高等院校计算机应用系列教材

Python 程序设计基础教程

刘　建　邹　杰　时月梅　主　编

吴春容　贾　鹏　副主编

清華大學出版社

北　京

内容简介

本书以 Python 3.9.10 为基础，围绕 Python 3 版本进行全面讲解，内容以实用为基本目标，由浅入深，循序渐进，实用性强，可操作性强。在每一章的讲解过程中，都穿插了大量的实例，并给出了源代码和运行测试结果，可帮助读者很好地理解和掌握每个知识点。

本书共分为 9 章，具体包括 Python 概述、Python 语言基础、流程控制语句、字符串和正则表达式、函数和模块、组合数据类型、面向对象编程、文件操作和异常、项目实训等内容。第 9 章配备的 10 个综合实训项目实用性强，可帮助读者进一步理解 Python 的应用。

本书内容全面，重视实际应用技能的培养，使学生能够学以致用。本书可作为高等院校计算机相关专业教材，还可供计算机入门者阅读参考。

图书在版编目(CIP)数据

Python 程序设计基础教程 / 刘建, 邹杰, 时月梅主编.
北京 : 清华大学出版社, 2026. 5. -- (高等院校计算机应用系列教材). -- ISBN 978-7-302-71280-0

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 20265BA883 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：妙思品位
责任校对：马遥遥
责任印制：宋　林

出版发行：清华大学出版社
网　　址：https：//www.tup.com.cn，https：//www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：北京瑞禾彩色印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**21.5　　**字　　数：**551 千字
版　　次：2026 年 5 月第 1 版　　**印　　次：**2026 年 5 月第 1 次印刷
定　　价：78.00 元

产品编号：110570-01

前　言

随着云计算、物联网、大数据、人工智能、移动互联网等技术全面渗透到社会生产、工作、生活的各个方面，计算机科学技术在当今社会呈现出蓬勃发展的态势。计算机已然成为人们生活中不可或缺的存在，掌握计算机相关知识也逐渐成为衡量个人素养的重要指标。编程语言是能深入探究计算机运行机制的基础工具，而其中 Python 又是当下主流的编程语言之一，因此作为一名程序员或编程爱好者，了解并掌握一些关于 Python 的编程方法和技巧，是很有必要的。

本书内容共分为 9 章，具体包括 Python 概述、Python 语言基础、流程控制语句、字符串和正则表达式、函数和模块、组合数据类型、面向对象编程、文件操作和异常、项目实训等内容。

编者在编写本书时，按照以下原则进行：

第一，面向高等院校相关专业学生，立足于概念的深入浅出；

第二，内容组织以够用为度，着重概念引入和知识轮廓的构建；

第三，理实结合，强调知识的实用性，注重学生操作能力的培养。

本书由刘建、邹杰、时月梅任主编，吴春容、贾鹏任副主编。全书由时月梅、吴春容、贾鹏校对、统稿，由刘建审稿。本书是作者根据多年讲授计算机相关课程的教学实践经验编写而成的，内容由浅入深，循序渐进，实用性强，可操作性强，每一章精选了大量案例，并且所有案例都能在 IDLE(integrated development and learning environment，集成开发与学习环境)上调试通过，可直接引用。

本书在编写过程中，得到了四川大学、电子科技大学、成都理工大学、成都信息工程大学、西南石油大学和四川师范大学相关专家、教授的大力支持和帮助，此外，成都文理学院人工智能与大数据学院胡念青和陈坚也提出了很多宝贵的意见和建议，在此一并表示衷心感谢。

为便于教学，本书提供教学大纲、授课教案、教学课件、案例代码等教学资源，读者可通过扫描下方二维码下载。


教学大纲


授课教案


教学课件


案例代码

由于编者水平有限，书中难免存在欠妥和疏漏之处，敬请各位专家及读者批评指正。

服务邮箱：476371891@qq.com。

编写组

2026 年 1 月

目　录

第 1 章

Python 概述

本章重点内容

- 计算机基础；
- 认识 Python；
- Python 开发环境搭建。

1.1 计算机基础

1946 年，美国宾夕法尼亚大学成功研制出世界上第一台通用电子计算机——ENIAC (electronic numerical integrator and computer，电子数值积分计算机)。现代电子计算机之父冯·诺依曼(John von Neumann)指出计算机应当按照以下模式工作。

(1) 计算机中的指令和数据均以二进制形式存储，指令由操作码和地址码组成。

(2) 像存储数据一样存储程序。

(3) 指令的执行是顺序的，即一般按照指令在存储器中的存放顺序执行，程序的分支由转移指令实现。

(4) 计算机由运算器、控制器、存储器、输入设备和输出设备五部分组成。

运用“存储程序”和“程序控制”相结合的原理，将程序和数据存放在内存中，在程序的控制下自动完成操作。这种结构一直延续至今，因此现在的一般计算机被称为冯·诺依曼结构计算机。

计算机最初作为一种计算工具应用于各个领域，而在计算机问世之前，人类社会就已有各种各样的计算工具。纵观计算工具的发展历史，人类的计算工具已历经结绳记事、算筹、算盘、计算尺、机械计算机、电子管计算机等多个阶段，如今正朝着超导计算机、量子计算机、DNA 计算机等方向探索。

1.1.1 计算机结构

计算机作为20世纪最伟大的发明之一，其结构和工作原理相比传统计算工具要复杂得多。一个完整的计算机系统可分为硬件系统和软件系统两大部分。计算机硬件系统，即遵循冯·诺依曼结构的五大物理设备组件；而计算机软件系统通常包含系统软件和应用软件，是能完成一定功能的各种算法与数据的集合。计算机系统结构如图 1.1 所示。

