,以帮助学生更好地掌握相关知识。本书知识完整、实用性强,讲解基础知识的同时,还介绍使用 Python 进行数据处理的方法。

本书既可以作为程序设计初学者和高等院校学生学习 Python 程序设计的基础教材,也可以供参加全国计算机等级考试的人员参考。

图书在版编目(CIP)数据

Python 语言程序设计 / 胡滨, 石礼娟, 万世明主编. —北京: 科学出版社, 2020.3

(普通高等教育公共基础课系列教材·信息技术类)

ISBN 978-7-03-064429-9

I. ①P… II. ①胡… ②石… ③万… III. ①软件工具—程序设计—高等学校—教材 IV. ①TP311.561

中国版本图书馆 CIP 数据核字(2020)第 025865 号

责任编辑: 戴 薇 王国策 袁星星 / 责任校对: 马英菊
责任印制: 吕春珉 / 封面设计: 东方人华平面设计部

科 学 出 版 社 出 版

北京东黄城根北街 16 号

邮政编码: 100717

<http://www.sciencep.com>

三河市骏杰印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2020 年 3 月第 一 版 开本: 787×1092 1/16

2020 年 3 月第一次印刷 印张: 15 1/4

字数: 356 000

定价: 46.00 元

(如有印装质量问题, 我社负责调换〈骏杰〉)

销售部电话 010-62136230 编辑部电话 010-62135763-2047

版权所有, 侵权必究

举报电话: 010-64030229; 010-64034315; 13501151303



前言

Python 是一种面向对象的解释型计算机程序设计语言，由荷兰人吉多·范罗苏姆（Guido van Rossum）于 1989 年发明，1991 年公开发布第一个版本。Python 的设计哲学是优雅、明确、简单，它强调代码的可读性和语法的简洁性。Python 通过缩进划分不同结构，所以 Python 代码总是清晰明了的，这也对编程者在编码的规范性上提出了要求。Python 具有非常优秀的可扩展性，提供了海量的标准库和第三方库，主要用于小规模程序设计，如处理计算量大的矩阵、进行数据分析、画饼图等。

程序设计是高等院校普遍开设的计算机基础课程，它面向计算机专业和非计算机专业的学生，旨在使学生掌握程序设计的基本思想和方法。随着大数据、人工智能时代的到来，Python 以其简单易学的特点和丰富的数据处理功能得到了广泛的应用。因此，Python 语言程序设计适合作为程序设计的入门课程。

本书由工作在教学第一线的高校教师编写完成。在编写本书时，编者注重保持知识的完整性和系统性，精选教学案例，由浅入深，脉络清晰，既有对具体问题的思路解析，又有对代码的具体讲解。书中教学案例提供相应的源代码，方便教学。

全书共分 9 章，主要内容如下：

第 1 章主要介绍 Python 的发展过程、特点、编程环境的配置和使用方法。

第 2 章主要介绍 Python 的编码规则、变量的声明及使用、基本数据类型、运算符的使用、输入函数 `input()` 和输出函数 `print()` 的使用。

第 3 章主要介绍 Python 程序的 3 种基本控制结构、常用算法及其应用。

第 4 章主要介绍字符串、元组、列表、字典、集合的相关知识和应用。

第 5 章主要介绍函数的定义和调用方法、参数传递的多种方式、嵌套函数的使用方法、`lambda` 函数的使用方法、变量的作用域。

第 6 章主要介绍 Python 中文本文件的读写方法、二进制文件的读写方法、`os` 模块中实现文件级和目录级操作的方法、`jieba` 库的应用和第三方库的安装、Python 中的异常处理机制。

第 7 章主要介绍基于 `turtle` 库的绘图方法及应用、`random` 库的常用函数及应用、`time` 库的常用函数及应用。

第 8 章主要介绍 `numpy` 和 `matplotlib` 库的应用方法、Python 中科学计算的综合应用方法。

第 9 章主要介绍网络爬虫的概念及工作原理、`urllib` 库 `request` 请求模块的使用方法、`urllib` 库 `parse URL` 解析模块的使用方法、常见的网络异常、`urllib` 库 `error` 异常处理模块的使用方法、`requests` 库的使用方法、服务器返回的数据格式、网页的结构、正则表达式的使用、`re` 模块解析网页数据、`lxml` 库解析网页数据。

