Python

第 1 章 概述

程序设计语言是用于书写计算机程序的语言，是人指挥计算机工作的工具。20 世纪 60 年代以来，世界上公布的程序设计语言已有上千种，但是只有很小一部分得到了广泛的应用。Python 语言从 20 世纪 90 年代初诞生至今，已经成为最受欢迎的程序设计语言之一。

本章介绍程序设计语言的基本概念、程序设计方法、程序运行过程、程序设计方法、Python 安装与开发环境、Python 库、程序打包，最后介绍一个简单的实例。

1.1 程序设计语言

程序设计语言是人与计算机交流的语言。人类需要计算机完成的任务必须用某种程序设计语言书写出来，形成程序，然后交给计算机去执行。程序设计语言经过多年的发展，从机器语言、汇编语言，发展到了高级语言。

1.1.1 机器语言

在计算机发展的早期，使用的程序设计语言称为机器语言。因为计算机的内部电路是由开关和其他电子器件组成的，这些器件只有两种状态，即开或关。一般情况下，“开”状态用 1 表示，“关”状态用 0 表示，计算机所使用的是由 0 和 1 组成的二进制数，所以二进制是计算机语言的基础。

为了能与计算机交流，指挥计算机工作，人们必须学会用计算机语言与计算机交流，即要写出一串由 0 和 1 组成的二进制指令序列并交给计算机执行，这时所使用的语言就是机器语言。

机器语言是面向机器的指令系统，计算机可以直接识别它，且不需要进行任何解释或翻译。机器语言是严格与机器相关的，每台机器的指令格式和代码所代表的含义都是硬性规定的，对不同型号的计算机来说，机器语言一般是不同的。由于使用的是针对特定型号的计算机语言，因此机器语言的运算效率是所有语言中最高的。

尽管机器语言对计算机的工作是直接的、高效的，但是，能够使用机器语言的人还是比较少的。使用机器语言的人必须懂得计算机工作的原理，这对于大部分非专业人士来说是不可能的。

表 1-1 是一个机器语言程序，该程序的功能是实现两个整数相加。

表 1-1　一个机器语言程序

机器语言指令	完成的操作
0001 0000 0010 0000	从内存单元 20 中取数，置于寄存器 A 中
0011 0000 0010 0001	寄存器 A 的数值加上内存单元 21 中的数值，和存入寄存器 A 中
0010 0000 0010 0010	把寄存器 A 的数值存入内存单元 22 中
0000 0000 0000 0000	结束程序运行

从以上程序可以看出，机器语言程序可读性差。另外，由于不同型号计算机的指令系统不同，针对一种型号计算机书写的程序不能直接拿到另一种不同型号的计算机上运行，因此程序可移植性差。

1.1.2　汇编语言

汇编语言也是一种面向机器的语言。为了帮助人们记忆，它采用了符号（称为助记符）来代替机器语言的二进制码，所以又称为符号语言。

因为使用了助记符，所以用汇编语言书写的程序不能被计算机直接识别，需要一种程序将汇编语言翻译成机器语言才能在计算机上执行，这种翻译程序称为汇编程序（assembler）。把汇编语言程序翻译成机器语言程序的过程称为汇编。

表 1-2 所示是一个用汇编语言编写的程序，该程序的功能是实现两个整数相加。

表 1-2　一个汇编语言程序

汇编语言指令	完成的操作
LOAD X	从内存单元 X 中取数，置于寄存器 A 中
ADD Y	寄存器 A 的数值加上内存单元 Y 的数值，和存入寄存器 A 中
STORE SUM	把寄存器 A 的数值存入内存单元 SUM 中
HALT	结束程序运行

汇编语言比机器语言易于读写、调试和修改，用汇编语言写的程序与机器语言程序一样，具有执行效率高、占用内存少等特点，可以有效地访问、控制计算机的各种硬件设备。

但汇编语言仍依赖于具体的处理器体系结构，用汇编语言编写的程序也不能直接在不同类型处理器的计算机上运行，可移植性差。另外，要掌握好汇编语言也不容易，它要求程序员熟悉各种助记符与硬件的关系，所以不被大多数非专业人士所接受。

1.1.3 高级语言

尽管汇编语言极大提高了编程效率，但仍然需要程序员在所使用的计算机硬件上投入大量的精力。另外，汇编语言也很枯燥，因为每条机器指令都需要单独编码。为了提高程序员的效率，把程序员的注意力从关注计算机的硬件转移到解决实际应用问题上来，导致了高级语言的产生与发展。

高级语言是面向用户的、基本上独立于计算机硬件结构的语言。其最大的优点是形式与算术语言和自然语言接近，概念与人们通常使用的概念接近。高级语言的一个命令可以代替几条、几十条，甚至几百条汇编语言指令。高级语言种类繁多，可以从应用角度和对客观世界的描述两个方面对其进行分类。

1. 从应用角度分类

从应用角度来看，高级语言可以分为基础语言、结构化语言和专用语言。

（1）基础语言。基础语言也称为通用语言，其历史悠久，流传很广，有大量的已开发的软件库，拥有众多的用户，为人们所熟悉和接受。属于这类语言的有 FORTRAN、COBOL、BASIC、ALGOL 等。FORTRAN 语言是国际上广为流行、也是使用得最早的一种高级语言，从 20 世纪 90 年代起，在工程与科学计算中一直占有重要地位，且备受科技人员的喜爱。BASIC 语言是 20 世纪 60 年代初为适应分时系统而研制的一种交互式语言，可用于一般的数值计算与事务处理，其结构简单、易学易用、具有交互能力，是许多初学者学习程序设计的入门语言。

（2）结构化语言。20 世纪 70 年代以来，结构化程序设计和软件工程的思想日益为人们所接受，在其影响下，先后出现了一些很有影响的结构化语言。Pascal 语言、C/C++ 语言、Python 语言就是它们中的突出代表。

Pascal 语言是第一个系统地体现结构化程序设计概念的高级语言。由于它模块清晰、控制结构完备、有丰富的数据类型和数据结构、语言表达能力强、可移植性好，而被国内外许多高校用作教学语言。

C/C++ 语言具有功能丰富、表达能力强、运算符和数据类型丰富、使用灵活且方便、应用面广、可移植性好、目标程序效率高等高级语言的特点，同时也具有低级语言的许多特点，如允许直接访问物理地址、能进行位操作、能实现汇编语言的大部分功能、可直接对硬件进行操作等。用 C/C++ 语言编译程序产生的目标程序，其质量可以与汇编语言产生的目标程序相媲美，让 C/C++ 语言成为编写应用软件、操作系统和编译程序的重要语言之一。

（3）专用语言。专用语言是为某种特殊应用而专门设计的语言。一般来说，这种语言应用范围较窄，可移植性和可维护性不如结构化程序设计语言。随着时间的推移，业界已出现了数百种专用语言，应用比较广泛的有 APL 语言、Forth 语言、LISP 语言等。

2. 从对客观世界的描述角度分类

从对客观世界的描述角度来看，程序设计语言可以分为面向过程语言和面向对象语言。

（1）面向过程语言。以“数据结构 + 算法”程序设计范式构成的程序设计语言，称为面向过程语言。前面所述 Pascal 语言、C 语言是面向过程语言的典型代表。

（2）面向对象语言。以“对象 + 消息”程序设计范式构成的程序设计语言，称为面向对象语言。比较流行的面向对象语言有 Java、C++、Python 等。

Java 语言是一种面向对象的、不依赖于特定平台的程序设计语言，具有简单、可靠、可编译、可扩展、多线程、动态存储管理、易于理解等特点，它是一种理想的用于开发网络应用软件的程序设计语言。

C++ 语言是在 C 语言的基础上引入了面向对象的机制而形成的一门计算机编程语言。C++ 继承了 C 语言的大部分特点：一方面，C++ 语言将 C 语言作为其子集，使其能与 C 语言相兼容；另一方面，C++ 语言支持面向对象的程序设计。

1.1.4 Python 语言

Python 是一种面向对象的结构化编程语言，是由荷兰数学和计算机科学研究学会的吉多·范罗苏姆（Guido van Rossum）于 1990 年代初设计的。1989 年圣诞节期间，Guido 为了打发圣诞节的无趣，决心开发一个新的脚本程序。之所以选中 Python（意为蟒蛇）作为该编程语言的名字，是取自英国 20 世纪 70 年代首播的电视喜剧《蒙提·派森的飞行马戏团》（*Monty Python's Flying Circus*）。