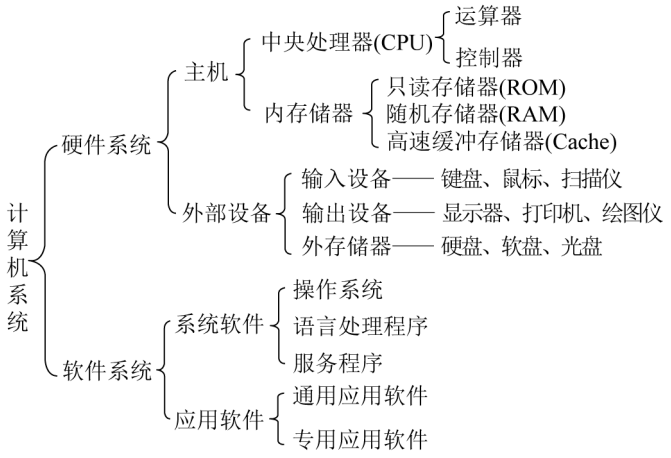

图 1.1　计算机系统结构

1.1.2　计算机语言

计算机语言，也称程序设计语言，是用于人与计算机之间通信的语言，是人机之间传递信息的媒介。计算机系统最显著的特征是，人们能用一种语言向机器传达指令，让机器自动、高速地完成指定工作。为使计算机能够完成各类工作，必须构建一套专门用于编写计算机程序的数字、字符及语法规则体系。由这些数字、字符和语法规则组成各种指令(或各类语句)，它们所构成的，便是计算机所能接受的语言。

程序设计语言经过多年发展，从最初的机器语言、汇编语言阶段，逐步发展到了现今的高级语言阶段。

1. 机器语言

计算机发明之初，人们只能用低级的语言命令计算机工作，换言之，就是只能写出一串串由“0”和“1”组成的指令序列交由计算机执行，这种计算机能够直接识别的语言就是机器语言。使用机器语言编程十分痛苦，阅读程序如同天书，修改程序错误时更是如此。

一条机器语言指令通常称为一条指令。指令是不可分割的最小功能单元。而且，每台计算机的指令系统往往各不相同，因此，在一台计算机上执行的程序，若想移植到另一台计算机上执行，则必须另编程序。由于机器语言使用的是针对特定型号计算机的语言，故其运算效率是所有语言中最高的。

2. 汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的助记符来替代特定指令的二进制串，如用 ADD 代表加法、用 MOV 代表数据传递等。这样，人们即可很容易读懂并理解程序的功能，便于纠错及维护，这种编程语言称为汇编语言，即第二代计算机语言。然而，计算机无法识别这些助记符，因此需要有一个专门的程序负责将这些符号翻译成二进制形式的机器语言，这类程序称为汇编程序。

汇编语言同样依赖于机器硬件，可移植性不好，但效率仍很高(与机器语言相同)。针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精炼且质量高，所以至今仍是一种强有力的常用编程语言。

汇编语言的实质与机器语言是一样的，都是直接对硬件进行操作，只不过指令采用了助记符，更容易识别和记忆，但它同样需要编程人员将每一步具体的操作以命令的形式写出来。

汇编语言程序的每一条指令只能对应实际操作过程中的一个细小的动作，如加减、移动、自增等。因此，汇编语言源程序一般冗长、复杂、容易出错，而且使用汇编语言编程需要具备更多的计算机专业知识。但其所能完成的操作不是一般高级语言能够实现的，而且源程序经汇编生成的可执行文件不仅比较小，而且执行速度快。

3. 高级语言

从最初与计算机交流的痛苦经历中，人们认识到应该设计一种既接近于数学语言或人类自然语言，又不依赖于计算机硬件的语言，且用该语言编出的程序能在所有机器上通用。经过计算机科学家们的持续努力，1956 年，世界上第一个完全脱离机器硬件的高级语言 Fortran 正式问世。半个多世纪以来，已有数百种高级语言出现，其中影响较大、使用较普遍的有Fortran、ALGOL、COBOL、Basic、LISP、SNOBOL、PL/1、Pascal、C、PROLOG、Ada、C++、VC、Python、Delphi、Java 等。

高级语言是绝大多数编程人员的选择。与汇编语言相比，它不仅将许多相关的机器指令合成为单条指令(语句)，还去掉了与具体操作有关但与完成工作无关的细节(如使用堆栈、寄存器等)，这大大简化了编程。同时，由于省略了很多细节，编程人员也无须具备太多专业知识。

1.2 认识 Python

Python 是一种简单易学、功能强大的编程语言，它继承了传统编译语言的强大性和通用性，具有高层次的数据结构，支持面向对象的编程方法。Python 凭借优雅的语法、动态的类型，连同它天然的解释性，成为大多数平台下多个领域快速应用开发的理想语言。

同样的功能，用 Python 只需很少的代码就可以实现，编写的程序清晰易懂、优雅美观，用“高效开发+简单易学”来形容它非常合适。

1.2.1　Python 的发展史

Python是一种面向对象的结构化编程语言，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 20 世纪 90 年代初设计。1989 年圣诞节期间，吉多为了打发圣诞节的无聊，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，是因为该单词取自英国 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)。后来，Python 逐渐发展为最受欢迎的程序设计语言之一。

1991 年，吉多编写的第一个 Python 编译器完成，这标志着 Python 语言的第一个版本正式诞生。

2000 年 10 月 16 日，Python 2.0 发布，新增了许多重要特性，包括内存回收机制和对 Unicode 编码的支持。Python 2.0 的发布奠定了现代 Python 语言框架的基础。此后近十年间，又陆续发布了 Python 2.4、Python 2.5 和 Python 2.6。