目 录

第 1 章 概述	1
1.1 Python 的特点	1
1.2 Python 3.x 编程环境的配置与编程实例	3
1.2.1 Python 2.x 和 Python 3.x 的区别	5
1.2.2 编写简单 Python 程序	6
1.3 计算机程序设计概述	8
1.3.1 计算机程序设计语言概述	8
1.3.2 结构化程序设计	9
1.3.3 面向对象程序设计	10
本章小结	11
第 2 章 Python 语言基础	12
2.1 Python 的语法特点	12
2.1.1 注释	12
2.1.2 代码缩进	14
2.1.3 编码规范	15
2.2 保留字与标识符	16
2.2.1 保留字	17
2.2.2 标识符	18
2.3 变量	18
2.3.1 理解 Python 中的变量	18
2.3.2 变量的定义与使用	18
2.4 基本数据类型及其转换	19
2.4.1 数字类型	19
2.4.2 字符串类型	20
2.4.3 布尔类型	21
2.4.4 数据类型转换	21
2.5 运算符	22
2.5.1 算术运算符	22
2.5.2 赋值运算符	23
2.5.3 关系运算符	23
2.5.4 逻辑运算符	23
2.5.5 位运算符	24



2.6	基本输入和输出	26
2.6.1	使用 input() 函数输入	26
2.6.2	使用 print() 函数输出	26
	本章小结	28
第 3 章	Python 程序的控制结构	29
3.1	程序的顺序结构	29
3.2	程序的分支结构	30
3.2.1	单分支结构	30
3.2.2	双分支结构	32
3.2.3	多分支结构	33
3.2.4	分支结构的嵌套	35
3.3	程序的循环结构	38
3.3.1	while 循环	38
3.3.2	for 循环	39
3.3.3	循环控制: break 和 continue	40
3.3.4	循环的嵌套	41
3.4	常用算法及其应用	44
3.4.1	穷举法	44
3.4.2	解析法	46
	本章小结	48
第 4 章	组合数据类型	49
4.1	组合数据类型的基本概念	49
4.2	字符串(字符序列)	49
4.2.1	字符串的定义	49
4.2.2	字符串的操作函数	51
4.2.3	字符串的操作方法	53
4.2.4	字符串的应用	58
4.3	元组	60
4.3.1	元组的定义与特点	60
4.3.2	元组的索引	61
4.3.3	元组的切片	61
4.3.4	元组的操作函数	61
4.3.5	元组的操作方法	63
4.3.6	元组的应用	64
4.4	列表	65
4.4.1	列表的定义与特点	65



4.4.2	列表的索引	66
4.4.3	列表的切片	67
4.4.4	列表的操作函数	68
4.4.5	列表的操作方法	69
4.4.6	列表推导式	71
4.4.7	列表的应用	72
4.5	字典	79
4.5.1	字典的定义	80
4.5.2	字典的操作函数	80
4.5.3	字典的操作方法	84
4.5.4	字典的应用	86
4.6	集合	88
4.6.1	集合的定义与特点	89
4.6.2	集合的运算	90
4.6.3	集合的操作函数	92
4.6.4	集合的操作方法	93
4.6.5	集合的应用	94
	本章小结	95
第 5 章	函数	97
5.1	函数的使用	98
5.1.1	函数的定义	98
5.1.2	函数的调用与返回值	100
5.1.3	形参和实参	101
5.2	函数的参数	102
5.2.1	引用传递	102
5.2.2	必备参数	103
5.2.3	命名参数	104
5.2.4	默认参数	104
5.2.5	不定长参数	105
5.2.6	经典案例	106
5.3	嵌套函数和 lambda 函数	108
5.3.1	嵌套函数	108
5.3.2	lambda 函数	108
5.4	变量的作用域	110
5.4.1	global 声明全局变量	111
5.4.2	嵌套函数中的 nonlocal 声明	111
	本章小结	113