Python 逐渐发展为最受欢迎的程序设计语言之一。1995 年，Guido 在弗吉尼亚州的国家创新研究公司继续他在 Python 上的工作，并在那里发布了该软件的多个版本；2000 年 5 月，Guido 的核心开发团队转到 BeOpen.com，并组建了 BeOpen Python Labs 团队；Python 2 于 2000 年 10 月发布，稳定版本是 Python 2.7；2001 年，Python 软件基金会（PSF）成立，PSF 是一个专为拥有 Python 相关知识产权而创建的非营利组织；2004 年以后，Python 的使用率呈线性增长；Python 3 于 2008 年 12 月发布，不完全兼容 Python 2；2011 年 1 月，它被 TIOBE 编程语言排行榜评为 2010 年度语言；2018 年 3 月，该语言作者在邮件列表上宣布 Python 2.7 将于 2020 年 1 月 1 日终止支持；2021 年 10 月，被加冕为最受欢迎的编程语言，20 年来首次被置于 Java、C 和 JavaScript 之上。

Python 语言是一种简单、易学、易读易维护、可扩展性好、可移植性好的程序设计语言。Python 是一种代表简单主义思想的语言。阅读一个良好的 Python 程序就像是读英语一样，使人能够专注于解决问题而不是去搞明白语言本身；因为 Python 有及其简单的文档，所以极易上手；风格清晰统一、强制缩进，使得 Python 易读易维护；Python 提供了丰富的 API 和工具，程序员能够轻松地使用 C 语言、C++ 语言来编写扩充模块；由于

它的开源本质，Python 已被移植到 Linux、Windows、FreeBSD、macOS、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE、PocketPC、Symbian 以及 Android 等平台上。

表 1-3 所示是一个用 Python 语言写的程序，该程序的功能是实现两个整数相加。

表 1-3　一个 Python 程序

Python 语句	完成的操作
x=3	被加数 x，赋值 3
y=4	加数 y，赋值 4
sum=x+y	x 加 y 的和数，存入 sum

该程序可读性好，可移植性好。

1.2　程序设计方法

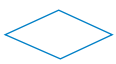
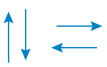
计算机程序的设计方法起源于日常解决问题的方法，流程图是描述问题解决思路的常用工具之一。本节首先认识程序流程图，接着简要介绍两种常用的程序设计方法：结构化程序设计方法和面向对象程序设计方法，最后介绍程序设计的步骤。

1.2.1　程序流程图

程序流程图又称为程序框图，它是用统一规定的标准符号描述程序运行具体步骤的图形表示，流程图着重说明程序的逻辑性与处理顺序，描述计算机解题的逻辑及步骤。流程图是进行程序设计的最基本依据。

流程图常用符号及含义如表 1-4 所示。

表 1-4　流程图常用符号及含义

名　称	符　号	含　义
起、止框		表示程序的开始或结束，框内一般填写“开始”或“结束”
输入、输出框		表示程序的输入、输出，框内填写需要输入、输出的各项
处理框		表示程序中各种处理，框内填写的是用于处理的指令序列
判断框		表示程序中的条件判断，框内填写条件
连接点		当流程图在一个页面画不下时，常用它来表示相对应的连接处
流向线		表示程序的执行流程，箭头指向流程的方向

1.2.2 结构化程序设计方法

结构化程序设计（structured programming）方法又称为面向过程（procedure oriented）的程序设计方法，它是 20 世纪 70 年代由知名的计算机科学家 E.W.Dijkstra 提出来的。该方法指按照层次化、模块化的方法来设计程序，从而提高程序的可读性和可维护性。其主要思想如下。

（1）程序模块化。程序模块化指把一个复杂的程序分解成若干个部分，每个部分称为一个模块。它通常按功能划分模块，使每个模块实现相对独立的功能，使模块之间的联系尽可能地简单。

（2）语句结构化。语句结构化指每个模块都用顺序结构、选择结构或循环结构来实现流程控制。

顺序结构是指顺序执行的结构，即按照程序语句行的书写顺序，逐行执行程序。如图 1-1 所示，先执行语句 A，再执行语句 B，然后执行语句 C。

选择结构又称为分支结构，根据条件成立与否决定执行哪个分支。如图 1-2 所示，当条件成立（真）时执行语句 A，当条件不成立（假）时执行语句 B，二者选一执行。

循环结构又称为重复结构，根据给定的条件，决定是否重复执行某段程序。循环结构有两种：对先判断条件后执行语句（称为循环体）的称为当型循环结构，如图 1-3 所示，当条件成立（真）时执行循环体，条件不成立（假）时退出循环；对先执行循环体后判断条件的称为直到型循环结构，如图 1-4 所示，先执行一次循环体，然后判断条件，条件成立（真）时继续执行循环体，条件不成立（假）时退出循环。

图 1-1　顺序结构图

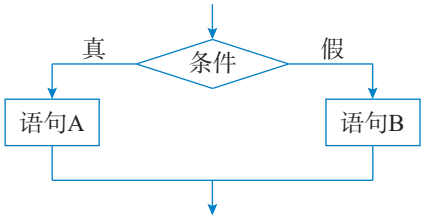


图 1-2　选择结构图

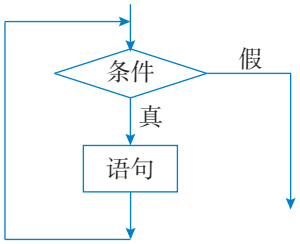


图 1-3　当型循环结构图

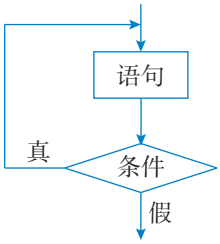


图 1-4　直到型循环结构图

（3）自顶向下、逐步求精的设计过程。自顶向下是指将复杂的、大的问题划分为小问题，找出问题的关键、重点所在，然后用精确的思维定性、定量地去描述问题。逐步求精是指将现实世界的问题经抽象转换为逻辑空间或求解空间的问题，复杂问题经抽象化处理变为相对比较简单的问题，再经若干步抽象（精化）处理，直到求解域中只是比较简单的编程问题，用 3 种基本程序结构即可实现。

（4）限制使用转向语句，如 goto 语句。因为滥用转向语句将使程序流程无规律，程序可读性差。

结构化程序设计方法的有两个优点：第一，程序易于理解、使用和维护，即程序员采用结构化编程方法，便于控制、降低程序的复杂性，因此容易编写程序，而且便于验证程序的正确性，结构化程序清晰易读，可理解性好，程序员能够进行逐步求精、程序证明和测试，以确保程序的正确性，程序容易阅读并被人理解，便于用户使用和维护；其二，提高了编程工作的效率，降低了程序的开发成本。由于结构化编程方法能够把错误控制在最低限度，因此能够减少调试和查错的时间。结构化程序是由一些为数不多的基本结构模块组成，这些模块甚至可以由机器自动生成，从而极大地减轻了编程工作量。因此，结构化程序设计方法得到了广泛应用。

支持结构化程序设计的程序设计语言有 Pascal 语言、C 语言、Python 语言等。

本书第 1 ～ 5 章主要介绍 Python 语言面向过程的程序设计方法。

1.2.3 面向对象程序设计方法

面向对象（object oriented）程序设计方法是一种支持模块化设计和软件重用的实际可行的编程方法。它把程序设计的主要活动集中在建立对象和对象之间的联系上，从而完成所需要的计算。一个面向对象的程序就是实现相互联系的对象集合。由于现实世界可以抽象为对象和对象联系的集合，因此面向对象的程序设计方法更接近现实世界、更自然。

面向对象程序设计中有几个基本概念：对象、消息、类、封装、继承和多态性。

（1）对象。对象是面向对象程序设计的基本要素。对象由一组属性和对这组属性进行操作的一组方法构成。其中，属性描述对象的静态特征，如一个圆由圆心、半径等属性来描述；方法描述对象的动态特征，如输入圆心、半径，输出圆心、半径等都是对圆对象的属性进行操作。

（2）消息。通过向对象发送消息来处理对象。每个对象根据消息的性质来决定要采取的行动，即响应一个消息。

（3）类。类是数据抽象和信息隐藏的工具，它是具有相同属性和方法的一组对象的抽象描述。对象是类的实例。发送给一个对象的所有消息都在该对象的类中来定义，并以方法来描述。

（4）封装。封装是一种组织软件的方法。它的基本思想是把客观世界中联系紧密的元素及相关操作组织在一起，使其实现细节隐藏在内部，以简单的接口对外提供服务。

（5）继承。继承用于描述类之间的共同性质。它减少了相似类的重复说明，体现出一般化及特殊化的原则。例如，可以把“汽车”作为一个一般化的类，而把“卡车”作为一种更具体的类，它从汽车类继承了许多属性及方法，并且允许添加卡车类特有的属性和方法。