与此同时，随着 Python 自身功能的完善，各类基于 Python 的生态系统也逐渐应运而生。

2004 年，基于 Python 的 Django 框架开始应用于 Web 开发，该框架目前已成为最流行的 Web 开发框架之一。2010 年，另一个流行的轻量级 Web 开发框架 Flask 也正式诞生。此后，以豆瓣网、知乎和 Dropbox 等为代表的企业和机构，均使用 Python 进行网站开发，这预示着 Python 应用到 Web 开发领域逐渐成为新趋势。

2008 年 6 月 26 日，用 Python 编写的 Web 爬虫框架 Scrapy 发布，它降低了网络爬虫技术的使用门槛，让更多人能运用 Scrapy 框架从互联网获取数据。

2008 年发布的 Numpy、Scipy，以及 2009 年发布的 pandas，共同奠定了 Python 在数据分析与科学计算领域的地位。

此后，Python 在 Web 开发、网络爬虫、数据科学与数据分析、人工智能等应用方面逐渐崭露头角。

2008 年 12 月，Python 3.0 发布。

2010 年 7 月发布的 Python 2.7，是 Python 2.x 系列的最后一个版本，其主版本号为 2.7。

从版本发布时间来看，Python 3.0 早于 Python 2.7 发布，且这两个版本互不兼容、完全独立，因此 Python 官网提供了两个版本的下载。目前 Python 处于 Python 2.*和 Python 3.*共存的阶段，原因是仍存在大量基于 Python 2.*的开发人员和第三方库。

尽管 Python 3.0 发布于 Python 2.7 之前，但 Python 3.0 中的许多特性被反向移植到了 Python 2.7 中。相较于 Python 2.6，Python 2.7 有显著改进，集成了 Python 3.0 的大量特性和库，同时兼顾了 Python 2.x 的开发人群。

自 Python 3.0 发布后，Python 3.2、Python 3.3、Python 3.4、Python 3.5、Python 3.6、Python 3.7、Python 3.8、Python 3.9、Python 3.10、Python 3.11、Python 3.12 等版本相继推出。2024 年 10 月，Python 发布了目前的最高版本 3.13(各个版本的发布时间可参见官网：https://www.python.org/)。

1.2.2 Python 的特点

很多初学 Java 的人都会在 Classpath 的概念上感到困惑，甚至因 Classpath 配置错误，导致连最简单的“Hello World!”程序都无法运行。而用 Python 就不会出现此类问题，因为 Python 是一种解释型语言，同时也是一种脚本语言，写好代码即可直接运行，省去了编译、连接等一系列步骤。对于需要多动手实践的初学者而言，这减少了很多出错的机会。不仅如此，Python 还支持交互的操作方式，如果只是运行一段简单的小程序，可以省略编辑器，直接输入即可运行。

Python 是一种结构清晰的编程语言，使用缩进的方式来表示程序的嵌套关系，这可谓是一种创举。此举把过去软性的编程风格升级为硬性的语法规定，开发人员无须再在不同的编程风格之间进行选择。与 Perl 语言不同，Python 中没有各种隐晦的缩写，也不需要强记各种符号的含义。用 Python 书写的程序通俗读懂，这是很多开发人员的共识。虽然 Python 是一种面向对象的编程语言，但它的面向对象不像 C++那样强调概念，而是更注重实用。说到底，Python 不

会为了体现对概念的完整支持而把语言搞得很复杂，而是用最简单的方法让编程人员能够享受到面向对象带来的好处，这正是 Python 能够吸引众多支持者的原因之一。

Python 语言主要特点如下。

1) 易学易用

Python 是一门易于学习、使用且功能强大的编程语言。Python 语言用 C 语言开发，但摒弃了 C 语言中一些难以理解的语法(如指针)。它的语法结构简单，类似英语语言，不使用分号或花括号，采用文字排版中的缩进来定义代码块，且支持面向过程和面向对象的编程方法，简化或封装了面向对象中复杂的语法。总之，Python 坚持简单优雅，便于学习人员和开发人员使用。

2) 跨平台可移植性好

Python 语言是跨平台编程语言，支持 Windows、UNIX/Linux 到 Mac OS 等不同的操作系统。Python 语言也具有可移植性，例如，在 Windows 平台编写的 Python 代码，几乎无须修改即可直接在 Linux/UNIX 和 Mac OS 平台上运行，开发人员无须为不同机器编写不同代码。

3) 可扩展性强

由于 Python 语言本身用 C 语言开发，因而可使用 C 语言扩展以增加新功能。同时，它提供了丰富的应用程序编程接口(application programming interface，API)，既可以调用 C++、Java 等其他语言编写的模块，也可以嵌入其他语言开发的项目中。正因为如此，Python 语言也被称为“胶水语言”，能像胶水一样将不同的语言黏合在一起。

4) 拥有庞大的标准库和第三方库

Python 安装包中包含大量标准库，涉及范围广泛，从 C 语言编写的系统级模块，到 Python 编写的提供日常编程的模块，包括内置函数、数据类型、数字和数学模块、函数式编程模块、文件和目录访问、数据持久化、数据压缩和存档、文件格式、加密服务、调用操作系统服务、并发执行、网络与进程间通信、互联网数据处理及 Tkinter 图形库等。这些标准库已内置在 Python 语言中，无须单独安装。表 1.1 列出了部分比较常用的第三方库，也称为第三方模块。

表 1.1　Python 中常用的部分第三方库

第三方库名称	含义和用途
pip	包和依赖关系管理工具
pyinstaller	Python 脚本打包工具
Openpyxl	用于读写 Excel
pandas	数据分析工具包
matplotlib	2D/3D 类
numpy	科学计算的基础软件包
tkinter	GUI 图形库
wxpython	GUI 图形库
pyQt	GUI 图形库
requests	http 请求的模块
pillow	图像处理库