第 6 章 文件与异常	114
6.1 文件的定义、引用与分类	114
6.1.1 文件的定义	114
6.1.2 文件的引用	114
6.1.3 文件的分类	115
6.2 文件的打开与关闭	115
6.3 文本文件的读与写	117
6.3.1 文本文件的读操作	117
6.3.2 文本文件的写操作	120
6.3.3 文件指针的移动	121
6.3.4 文本文件的综合应用	122
6.4 二进制文件的读与写	132
6.4.1 使用 pickle 模块读写二进制文件	132
6.4.2 使用 struct 模块	133
6.5 os 模块	135
6.6 jieba 库及第三方库安装	138
6.6.1 jieba 库概述	138
6.6.2 jieba 库的解析与应用	140
6.7 异常处理	144
6.7.1 异常概述	144
6.7.2 使用 try...except 处理异常	146
6.7.3 使用 try...except...except 处理异常	149
6.7.4 使用 try...except...else 处理异常	149
6.7.5 使用 try...except...finally 处理异常	151
本章小结	152
第 7 章 Python 常用标准库	153
7.1 turtle 库	153
7.1.1 turtle 库与基本绘图	153
7.1.2 绘图窗口设置	153
7.1.3 画笔状态控制函数	154
7.1.4 画笔运动控制函数	155
7.1.5 turtle 库应用	156
7.2 random 库	163
7.2.1 random 库常用函数	163
7.2.2 random 库应用	163
7.3 time 库	165
7.3.1 时间表达方式	165

- 7.3.2 time 库常用函数 166
- 本章小结 170
- 第 8 章 科学计算** 171
 - 8.1 numpy 库 171
 - 8.1.1 numpy 库概述 171
 - 8.1.2 numpy 库应用 172
 - 8.2 matplotlib 库 184
 - 8.2.1 matplotlib 库概述 184
 - 8.2.2 matplotlib 库应用 185
 - 8.3 综合实例 193
 - 8.3.1 图像的显示 193
 - 8.3.2 天气预报图 195
 - 本章小结 200
- 第 9 章 网络爬虫开发** 201
 - 9.1 网络爬虫概述 201
 - 9.1.1 获取网页 201
 - 9.1.2 提取数据 201
 - 9.1.3 保存数据 202
 - 9.1.4 自动化程序 202
 - 9.2 Python 的网络请求 202
 - 9.2.1 urllib 库的使用 202
 - 9.2.2 requests 库的使用 211
 - 9.3 网页解析 214
 - 9.3.1 网页数据格式和网页结构 214
 - 9.3.2 正则表达式 215
 - 9.3.3 BeautifulSoup 217
 - 9.4 网络爬虫开发实战 224
 - 本章小结 230
- 参考文献** 231

第1章 概述

学习要点

1. 了解 Python 语言的发展过程。
2. 了解 Python 语言的特点。
3. 掌握 Python 语言编程环境的配置和使用方法。
4. 了解 Python 程序设计的基本流程。
5. 了解程序设计语言的发展过程和三大基本结构。

Python 是一种计算机程序设计语言。Python 设计之初主要用于编写自动化脚本，随着版本的不断更新和语言功能的添加，其越来越多地用于独立的、大型项目的开发。Python 的创始人是荷兰人吉多·范罗苏姆。1989 年圣诞节期间，吉多·范罗苏姆决心开发一个新的脚本解释程序，即 Python。Python 的语法非常简单，初学者可以轻松上手。因此，用 Python 做科学计算的研究机构日益增多，一些知名大学已经采用 Python 来教授程序设计课程。例如，卡耐基-梅隆大学的编程基础、麻省理工学院的计算机科学及编程导论就使用 Python 语言讲授。近年来，随着 Python 功能的完善，其应用越来越广泛，编程时需要的绝大多数功能可以找到相应的类库，从而节省了使用者大量的时间和精力。对于初级程序员而言，Python 是一种伟大的语言，它支持广泛的应用程序开发，如简单的文字处理、WWW 浏览器开发、游戏开发。另外，Python 也可以应用在人工智能、大数据和机器学习等前沿科技领域。此外，Python 的社区活跃度非常高。