（6）多态性。多态性指相同的语句组可以代表不同类型的实体或对不同类型的实体进行操作。

用面向对象程序设计方法编写的程序，其结构与求解的实际问题的结构基本一致，具有很好的可读性和可维护性。另外，利用继承、多态、模板等机制，程序设计者能够很好地实现代码重用，极大地提高设计程序的效率。目前，面向对象程序设计方法已成为主流的程序设计方法，在软件开发过程中被广泛使用。

支持面向对象的程序设计语言有 C++ 语言、Java 语言、Python 语言等。

本书第 6 章介绍 Python 语言面向对象的程序设计方法。

1.2.4 程序设计的步骤

人们用程序设计语言书写程序的过程称为程序设计。程序设计过程包括分析、设计、编码、测试和排错、编写文档等不同阶段。

1. 分析

分析指对于接受的任务进行认真分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法，完成实际问题。

2. 设计

设计指设计出解决问题的方法和具体步骤，这些方法和步骤统称为算法。

3. 编码

编码指将算法翻译成用计算机语言表示的程序。用高级语言编写的程序称为源程序。源程序是文本文件，便于人们阅读和修改。计算机不能直接识别源程序，必须将源程序翻译成机器语言表示的可执行程序，才能在计算机上运行。翻译的方式有两种：一种称为解释方式，另一种称为编译方式。每一种高级语言都配有解释器或编译器，用于完成对源程序的解释或编译。

解释方式是由解释器对源程序逐语句进行语法检查，一边解释，一边执行。如图 1-5 所示，解释结束，程序的运行结束。

编译方式是由编译器对源程序文件进行语法检查，并将之翻译为机器语言表示的二进制程序，即目标程序。程序的编译和执行如图 1-6 所示。

图 1-5 和图 1-6 中编辑需要用到文本编辑器，以实现源代码的输入、修改及存盘等操

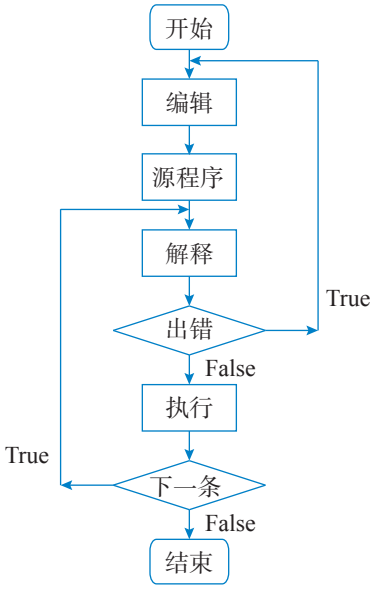


图 1-5　程序的解释和执行

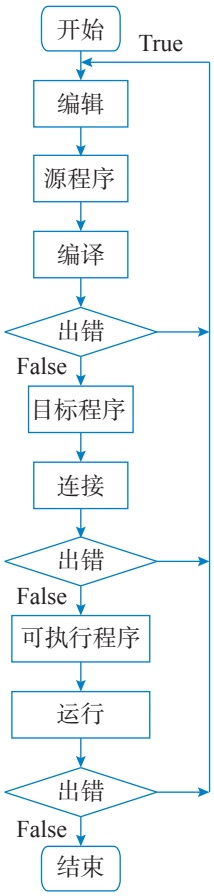


图 1-6　程序的编译和执行

作，形成源程序文件。不同的高级语言源程序文件，其文件扩展名不同。例如，C 语言源程序文件的扩展名为 .c，C++ 源程序文件的扩展名为 .cpp，Python 源程序文件的扩展名为 .py 等。

连接用到程序设计语言的连接器。连接器将编译得到的目标程序与系统提供的库文件代码结合生成可执行程序。

解释方式和编译方式的主要区别包括以下三点。

（1）编译方式是一次性地完成翻译，一旦成功生成可执行程序，则不再需要源代码和编译器即可执行程序；解释方式在每次运行程序时都需要源代码和解释器。

（2）解释方式执行需要源代码，所以程序纠错和维护十分方便；另外，只要有解释器负责解释，源代码可以在任何操作系统上执行，可移植性好。

（3）编译所产生的可执行程序执行速度比解释方式的执行速度更快。

4. 测试和排错

测试和排错，即运行程序、分析结果。运行程序，能得到结果并不意味着程序正确，

要对结果进行分析，看它是否合理。不合理时，要对程序进行调试，即发现和排除程序中的故障，直至结果正确。

5. 编写文档

程序是提供给别人使用的。如同正式的产品应当提供产品说明书一样，正式提供给用户使用的程序必须向用户提供程序说明书，内容应包括程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据，以及使用注意事项等。

1.3 程序的运行过程

计算机程序是由计算机运行的，现代计算机是基于冯·诺依曼体系结构的，如图 1-7 所示。计算机由主机和外设组成，其中主机包含运算器、控制器和内存储器，外设含输入设备、输出设备和外存储器，各个部件之间通过总线（含控制总线、数据总线和地址总线）来通信。

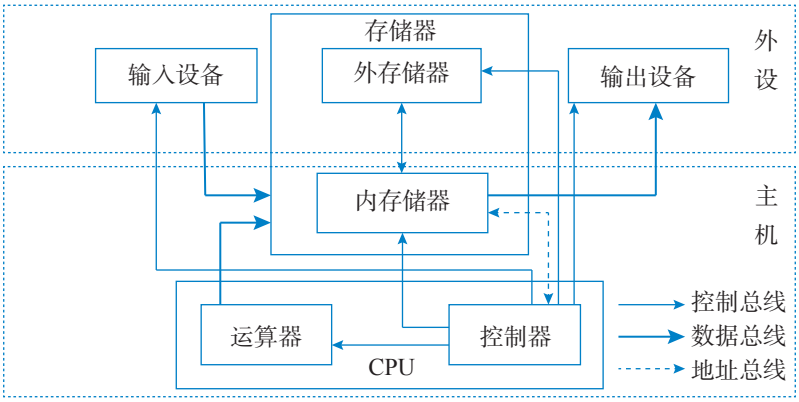


图 1-7 冯·诺依曼体系结构

程序和程序要处理的数据通过输入设备输入内存储器（简称内存），程序的运行结果由输出设备输出。CPU 执行程序的过程如下。

以两个整数相加的程序为例，一台简单体系结构的计算机完成该工作至少需要 4 条指令。这 4 条指令和两个输入的整数在程序开始执行之前存储在内存之中，如图 1-8 所示，程序执行后的结果也存放在内存中。

图 1-8 中，R1、R2、R3 是数据寄存器（data register），用于暂存运算器的运算数据；I 是指令寄存器（instruction register），用于暂存 CPU 即将执行的指令；PC 是程序计数寄存器（program counter register），用于指出下一条即将取出的指令所在的内存地址。程序开始执行时，PC 中存放的 070 表示第 1 条要执行的指令在内存地址为 070 的位置，里面存放的指令为：

```
Load 200 R1
```