(续表)

第三方库名称	含义和用途
django	重量级 Web 服务器框架
Tornado	非阻塞式 Web 服务器框架
Beautiful Soup 解析库	XML 和 HTML 的解
tensorflow	深度学习框架
scikit-learn	机器学习工具包

5) 面向对象

面向对象的程序设计降低了结构化程序设计的复杂性，让程序设计更贴近现实生活。结构化程序设计将数据和逻辑混合在一起，不便于程序维护；而面向对象的程序设计会抽象出对象的行为和属性，将行为和属性分离开，又能合理地组织在一起。Python 语言具有很强的面向对象特性，且简化了面向对象的实现，它消除了保护类型、抽象类、接口等面向对象元素，让面向对象的概念更容易理解。

6) 解释性和(字节)编译性

Python 是一种解释性语言，这意味着开发过程中没有编译环节。一般来说，由于不是以本地机器码运行，纯粹的解释性语言通常比编译性语言运行得慢。但与 Java 类似，Python 实际上支持字节编译，能生成一种近似机器语言的中间形式。这不仅改善了 Python 的性能，还让它保留了解释性语言的优点。

7) 内置的数据结构

Python 提供了一些内置的数据结构，这些数据结构实现了类似 Java 中集合类的功能，包括元组、列表、字典等。内置的数据结构简化了程序设计，元组相当于“只读”的数组，列表可作为可变长度的数组使用，字典则相当于 Java 中的 HashTable 类型。

1.2.3 Python 的应用

Python 语言的应用范围十分广泛，几乎涵盖各个领域。图 1.2 显示了 Python 最主要的应用领域。

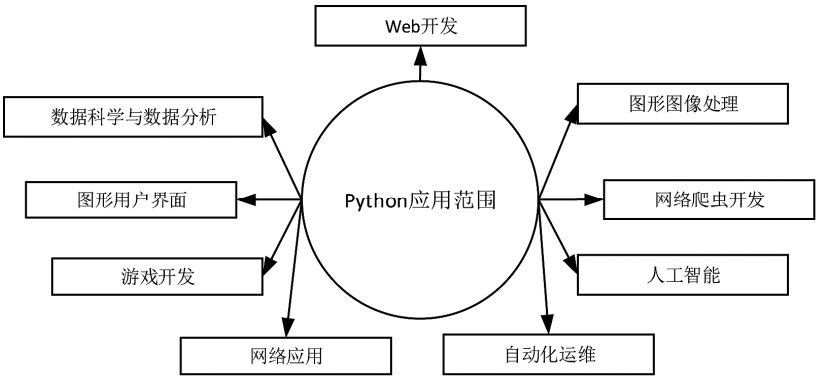

图 1.2 Python 应用领域

下面选择其中的五个应用领域加以介绍。

1) Web 开发

Python 在 Web 应用程序快速开发领域的应用日益广泛，它提供了多款适用于 Web 应用开发的框架和库。这些框架经过不断使用和完善，提供了更多的安全性、可扩展性和便捷性。

目前，知乎、豆瓣等知名应用均采用 Python 开发，而 Django、Flask 和 Tornado 则是当下比较流行的 Web 框架。

2) 网络爬虫开发

网络爬虫技术已成为自动获取、采集互联网数据的主要方式之一，作为数据分析和大数据最核心的数据来源，其在互联网行业中的地位越来越重要。Python 自带的 urllib 库、第三方的 requests 库和 Scrapy 框架，极大地降低了爬虫开发的难度，让爬虫开发变得简单高效。

3) 数据分析

数据分析指用适当的统计分析方法，对收集来的海量数据进行处理分析，通过汇总、理解与消化，最大限度地挖掘数据价值，发挥数据的作用。

数据分析的数学基础在20世纪早期就已确立，但直到计算机出现，才使其实际应用成为可能，并使得数据分析得以推广。在数据处理与分析环节，第三方库 Numpy 和 Scipy 提供了丰富的科学计算和数据分析功能，包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像图例、常微分方程求解、矩阵解析和概率分布等；在数据可视化环节，第三方库 Matplotlib 则能生成各类丰富的数据可视化图表。

4) 人工智能

人工智能(artificial intelligence)、机器学习(machine learning)和深度学习(deep learning)是当前最热门的技术话题。从宏观范畴来看，机器学习与深度学习均隶属于人工智能领域。

Python 始终伴随着人工智能的发展进程，积累了丰富的算法、第三方库和机器学习与深度学习框架。其中，第三方库 Numpy 常用于科学计算，特别是在机器学习和深度学习的矩阵运算中发挥重要作用。此外，TensorFlow、Caffe 等主流深度学习框架，其主体开发语言为 Python，提供的原生接口也是面向 Python 的。

5) 自动化运维

随着技术的不断进步和业务需求的快速增长，一名运维人员通常要管理成百上千台服务器，运维工作也因此变得重复且繁杂。Python 作为运维工程师首选的编程语言，借助自动化运维技术，能够将运维人员从复杂的服务器管理工作中解放出来，使运维工作变得简单、快速和准确。

在众多操作系统中，Python 是标准的系统组件，如大多数 Linux 发行版和 Mac OS 都集成 Python，可在终端中直接运行 Python 程序。Python 标准库包含多个可调用操作系统功能的库，通过第三方软件包 Pywin32，Python 能够访问 Windows 的 COM 服务及其他 Windows API；借助 IronPython，Python 程序能够直接调用.NET Framework。