值得注意的是，Python 目前有两个主流的版本：一个是 Python 2.x，另一个是 Python 3.x。这两个版本相差较大，并且不完全兼容。本书是基于 Python 3.7.3 版本进行编写的。

1.1 Python 的特点

1. Python 的优点

(1) 简单易学。Python 语言力求代码简洁、优美。其采用强制缩进的方式来标识代码块，通过减少无用的结构符号使代码具有极佳的可读性。阅读一段结构良好的 Python 程序时，读者能够专注于要解决的问题，而不用太纠结编程语言本身的语法。相比其他语言经常使用英文关键字和标点符号的情况，Python 有相对较少的关键字，明确定义的语法，且结构简单，容易理解和学习。

(2) 解释型、交互式 and 面向对象的脚本语言。Python 是一种解释型语言，故在开发过程中没有编译环节。Python 是一种交互式语言，故可以在一个提示符“>>>”后直接执行代码。Python 是面向对象的语言，故其支持面向对象的风格及将代码封装在对象中的编程



技术。

(3) 可移植性、可扩展性、可嵌套性。可移植性基于 Python 开放源代码的特性。Python 程序可以在绝大多数系统平台上运行,包括 Linux、Windows、FreeBSD、Macintosh、Solaris 及 Android 等。可扩展性是指 Python 程序中可以使用 C 或 C++编写的程序。例如,如果需要一段运行很快的关键代码,或是需要编写一些不便公开的算法,可以先使用 C/C++完成关键部分的代码,然后从 Python 程序中调用这部分代码即可。可嵌套性是指能够把 Python 程序嵌入 C/C++程序中,从而向程序用户提供脚本功能。

(4) 丰富的库资源。Python 拥有一个功能丰富的标准库。Python 语言的核心只包含数字、字符串、列表、字典、文件等常见类型和函数,其标准库提供了系统管理、网络通信、文本处理、数据库接口、图形系统、XML 处理等功能。另外,Python 还提供了大量的第三方库,其使用方式与标准库类似,功能覆盖科学计算、Web 开发、数据库接口、图形系统等多个领域。

下面对解释型语言做进一步说明。与解释型语言对应的为编译型语言。因为计算机只能直接理解机器语言,不能直接理解高级语言,所以必须将高级语言翻译成机器语言,这样计算机才能执行高级语言编写的程序。翻译的方式有两种,一种是编译,另一种是解释。两种方式的本质相同,只是翻译的时间点不同。

解释型语言编写的程序在运行时翻译。这样,解释型语言每执行一次就要翻译一次,执行效率较低。编译型语言编写的程序在编译时直接编译成机器可以执行的文件(通常为.exe 文件),编译和执行是分开的,但是不能跨平台,如 C++、C 语言。这样,以后运行该程序时直接使用编译的结果(.exe 文件)即可,程序执行效率高。

2. Python 的缺点

(1) 速度慢。Python 的运行速度相比 C、C++、Java 确实慢很多,但这里所指的运行速度慢在大多数情况下用户是无法直接感知到的。在大多数情况下,Python 已经完全可以满足用户对程序速度的要求。若有速度要求,可以用 C/C++来改写关键部分。

(2) 代码不能加密。Python 的开源性使其不能加密,即它的源码都是以明文形式存放的。如果项目要求源代码必须是加密的,则不宜用 Python 实现。

(3) 对多处理器支持受限。这是 Python 较突出的缺点。全局解释器锁(global interpreter lock, GIL)是计算机程序设计语言解释器用于同步线程的工具,当 Python 的默认解释器要执行字节码时,都需要先申请这个锁。如果试图通过多线程扩展应用程序,将被 GIL 限制,使任何时刻仅有一个线程在执行。即使在多核 CPU 平台上,由于 GIL 的存在,Python 也禁止多线程的并行执行。

(4) 构架选择太多。

(5) Python 2.x 系列与 Python 3.x 系列不兼容。在使用 Python 的过程中,不同系列互不兼容给所有的 Python 工程师造成了很多不便。