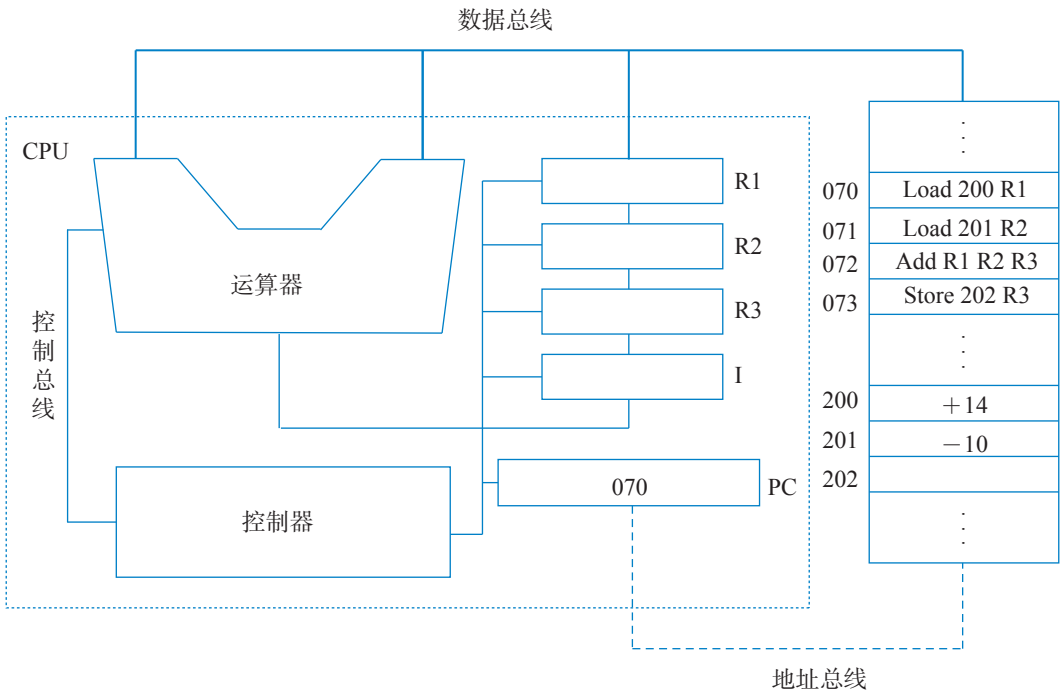


图 1-8　执行前内存和寄存器中的内容

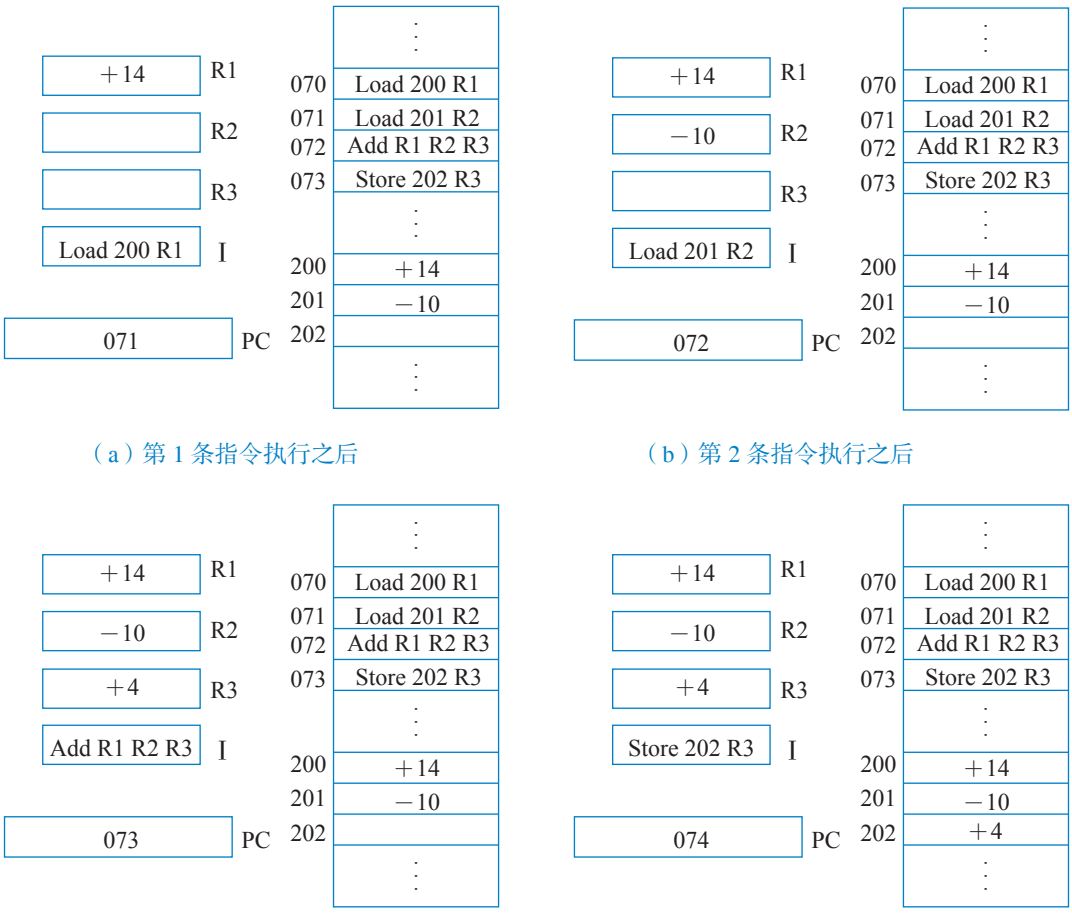
图 1-9 示意了 4 条指令的执行顺序。

图 1-9（a）示意了第 1 条指令执行后，CPU 内各寄存器的状态。指令寄存器中存放的是所执行程序的第 1 条指令，数据寄存器 R1 中存放的是指令执行后的结果，即从内存地址 200 处取出数据（+14）放入寄存器 R1 中，PC 中存放的是下一条要执行的指令的地址（071）。

图 1-9（b）示意了第 2 条指令执行后，CPU 内各寄存器的状态。指令寄存器中存放的是所执行程序的第 2 条指令（Load 201 R2），数据寄存器 R2 中存放的是第 2 条指令的执行结果，即从内存地址 201 处取出数据（−10），存入寄存器 R2 中，PC 中存放的是下一条要执行的指令地址（072）。

图 1-9（c）示意了第 3 条指令执行后，CPU 内各寄存器的状态。指令寄存器中存放的是所执行程序的第 3 条指令（Add R1 R2 R3），数据寄存器 R3 中存放的是第 3 条指令的执行结果，即将寄存器 R1 和 R2 中的数据由运算器进行加法运算，将运算结果（+4）存入寄存器 R3 中，PC 中存放的是下一条要执行的指令地址（073）。

图 1-9（d）示意了第 4 条指令执行后，CPU 内各寄存器的状态。指令寄存器中存放的是所执行程序的第 4 条指令（Store 202 R3），即将寄存器 R3 中的值（+4）写入内存地址 202 所在的位置，PC 中存放的是下一条要执行的指令地址（074），依此类推。



（a）第 1 条指令执行之后　　（b）第 2 条指令执行之后

（c）第 3 条指令执行之后　　（d）第 4 条指令执行之后

图 1-9　4 条指令的执行顺序

1.4　Python 安装与开发环境

一个好的开发环境可以极大提高编程开发的效率，用 Python 开发应用程序也不例外。要用 Python 语言编写程序，需要安装 Python 解释器。

1.4.1　Python 的安装

Python 是纯粹的自由软件、开源项目的优秀代表，其解释器的全部源代码都是开源的。Python 语言解释器是一个轻量级的小软件，支持交互式和批量式两种编程方式，可以在 Python 语言主网站上下载，其安装文件容量为 25 ～ 30MB。下面介绍 Python 的下载和安装步骤。

打开网页 https://www.python.org/down/oads/，进入图 1-10 所示的下载界面。

图 1-10 所示的 Python 官网上同时发行 Python 2.x 和 Python 3.x 两个系列的版本，两个系列版本互不兼容，很多内置函数的实现和使用方式也不一样，Python 3.x 对 Python 2.x

的标准库进行了重新拆分和整合。Python 2.x 的最新版本是 2020 年 4 月发布，之后停止更新。Python 3.x 的最近版本是 Python 3.10.7/3.9.14/3.8.14/3.7.14。需要说明的是，同系列版本中，高版本比低版本更加完善。

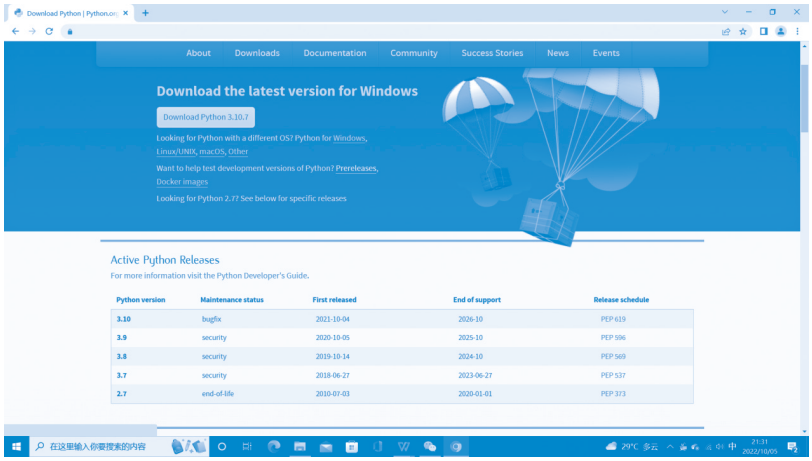


图 1-10　Python 下载界面

根据所用的操作系统，在图 1-10 中选择对应的 Python 3.x 系列安装程序。在图 1-10 中单击 Download Python 3.10.7 按钮即可下载 Python 的稳定版本；随着 Python 语言的发展，此页面处会有更新的稳定版本出现。本章内容以 Windows 操作系统版本 Python 3.10.6 为例进行安装和环境配置。如果在其他操作系统下，请打开图 1-10 中下部的相应链接进行选择。