总体而言，Python 编写的系统管理脚本，在可读性、代码可重用性和扩展性等方面，均优于普通的 Shell 脚本。

1.2.4　Python 常用解释器

当 Python 安装在计算机上后，就具备了运行 Python 程序的环境。该运行环境主要由解释器(interpreter)和众多支持库组成。初学者需先了解 Python 和 Python 解释器之间的关系。Python

是一门语言规范，类似于英语语法，人们通过写英语、说英语，才能将书本上的语法变成能交流的语言，这相当于语法的应用和实现。Python 解释器就类似于人们写英语、说英语的过程，是 Python 语言规范的具体实现。Python 解释器有多种实现，如使用 C 语言编写的 CPython、使用 Java 语言编写的 Jython、使用 Python 语言编写的 PyPy 等。其中，CPython 解释器是 Python 官方的发行版和标准实现，也是目前默认且应用最广泛的实现。平时所说的“Python”，大多指的就是 CPython。

对于大多数 Python 程序员而言，Python 解释器就像一个黑匣子——他们只需使用解释器运行程序并获取结果。然而，若能进一步了解 Python 解释器，便能更好地掌握 Python 的运行机制，明晰 Python 的工作原理。图 1.3 描述了 Python 解释器运行的机制。

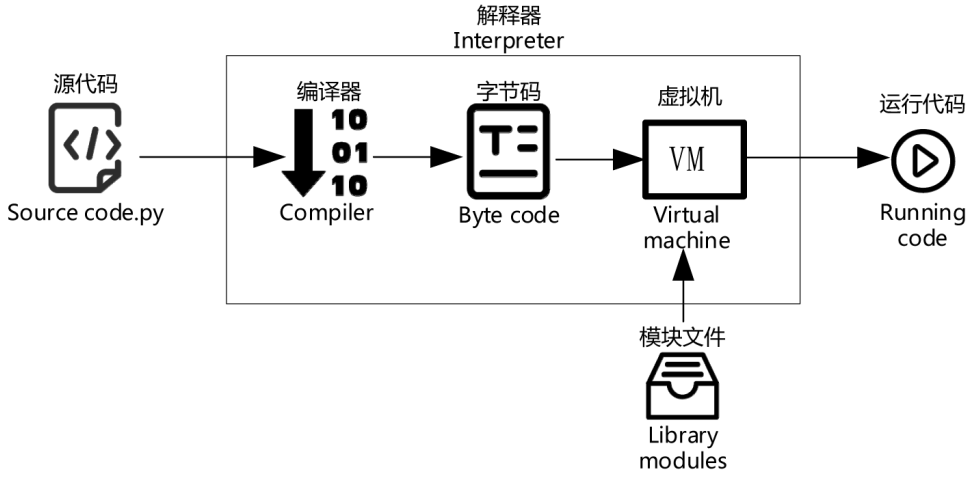

图 1.3　Python 解释器运行的机制

解释器由两部分组成：编译器(compiler)和虚拟机(virtual machine)。

编译器首先将源代码(.py 文件)编译成一种形式更简单的字节码，该字节码通常存储在.pyc 文件中。当源代码更新或在其他必要情况下，会重新生成字节码。为了将程序分发给已经安装了 Python 的用户，可以发布.py 文件或.pyc 文件。

字节码面向底层，与平台无关，每个源语句都会被翻译成一组字节码指令。从本质上看，字节码类似于 CPU 指令，但它们不是由 CPU 直接执行，而是由一个称为 Python 虚拟机(Python virtual machine，PVM)的软件来执行。之所以将源代码转换为字节码，主要是为了实现平台无关性，并加快执行速度，因为字节码比源代码语句运行得更快。

Python 虚拟机(PVM)实际上是模拟真实计算机的软件，它是安装在机器上的 Python 系统的一部分，也是真正运行脚本的组件，堪称 Python 的运行引擎。PVM 以迭代的方式执行字节码指令，通过一个大循环逐个完成操作。

源代码一旦转化为字节码，就会被输入 PVM，PVM 会一个接一个地遍历字节码指令，以执行它们的操作。每次运行解释程序时，解释器都必须将源代码转化为机器码，并引入运行库。这种转化过程使程序的运行速度比用编译型语言编写的同类程序要慢。

以上就是 Python 源代码到机器码的整个运行过程。具体而言，Python 源代码首先会被编译成字节码，然后由 PVM 对字节码进行解释，将其转化为机器码，最终得出输出结果。

这里需要说明的是，Python 虽为解释型语言，但在运行过程中也存在编译环节，只不过这种编译与其他编译型语言有很大区别。首先，Python 解释器中的编译对用户是隐藏的，由 Python 自动完成。对用户来说，无须手动操作编译器来编译源代码，只需在交互式提示符>>>或集成开发环境(integration development environment，IDE)中运行程序即可。而其中重要的是，PVM 负责对字节码进行解释执行，没有编译型语言那样的手工编译步骤，这正是 Python 被归类为解释型语言的原因。

1.3　Python 开发环境搭建

Python 可在多种平台上运行，但考虑到目前应用较多的是 Windows 平台和 Linux 平台，故本节分别介绍这两种平台下 Python 开发环境的搭建方法。

1.3.1　Windows 环境中 Python 的安装

1) 下载 Python 安装包

首先登录 Python 官方网站(www.python.org/downloads/)，然后单击 Downloads 按钮，即可选择适配自身软、硬件环境的安装包并下载，如图 1.4、图 1.5 和图 1.6 所示。