1.2 Python 3.x 编程环境的配置与编程实例

因为 Windows 系统未内置任何 Python 版本，必须手动安装 Python 程序。Python 可以在官网下载，网址为 <https://www.python.org/downloads/>。Python 官网下载界面如图 1-1 所示。下载时要注意选择所用的操作系统和版本，如本书选择 Python 3.7.3 版本的 Windows x86-64 executable installer 类型。

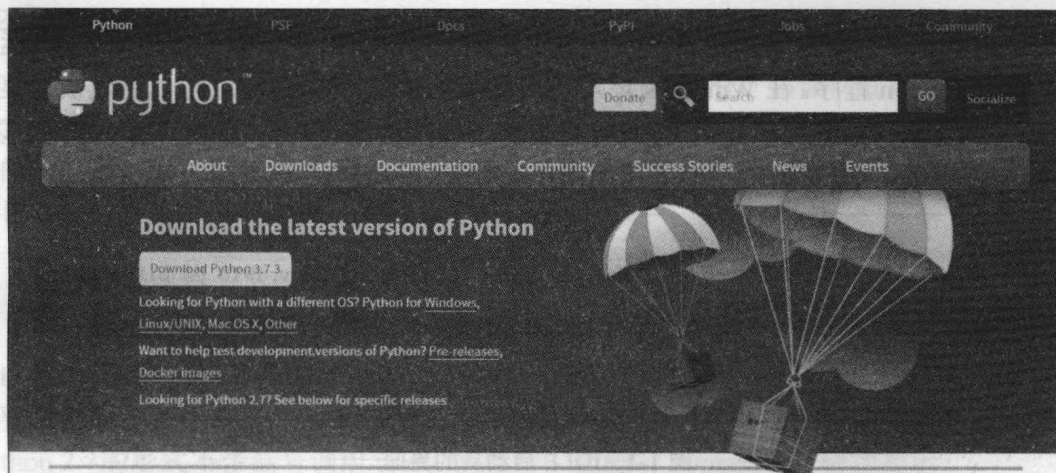


图 1-1 Python 官网下载界面

下载完成后直接双击安装包进行安装，安装过程中应选中“Add Python 3.7 to PATH”复选框，如图 1-2 所示。选中该复选框的目的是将 Python 加入环境变量中，以便在 cmd 命令行窗口任意目录下识别 Python 命令。

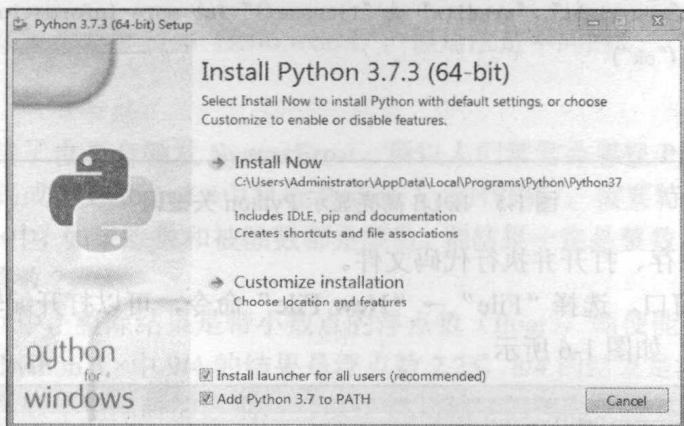


图 1-2 Python 安装界面

在图 1-2 所示安装界面选择“Install Now”选项开始安装。安装完成后，打开 cmd 命令行窗口，输入“python”命令，如图 1-3 所示。如果可以查看到相关版本信息，表示 Python 已经可以使用了。