找到所下载的文件 python-3.10.6-amd64，如图 1-11 所示，双击，即可启动 Python 解释器的安装，然后出现图 1-12 所示的安装程序引导过程启动界面，勾选 Add Python 3.10 to PATH 复选框，单击 Install Now 按钮，出现图 1-13 所示的安装界面，安装结束后，出现图 1-14 所示的安装成功界面。

图 1-11　下载的 Python-3.10.6-amd64 文件

图 1-12　安装程序引导过程启动界面

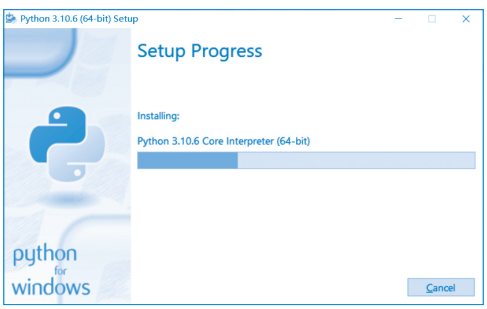


图 1-13　Python 3.10.6 安装界面

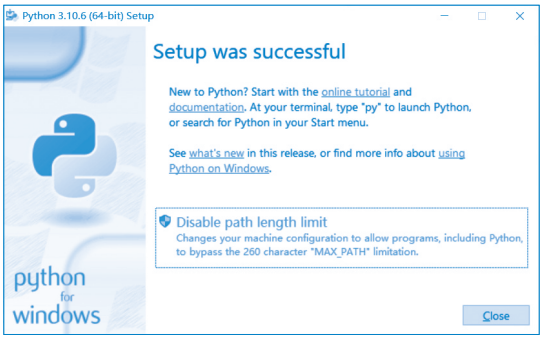


图 1-14　Python 3.10.6 安装成功界面

Python 安装包在系统中安装了一批与 Python 开发和运行相关的组件，其中最重要的两个是 Python 命令行和 Python 集成开发环境（Integrated Development and Learning Environment，IDLE）。图 1-15 示意了 Windows“开始”菜单中 Python 3.10 所包含的组件。

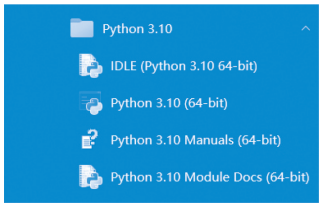


图 1-15　Python 3.10 所包含的组件

1.4.2　Python 程序运行方式

运行 Python 程序有交互式和文件式。

交互式是指 Python 解释器即时响应用户输入的每条代码。

文件式（也称为批量式）是指用户将 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。

交互式一般用于调试少量代码，文件式是最常用的编程方式。

【例 1.1】编写一个程序，运行输出“Hello China!”。

学习编程语言都是从编写运行最简单的 Hello 程序开始的，即程序运行时在屏幕上输出“Hello world!”，本例中改为输出“Hello China!”。这个程序虽小，却是初学者接触编程语言的第一步。使用 Python 语言编写的 Hello 程序如下：

```
print("Hello China!")
```

1. 交互式启动和运行程序

交互式启动和运行程序的方法有以下两种。

第一种方法是以命令方式启动。单击图 1-15 中的 Python 3.10(64-bit)，在所出现界面的命令提示符 >>> 后输入例 1.1 中的程序代码，按 Enter 键后显示输出“Hello China!”，如图 1-16 所示。

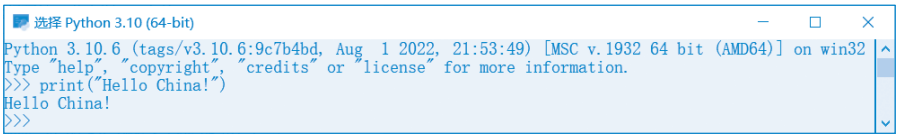


图 1-16　以命令方式启动交互式 Python 运行环境

第二种方法是通过调用安装的 IDLE 来启动 Python 运行环境。单击图 1-15 中的 IDLE(Python 3.10 64-bit)，启动 IDLE 的交互式 Python 运行环境，在该环境下运行 Hello 程序的效果如图 1-17 所示。

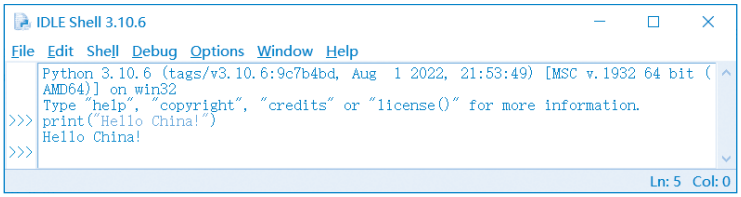


图 1-17　通过 IDLE 启动交互式 Python 运行环境

2. 文件式启动和运行程序

文件式启动和运行程序也有以下两种方法。

第一种方法是用文本编辑器（如记事本等）按照 Python 语法格式编写代码，如图 1-18 所示，并保存为 .py 格式的文件（此处命名为“hello.py”），然后运行 Windows 操作系统下的 cmd.exe 程序，在命令提示符后输入 python hello.py，即可得到图 1-19 的结果。

图 1-18　用记事本创建 hello.py 文件

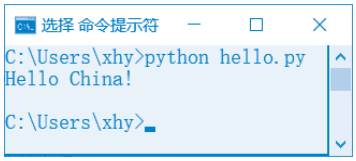


图 1-19　以命令方式运行 Python 程序文件

第二种方法是打开 IDLE，在菜单栏中选择 File → New File 命令，在显示的窗口中输入代码，并保存为 hello.py，如图 1-20 所示，然后选择 Run → Run Module 命令运行程序，得到图 1-21 所示的结果。

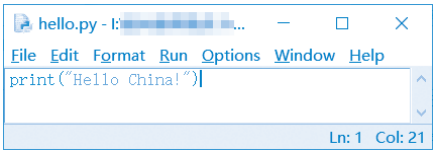


图 1-20　以 IDLE 方式创建 Python 程序文件

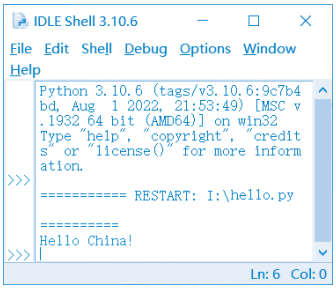


图 1-21　以 IDLE 方式运行 Python 程序文件

1.4.3 Python 开发环境

“工欲善其事，必先利其器。”在开始学习使用 Python 之前，选择一个合适的集成开发环境，有利于快速上手 Python，以期在学习中起到事半功倍的效果。下面介绍常用的 Python IDE 工具。

1. IDE 的总体分类

IDE 总体可以分为文本工具类和集成工具类。

文本工具类 IDE 包括 IDLE、Sublime Text、Notepad++、Vim&Emacs、Atom、Komodo Edit 等。

集成工具类 IDE 包括 PyCharm、Anaconda&Spyder、Wing、PyDev&Eclipse、Visual Studio Code、Canopy 等。

另外，还可以把所有的 IDE 分类为通用型 IDE 和专注于数据分析的 IDE。

2. 通用类型的 Python IDE

（1）IDLE。如 1.4.2 节所述，IDLE 是 Python 自带的、默认的、入门级编程工具，适用于 Python 入门学习，功能简洁、直观，适合代码量在 300 行以内的程序。

（2）Sublime Text。Sublime Text 是专门为程序员开发的第三方专用编程工具，它支持代码高亮、自动补齐、多种颜色搭配等多种编程风格。Sublime Text 包含收费版本和免费版本，两者功能相同。不注册使用的就是免费版本。

（3）Wing。Wing 是非常专业的 Python 集成工具类 IDE。它是由商业公司维护的收费工具，提供了丰富的调试功能，也提供了版本控制和版本同步，适合多人共同开发，适合用于编写数千行至上万行的程序。

（4）Visual Studio Code。Visual Studio Code 是在 Windows、macOS 和 Linux 上运行的独立源代码编辑器。它是 JavaScript 和 Web 开发人员的理想选择，几乎可支持任何编程语言的扩展，如适用于 Visual Studio Code 的 Python 扩展，并提供了视觉提示和工具，可让用户更好、更快地编写 Python 代码。

（5）Eclipse。Eclipse 是早年为 Java 程序员开发的开源 IDE。用户可以通过 PyDev 在 Eclipse 里配置 Python 的开发环境。用 Eclipse 配置 Python 开发环境，很多地方需要用户自己来定义，相对比较复杂，用户需要具备较好的专业经验才能配置好 Python IDE。

（6）PyCharm。PyCharm 是一款专门面向 Python 的全功能集成开发环境。PyCharm 与 Sublime Text 一样，分为社区（免费）版和收费版。绝大多数程序用社区的免费版就可以完成。