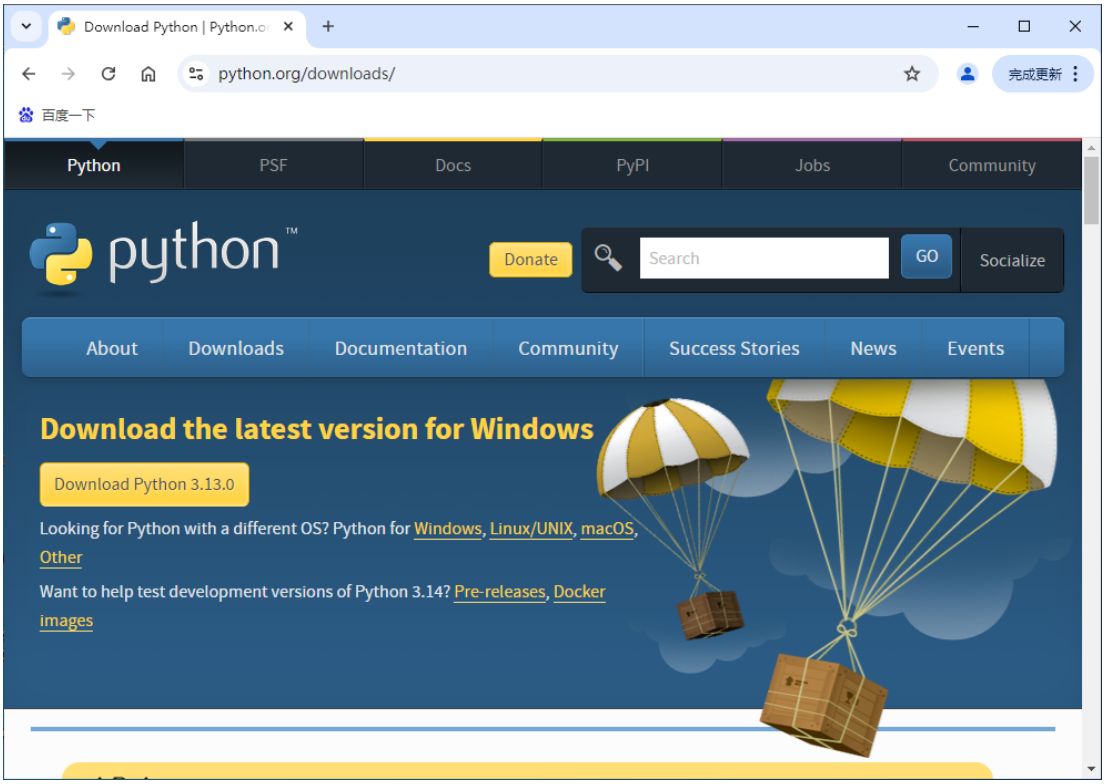

图 1.4　Python 官网界面

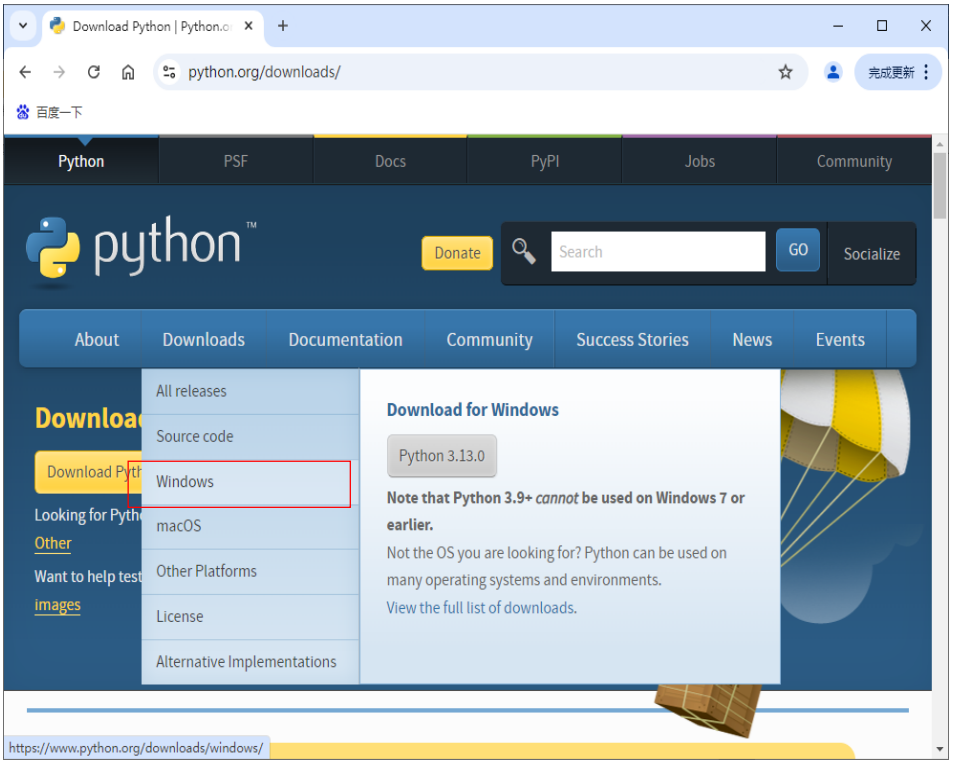

图 1.5　选择 Windows

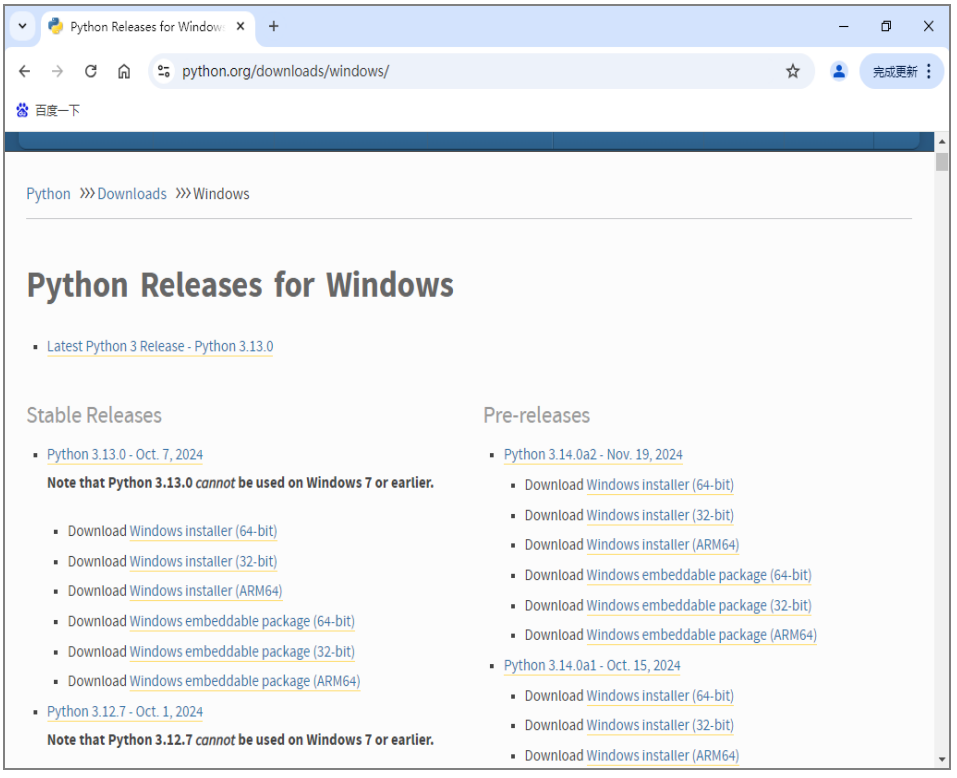

图 1.6　选择合适版本

此处建议不使用最新版本，本节将以下载 Python 3.9.7 64 位离线安装包为例，演示 Python 解释器的安装方法，如图 1.7 所示。

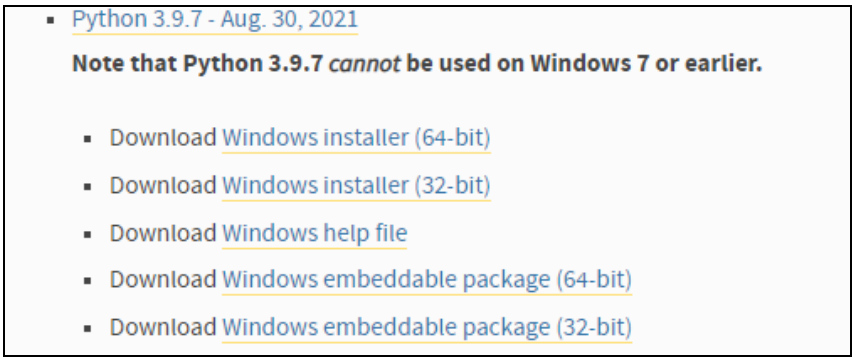

图 1.7　选择下载 Python 3.9.7 64 位安装包

2) 安装 Python

下载完成后，双击 Python 安装包启动安装流程。保持默认配置，选择好 Python 的安装路径后，单击 Install Now 按钮开始安装，如图 1.8 所示。安装成功后，会显示 Setup was successful，如图 1.9 所示。Python 安装完成后，需将其安装目录添加到系统的 PATH 环境变量中。不过高版本安装包无须手动配置，只要在安装时选中 Add Python 3.9 to PATH 复选框，即可在安装过程中自动完成环境变量配置。