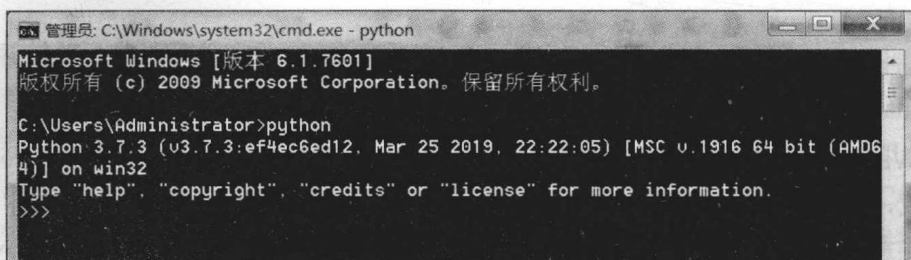


图 1-3 cmd 命令行窗口输入界面

Python 提供了一个简洁的集成开发环境 IDLE。利用 IDLE 可以较为方便地创建、运行、测试和调试 Python 程序。在 Windows 环境下启动 IDLE 有多种方式，如可以通过快捷菜单、桌面图标运行 IDLE，也可以进入 Python 安装目录直接运行 IDLE。启动 IDLE 后就可以编写 Python 程序了。IDLE 启动后的界面如图 1-4 所示。

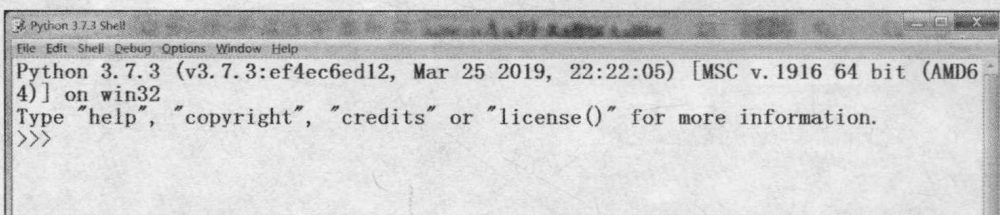


图 1-4 IDLE 启动后的界面

IDLE 本身就是一个 Python Shell，可以在 IDLE 窗口直接输入和执行 Python 语句。IDLE 自动对输入的语句进行排版和关键词高亮显示，如图 1-5 所示。

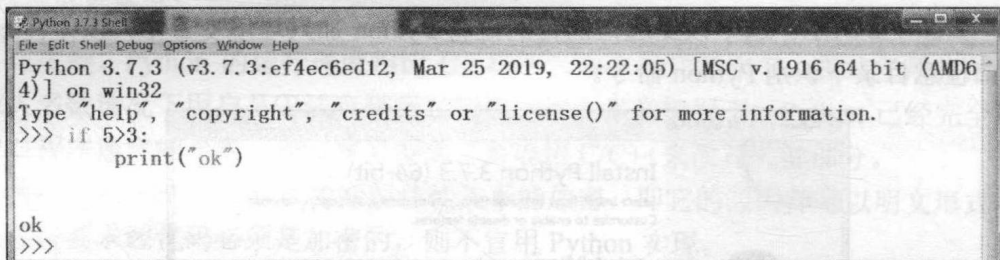


图 1-5 IDLE 高亮显示 Python 关键词

IDLE 还可以保存、打开并执行代码文件。

(1) 在 IDLE 窗口，选择“File”→“New File”命令，可以打开编辑窗口，在其中可以输入代码并保存，如图 1-6 所示。

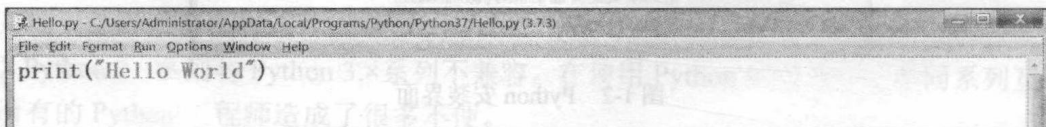
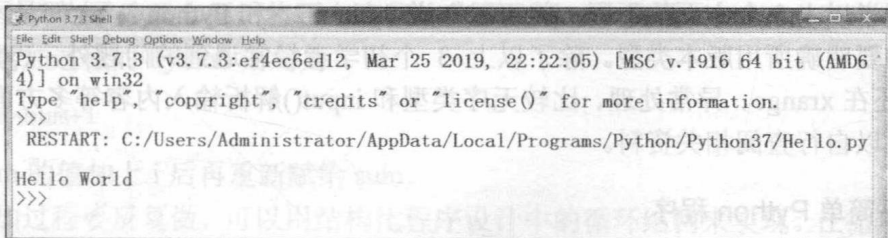