PyCharm 是所有的集成类工具中相对简单且集成度较高的，且使用人数最多，适合编写较大、较复杂的程序。

3. 科学计算与数据分析领域 Python IDE

（1）Canopy。Canopy 是由商业公司提供和维护的第三方工具。该工具是收费的，且

价格较贵。Canopy 支持 500 多个第三方库，是科学计算领域中集成度较高且使用方便的 IDE。

（2）Anaconda。Anaconda 是一款第三方、开源、免费的工具，支持 800 多个第三方库，包含多个主流的 Python 开发调试环境，十分适合在数据处理和科学计算领域使用。由于 Anaconda 是一款跨平台工具，因此它可以很好地在 Windows、Linux 和 macOS 上使用。

① Anaconda 的来源。Anaconda 和 Canopy 都是由 Travis Oliphant 开发和领导设计的。Travis Oliphant 也是 SciPy、NumPy 和 Numba 的创造者，Anaconda 公司的创始人和董事，NumFOCUS 公司的创始人，以及 Quansight 公司的 CEO。2008 年，Travis Oliphant 领导开发了 Canopy 工具。但是 Travis Oliphant 的开源开放理念与公司坚持 Canopy 收费的理念不同，因此在 2012 年，Travis 离开了原公司，带领新的团队开发了免费开源的 Anaconda 平台。由于 Anaconda 是开源免费的，因此获得了来自全球的优秀工程师共同改进，并将平台打造得非常好用。

② Anaconda 的本质。Anaconda 是一个集成各类 Python 工具的集成平台，将很多第三方的开发调试环境集成到一起。也就是说，Anaconda 是一个集合，包括 conda、某版本 Python、一批第三方库。

③ Anaconda 中的 conda。conda 是一款开源的包管理系统和环境管理系统，可运行在 Windows、Linux 和 macOS 上，还可快速安装、运行和更新包及其依赖项，也可轻松地在计算机上创建、保存、加载和切换环境。它是为 Python 程序而创造的，但它可以打包和分发任何语言的软件。

④ Anaconda 中的 Spyder。Spyder 是一款编写和调试 Python 语言程序非常优秀的第三方工具，由“Python 之父”吉多·范罗苏姆参与开发而成。Sypder 的优点是启动速度较快、功能简单、容易上手。与 MATLAB 一样，Sypder 对数据分析者非常有用，可满足数据处理和科学计算方面的各种需求。

⑤ Anaconda 中的 IPython。IPython 是一个能调用 Python 解释器的交互式编程环境，它是一个功能更强的交互式 Shell，能够显示很多的图形图像，适合进行交互式的数据可视化以及与 GUI 相关的应用。IPython 是一个前台的显示脚本，核心的功能是用作后台的 Python 解释器。

本书所有代码均使用 Python 自带的 IDLE 调试通过。

1.4.4 Python 文件名

文件的扩展名表示了文件的类型。在 Python 中，不同扩展名的文件也表示了不同的文件类型。常见的扩展名及其含义如下。

.py：是 Python 源文件的扩展名，该文件由 Python 解释器负责解释执行。

.pyw：是 Python 源文件的扩展名，常用于图形界面程序文件。

.pyc：是 Python 字节码文件的扩展名，用户不能用文本编辑器查看该类型文件的内容。

.pyo：是优化后的 Python 字节码文件的扩展名。从 Python 3.5 开始不再支持该类型的文件。

.pyd：一般是由其他语言编写并编译的二进制文件的扩展名，常用于实现某些软件工具的 Python 编程接口插件或 Python 的动态链接库。

1.5 Python 库

Python 库（又称为模块 module）是一个包含若干函数定义、类定义或者常量的 Python 源程序文件。这里的模块与儿童玩的积木模块类似，通过各种形状积木模块的拼接，可以轻易搭出各种造型或实物模型。程序员通过 Python 模块的调用，可以较快地完成某些问题的程序设计。通过调用模块进行编程的方法称为“模块编程”，它是 Python 语言的重要特点之一。Python 库分为内置函数、标准模块和扩展模块（又称第三方库）。内置函数是由 Python 解释器提供的，这些函数不需要导入任何模块即可直接使用。内置函数将在本书后续章节陆续介绍。本节主要介绍标准模块的概念和第三方库的概念与管理。

1.5.1 标准模块

标准模块在安装 Python 时被一起安装。一般需要使用 import 指令导入模块，才能使用对应模块中的函数。下面列出了 Python 中部分常见的标准模块。

math 模块：是数学计算的标准函数库，支持整数和浮点数运算。

random 模块：是生成随机数的函数库，主要用于生成随机数。

date、time 模块：是显示日期和时间的标准函数库，用于从系统中获取时间，以用户选择的格式输出。

turtle 库：图形绘制函数库。

os 模块：用于提供系统级别的操作，如对文件和目录进行操作。

sys 模块：用于提供对解释器相关的操作。

json 模块：用于处理 JSON 字符串。

logging：用于便捷记录日志且线程安全的模块。

hashlib 模块：用于加密相关操作，代替了 md5 模块，主要是提供 SHA1、SHA224、SHA256、SHA384、SHA512 和 MD5 算法。

1.5.2 第三方库

强大的标准库奠定了 Python 发展的基石，丰富和不断扩展的第三方库是 Python 壮大的保证。进入 PyPI 官网可以看到发布的第三方库已达十多万种，众多的开发者为 Python

贡献了自己的力量。下面列出了一些常用的第三方库。

openpyxl：用于读写 Excel 文件。

pymssql：用于操作 Microsoft SQL Server 数据库。

NumPy：用于数组计算与矩阵计算。

SciPy：用于科学计算。

Pandas：用于数据分析。

Matplotlib：用于数据可视化或科学计算可视化。

Scrapy：提供爬虫框架。

sklearn：用于机器学习。

TensorFlow：用于深度学习。

在默认情况下，安装 Python 时不会安装任何扩展库或第三方库，这些库需要单独安装。一般需要使用 import 指令导入模块，才能使用对应模块中的函数。Python 自带的 pip 工具是管理扩展库的主要方式，支持 Python 扩展库的安装、升级和卸载等操作。

1. 在线安装指定模块

命令格式：pip install Package[= =version]

使用说明：可以使用方括号内的格式指定扩展库版本。

例如，在线安装 Matplotlib 扩展包。操作命令和运行过程如图 1-22 所示。

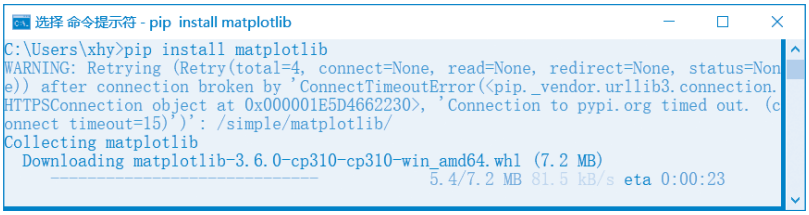


图 1-22　在线安装 Matplotlib 扩展包

2. 列出已安装模块及其版本号

命令格式：pip freeze[> filename.txt]

使用说明：我们可以使用重定向符 > 把扩展库信息保存到文件 filename.txt 中。

例如，用 pip freeze 列出已安装模块及其版本号。操作命令和运行结果如图 1-23 所示。

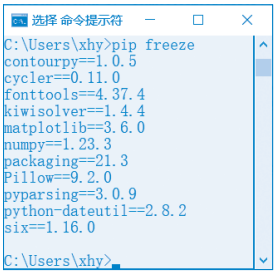


图 1-23　列出已安装模块及其版本号

3. 下载第三方库的安装包

命令格式：pip download Package

例如，用 pip 下载第三方库 Pandas 的安装包并验证下载成功。操作命令和运行结果如图 1-24 和图 1-25 所示。

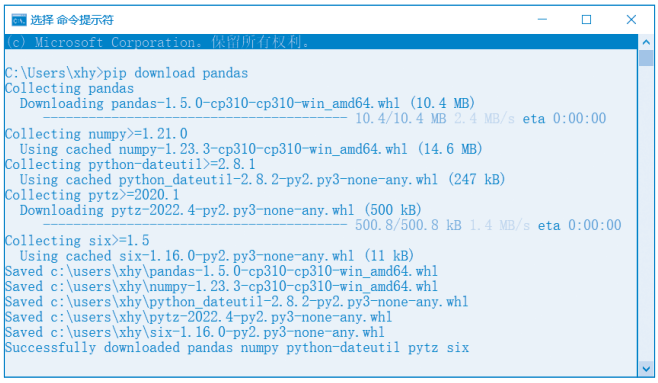


图 1-24　下载第三方库 Pandas

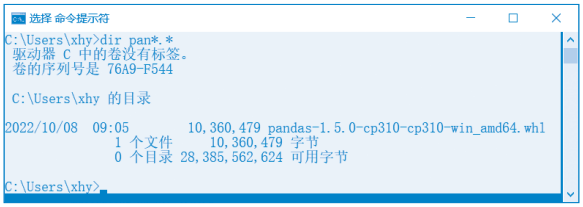


图 1-25　显示下载的第三方库 Pandas

4. 离线安装指定模块

命令格式：pip install Package.whl

对于大部分扩展库，使用 pip 工具直接在线安装都会成功。但是，有时会因为缺少 VC 编辑器或依赖文件而失败。在 Windows 平台上，如果在线安装扩展库失败，此时可以按上述方法下载扩展库编译好的 WHL 文件，然后在命令提示符环境中使用 pip 命令进行离线安装。

例如，离线安装上文中下载的 Pandas 库。操作命令和运行结果如图 1-26 所示。

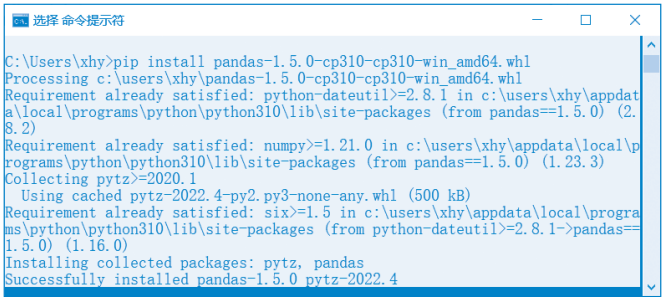


图 1-26　离线安装第三方库 Pandas

5. 升级指定模块

命令格式：pip install --upgrade Package

例如，升级 Pandas 库。操作命令和运行结果如图 1-27 所示。

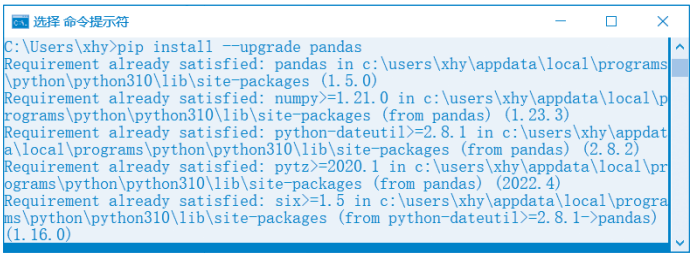


图 1-27　升级第三方库

6. 卸载指定模块

命令格式：pip uninstall Package [==version]

例如，卸载 Pandas 库。操作命令和运行结果如图 1-28 所示。

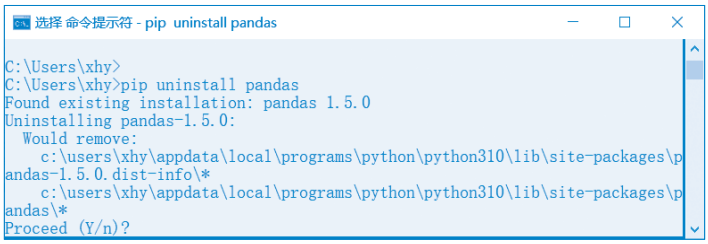


图 1-28　卸载 Pandas 库

7. 帮助信息

命令格式：pip -h

通过 pip -h 可以列出 pip 命令的用法和所有选项的功能。图 1-29 显示了操作命令和运行结果。

图 1-29　显示帮助信息

1.6 程序打包

前面介绍的 Python 程序运行方式即交互式运行方式和文件批量式运行方式都离不开 Python 编程环境。实际中，用户只对应用程序的使用感兴趣，这时，在没有安装编程环境的计算机上运行 Python 程序就变得十分重要。

一种解决办法就是把 Python 程序转换为二进制可执行程序（EXE 文件），这个过程称为打包。打包之后再发布的程序可以在没有安装 Python 环境和相应扩展库的系统中运行，从而极大地方便了用户。

程序打包需要专门的打包工具或软件。常用的打包工具有：py2exe（仅适用于 Windows 平台）、PyInstaller、cx_Freeze 等。

以 PyInstaller 为例，先使用 pip 安装，如图 1-30 所示。安装好后，在命令提示符环境中使用的命令格式如下。

```
pyinstaller 选项 Python 源文件
```

例如，将前述 hello.py 文件打包的命令为：pyinstaller hello.py，该命令可将 hello.py 及其所依赖的包打包成当前所用操作系统平台上的可执行文件，如图 1-31 所示。

图 1-31 中显示出了执行命令时的详细生成过程。当生成完成后，将会在图 1-31 中的 example1 目录下生成一个 dist 目录，并在该目录下生成一个名称为 hello 的文件夹，在该文件夹下有 hello.exe 文件，如图 1-32 所示，该文件就是 PyInstaller 工具生成的 EXE 程序。执行 hello.exe，将会看到输出结果：Hello China!。