图 1-6 Hello.py 文件编辑窗口

(2) 选择“Run”→“Run Module”命令或按 F5 键，即可执行代码文件。执行后的输出结果如图 1-7 所示。



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/Administrator/AppData/Local/Programs/Python/Python37/Hello.py
Hello World
>>>
```

图 1-7 执行 Hello.py 文件的输出结果

1.2.1 Python 2.x和 Python 3.x的区别

初学者学习 Python 或想要用 Python 开发一个新项目时，该如何选择 Python 版本呢？其实大部分 Python 库同时支持 Python 2.x和 Python 3.x版本，所以无论选择哪个版本都是可以的。但是，为了在使用 Python 时避开某些版本中常见的一些陷阱，或移植某个 Python 项目，依然有必要了解 Python 两个常见版本之间的主要区别。

1. print()函数

输出函数在 Python 3.x中有一个很小的改动，由于该函数大量使用，下面做简要介绍。Python 2.x中的 print 语句被 Python 3.x中的 print()函数取代，在 Python 3.x中必须用括号将需要输出的参数括起来。在 Python 2.x中使用额外的括号也是可以的。但在 Python 3.x中以 Python 2.x的形式不带括号调用 print()函数时，会触发 SyntaxError 错误提示。

Python 2.x中的 print 语句如下。

```
1 | print "Hello,world!"
```

Python 3.x中的 print()函数如下。

```
1 | print("Hello,world!")
```

两者输出的结果都是字符串"Hello,world!"，但语法是不同的。

2. 整除

因为即使写错了也不会触发 SyntaxError，所以人们常常会忽视 Python 3.x在整除上的改变。在移植代码或在 Python 2.x中执行 Python 3.x的代码时，需要特别注意这个改动。

在 Python 2.x中，如果除数和被除数都是整数，则结果一定是整数。例如，在 Python 2.x中 9/4 的结果是整数 2。

在 Python 3.x中，整除结果是带小数点的浮点数 (float)，即使能整除，其结果也是浮点数。例如，在 Python 3.x中 9/4 的结果是浮点数 2.25，8/4 的结果是浮点数 2.0。

为了避免不同版本整除时的区别，可以在 Python 3 的代码中用 float(9)/4 或 9.0/4 代替 9/4，使代码在 Python 2.x和 Python 3.x运行下能得到一致的结果。

3. Unicode

Python 2.x有基于 ASCII 码的 str()类型，其可通过单独的 unicode()函数转换成 unicode 类型，但没有 byte 类型。在 Python 3.x中，有 Unicode (UTF-8) 字符串，以及两个字节类：bytes 和 bytearray。



通过列举以上 3 个方面的不同，就可以知道 Python 2.x 和 Python 3.x 之间是不兼容的，使用过程中要明确所用版本类型。除了以上 3 个初学者经常遇到的问题外，Python 2.x 和 Python 3.x 还在 xrange、异常处理、比较无序类型和 input() 解析输入内容等多方面有很多区别，读者可以自行查阅相关资料。

1.2.2 编写简单 Python 程序

前面已经介绍了 Python 的开发环境和一些基本概念，下面通过两个应用程序实例来讲解编写应用程序的全过程。编写一个应用程序的基本过程如下：

- (1) 分析问题，明确目标与算法设计。
- (2) 输入所需数据。
- (3) 对数据进行处理（算法设计的核心所在）。
- (4) 输出数据和相应结果。
- (5) 保存程序文件。
- (6) 运行和调试程序。

例 1.1 输入圆半径，求出圆的周长和面积。

任务分析：

本例是根据输入函数 input() 给出圆半径 r，再根据 r 的值结合圆周长 c 和圆面积 s 的计算公式分别计算出相应结果。最后使用 print() 函数输出结果。

源程序：