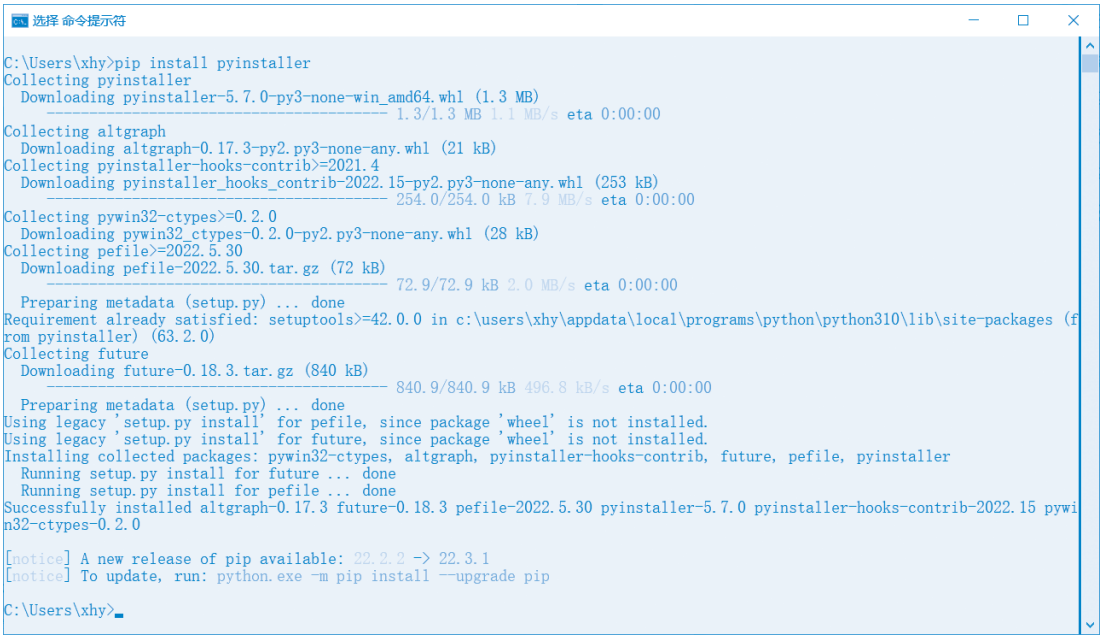


图 1-30　用 pip 安装 PyInstaller

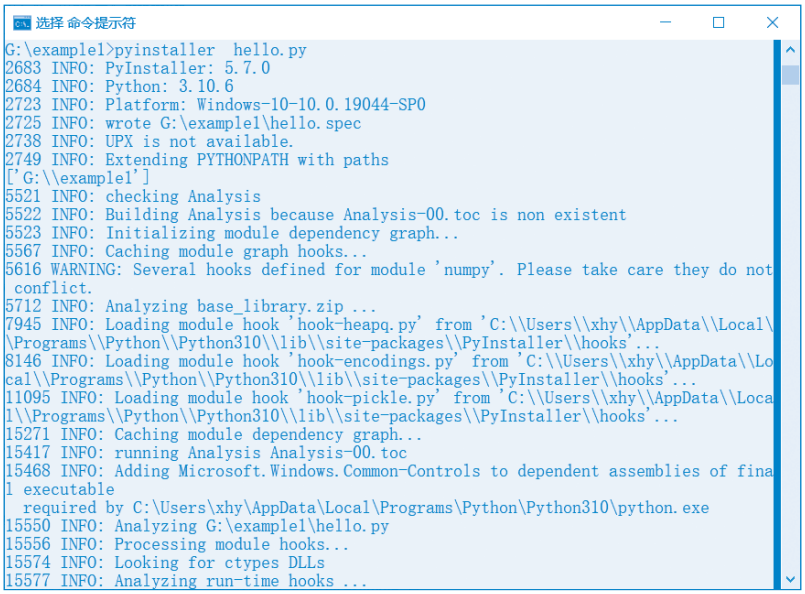


图 1-31　用 PyInstaller 打包 hello.py 程序

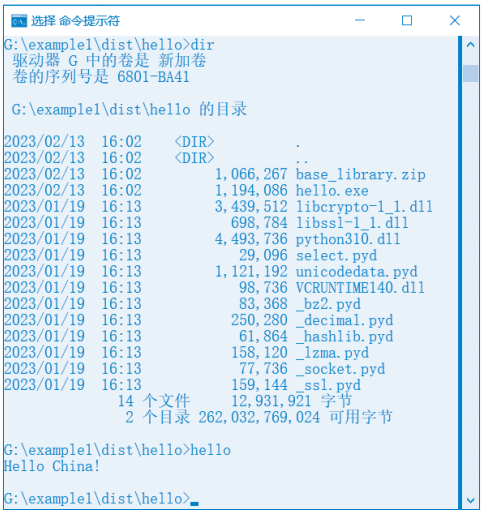


图 1-32　PyInstaller 生成的可执行文件

PyInstaller 工具是跨平台的，它既可以在 Windows 平台上使用，也可以在 macOS 平台上运行。在不同的平台上使用 PyInstaller 工具的方法是相似的。

其他打包工具的使用方法在此不再赘述。在需要的时候，请读者自行查阅相关文档学习。

1.7　应用举例

本节通过一个应用实例，将本章前面所介绍的知识进行具体的运用。

【例 1.2】编写程序，根据圆的半径，计算圆的周长和面积。

根据数学上所学知识，给出圆的半径 radius，计算周长 girth 和面积 area 的公式分别为：

girth = 2*π*radius

area = π*radius*radius

根据前述分析，可以设计出该问题的程序流程图，如图 1-33 所示。

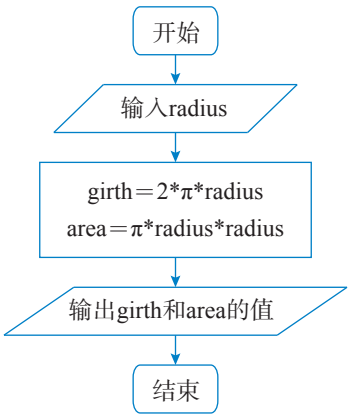


图 1-33　计算圆的周长和面积程序流程图

图 1-33 中，程序从“开始”的位置出发，沿着箭头的流向执行，分别是：输入半径、计算周长和面积、输出周长和面积，最后到“结束”的位置，表示程序结束。

大多数程序都可以按上述流程图所表示的步骤来进行设计，即：输入→计算→输出。

设计完成后，即可进入编码阶段。这一阶段可以借助 1.4.2 节结构化程序设计方法和 1.4.3 节面向对象的程序设计方法，使用 Python 语言来编写。

使用面向过程的程序设计方法编写的源代码如下所示。

```
1  #  1_2.py
2  import math
3
4  radius = eval(input(" 请输入半径: "))
5
6  girth = 2 * math.pi * radius
7  area = math.pi * radius *radius
8
9  print(" 半径 = ", radius)
10 print(" 周长 = ", qirth)
11 print(" 面积 = ", area)
```

其中，第 1 行是注释，说明了源程序的文件名，Python 源程序文件的扩展名是 .py；第 2

行导入 math 库，因为程序中要用到库里的常数 pi（π）；第 4 行由用户输入圆的半径；第 6 行和第 7 行分别使用求圆的周长和面积的公式，计算圆的周长和面积；第 9 ～ 11 行，输出半径、周长和面积。

使用面向对象的程序设计方法编写的 Python 源程序如下所示。

```
1  # 1.2_1.py
2  import math
3
4  class Circle:
5      def __init__(self,r):
6          self.__radius = r
7      def eval_girth(self):
8          return 2 * math.pi * self.__radius
9      def eval_area(self):
10         return  math.pi * self.__radius * self.__radius
11
12 ra = eval(input("请输入圆的半径："))
13 circle1 = Circle(ra)
14 print(" 周长 = ", circle1.eval_girth())
15 print(" 面积 = ", circle1.eval_area())
```

其中第 4 ～ 10 行是圆类定义，第 4 行的 class 是 Python 语言定义类的关键字，Circle 是类的名称，冒号（:）是类体的前导符号，类体各行通过行首的缩进与类形成包含关系，类 Circle 封装了类的属性 self.__radius 和方法“__init__”“eval_girth”“eval_area”，第 12 ～ 13 行定义了类 Circle 的对象 circle1，第 14 ～ 15 行对象 circle1 通过调用其方法 eval_girth() 和 eval_area() 来完成对对象的操作，即求计算圆的周长和面积。

Python 程序通过缩进来体现代码之间的逻辑关系。除了以上的类定义外，第 3 章和第 4 章将介绍选择结构、循环控制结构、异常处理、函数定义等，第 1 行末尾的冒号（:）和接下来一行的缩进表示了一个代码块的开始，缩进结束则表示代码块结束。

在同一个程序中，要求各个级别的代码块缩进量必须相同。在 IDLE 开发环境中，一般默认的缩进量是 4 个空格。

所编写的代码必须通过 Python 解释器解释。解释器一方面能检查源代码是否有语法错误，另一方面对没有语法错误的代码解释为机器语言表示的二进制程序，以便计算机执行。我们可以使用前面介绍的集成开发环境来完成代码编写、测试、排错。本章使用 Python 自带的 IDLE，分别建立面向过程的源程序文件 1_2.py 和面向对象的源程序文件 1.2_1.py，然后运行程序即可。程序运行时，当输入半径的值为 1 时，程序即可求出半径为 1 的圆的周长和面积。下面是程序的运行结果。

```
>>>
请输入半径：1
 半径 =  1
 周长 =  6.283185307179586
 面积 =  3.141592653589793
```

可见，半径为 1 的圆的周长为 6.283185307179586，面积为 3.141592653589793。

正式提供给用户使用的程序必须向用户提供程序说明书。本程序的说明书如表 1-5 所示，内容应包括程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据，以及使用注意事项。

表 1-5　例 1.2 程序的说明书

程 序 名 称	1_2.py
程序功能	根据圆的半径求圆的周长和面积
运行环境	IDLE 或其他 Python 解释器
程序的装入和启动	在 IDLE 下打开程序文件，然后按 F5 键；有关其他解释器的装入和启动方法，请读者自行查阅对应解释器的帮助文档
需要输入的数据	圆半径的值
注意事项	半径大于或等于 0 才有意义

因为本书的示例大部分都是用于学习，为了突出重点，后面将主要介绍问题分析、设计、编码、测试和排错等步骤，编写文档部分将弱化。

1.8　本章小结

程序设计语言是人与计算机交流的语言。程序设计语言从机器语言、汇编语言，发展到了高级语言。Python 是一种面向对象的高级程序设计语言。

流程图是进行程序设计的最基本依据。常用的程序设计方法有结构化程序设计方法和面向对象程序设计方法。

程序设计过程包括分析、设计、编码、测试和排错、编写文档等不同阶段。

程序和程序要处理的数据通过输入设备输入内存，CPU 从内存取指令、分析指令和执行指令，所有指令都被执行完后得到程序的运行结果，程序的运行结果由输出设备输出。

要用 Python 语言编写程序，需要安装 Python 解释器。我们可以在 Python 语言官方网站上下载 Python 解释器进行安装，也可以选用第三方集成开发环境，如 PyCharm、Visual Studio Code 等。

使用库或模块可以提高编程效率。Python 自带的库称为内置函数，安装好 Python 后即可使用。对于第三方提供的扩展库，我们可以使用 Python 的 pip 工具来对其进行下载安

装、升级、卸载等操作。

程序打包是指把 Python 程序转换为二进制可执行程序。程序打包需要专门的打包工具或软件。常用的打包工具有 py2exe（仅适用于 Windows 平台）、PyInstaller、cx_Freeze 等。

大多数应用程序按输入→计算→输出的步骤进行设计。

Python 程序的代码块依靠缩进来体现各自的逻辑关系。

习题

第 1 章
补充习题

1. 编写程序，在屏幕上输出“中国，你好！”。
2. 编写程序，在屏幕上输出下列图形。

```
* * * * *
* * * * *
* * * * *
```

3. 编写程序，输入矩形的长和宽，求矩形的周长和面积。