```

1 pi=3.14
2 r=float(input("请输入半径（默认单位厘米）："))
3 c=2*pi*r      #求圆周长 c
4 s=pi*r**2    #求圆面积 s
5 print("半径%f 的圆，其周长是%f，其面积是%f"%(r,c,s))

```

程序结果：

代码编写完后，一个完整的程序就设计好了，然后选择“Run”→“Run Module”命令或按 F5 键运行程序。如果有语法错误，系统显示错误信息，提示用户进行修改；如果没有语法错误，则正常执行程序。

对于初次接触计算机语言的学习者，程序运行时出现错误很正常，关键是要耐心去发现错误、纠正错误。最开始的编程要求是没有语法错误，随着学习的深入，需要掌握逻辑错误的排除方法。该程序的运行结果如下：

请输入半径（默认单位厘米）：3

半径 3.000000 的圆，其周长是 18.840000，其面积是 28.260000

读者可以通过保留小数位数的 round() 函数来得到想要的结果，如保留两位小数。

例 1.2 求 1~100 间所有偶数的和。

任务分析：

本例求在一定范围内（1~100）、满足一定条件（偶数）的若干整数的和，是一个累加和的问题。这类问题的基本解决方法是，设置一个变量（如 sum）作为累加的和，将其初值置为 0，再在指定的范围内（1~100）寻找满足条件（偶数）的整数，将它们一个一个

累加到 `sum` 中。为了处理方便，将正在查找的整数也用一个变量（如 `i`）来表示。
所以，累加过程的 Python 语句为

```
132 sum=sum+i
```

它表示把 `sum` 的值加上 `i` 后再重新赋给 `sum`。

这个累加过程要反复做，可以用结构化程序设计中的循环结构来实现。在循环过程中：

(1) 需要判断 `i` 是否满足问题要求的条件（偶数）。用分支结构实现将满足条件的整数累加到 `sum` 中。

(2) 需要对循环次数进行控制。这可通过 `i` 值的变化进行控制，即 `i` 的初值设为 1，每循环一次加 1，一直加到 100 为止；也可以将 `i` 的初值设为 2，每循环一次加 2，一直加到 100 为止，第二种方式可以省略 `i` 是否为偶数的判断。

基于上述解决问题的思路，就可以逐步明确解决问题的步骤，即解决问题的算法。

算法（algorithm）是一组明确的解决问题的步骤，它产生结果并可在有限时间内终止。可以用多种方式来描述算法，包括用自然语言和流程图。下面主要介绍流程图的使用。

流程图是算法的图形表示法。它用图的形式掩盖了算法的所有细节，只显示算法从开始到结束的整个流程。

对于求 1~100 之间偶数和的问题，可以用流程图来描述解决步骤（算法），如图 1-8 所示。

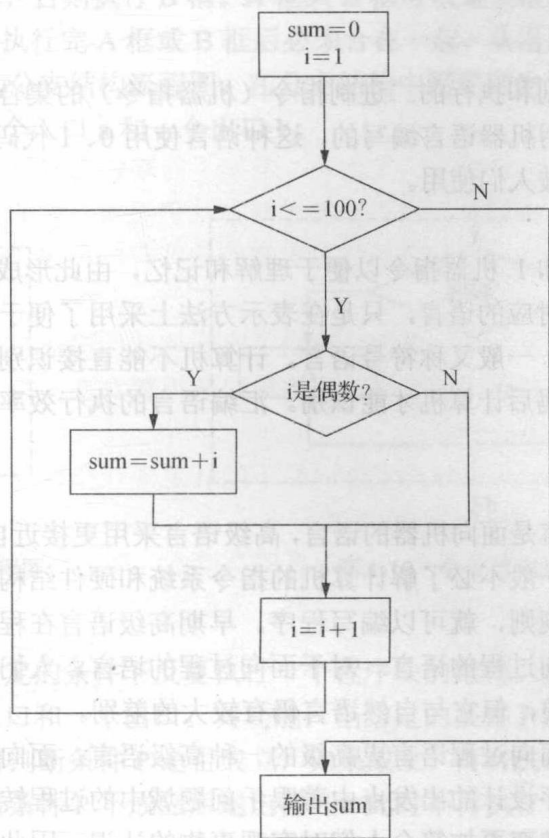


图 1-8 求 100 以内的偶数和流程图