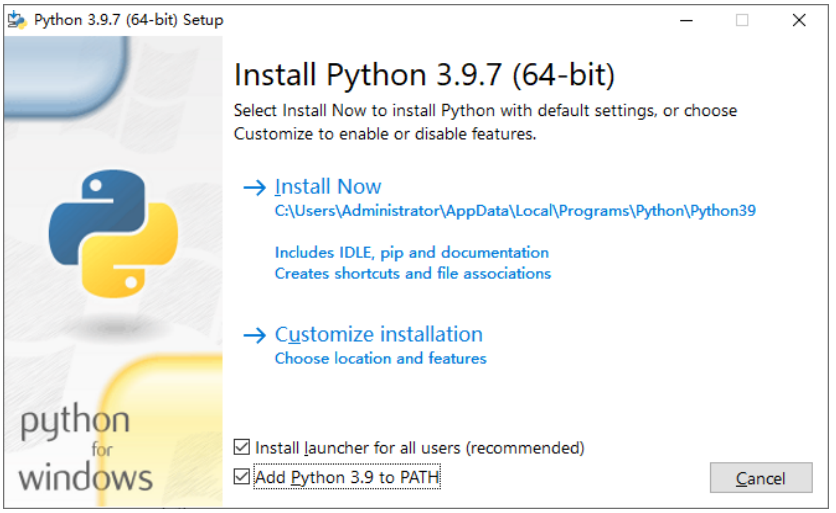

图 1.8　开始安装

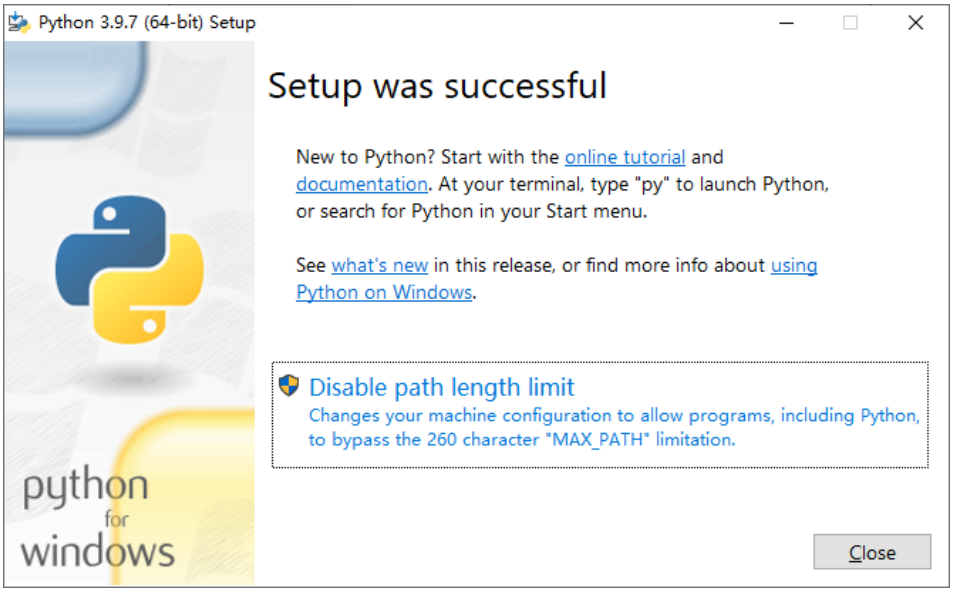

图 1.9　安装完成

3) 在 cmd 中输入命令验证安装结果

按 Win+R 组合键打开“运行”窗口，输入 cmd 后按回车键，进入 Windows 命令提示符窗口。在窗口中输入 python，若能进入 Python 3.9.7 的命令行模式，如图 1.10 所示，则说明 Python 3.9.7 安装成功。

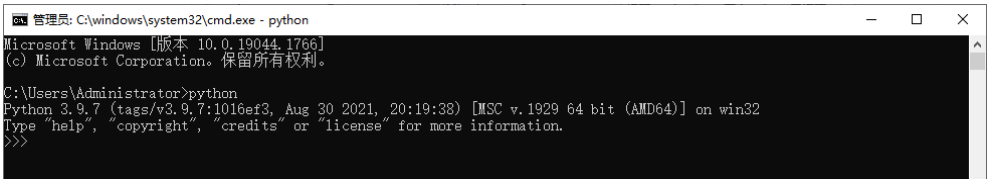

图 1.10　测试安装结果

4) 演示 Python 命令

在 Python 命令行模式中输入命令 print ("hello world")，可看到运行结果为 hello world，如图 1.11 所示。

图 1.11　在命令提示窗口执行 Python 命令

5) 集成开发环境的使用

通过以上测试可知，Python 环境已安装完毕。那么该如何开展软件开发？是否要采用命令行方式？答案当然是否定的，实际上我们可以借助各类 IDE 进行开发工作。

在 Windows 下安装 Python 时，会默认同步安装其自带的集成开发环境 IDLE，可通过执行“开始”→“所有程序”→Python 3.9→IDLE 命令启动，如图 1.12 所示。

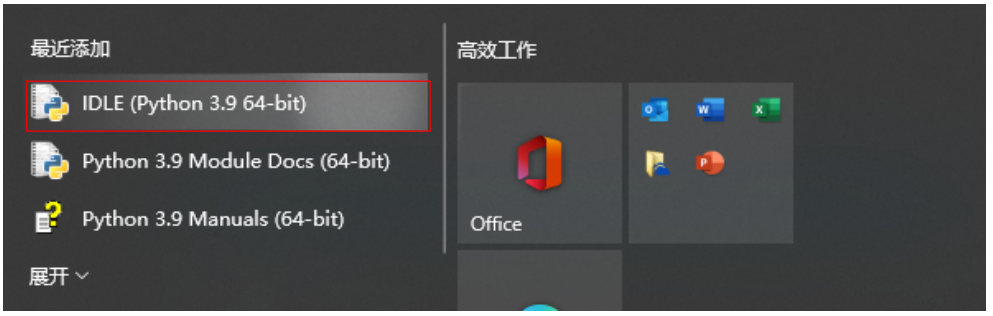

图 1.12　打开集成开发环境 IDLE

启动后的界面如图 1.13 所示。

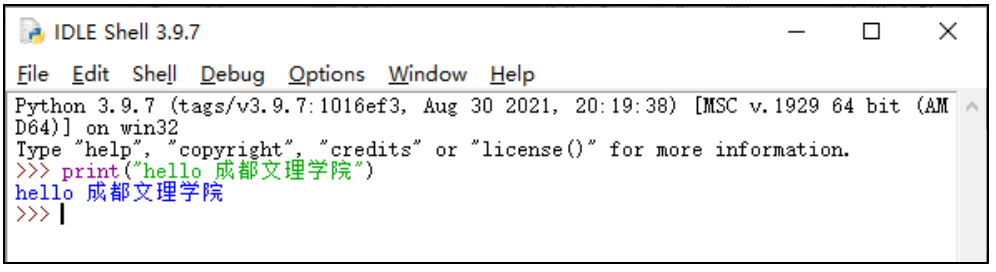

图 1.13　Python 的集成开发环境 IDLE 界面

1.3.2　Linux 环境中 Python 的安装

通常情况下，Ubuntu、Cent OS 这类常见的 Linux 系统，在安装时已默认预装了 Python。在 Linux 系统中，IDLE 被称为 Python 解释器，可在终端模拟器中输入“python”命令来启动它。Python 编程的入门操作多从 IDLE 编辑器开始，初学者熟悉基础后，可根据喜好选择更多 Python 编辑器，如专业级 Python 编辑器 Wing IDE。

遗憾的是，早期版本的 Linux 操作系统中，内置的 Python 多为 2.x 版本。若需在 3.x 版本环境下开发软件，则必须另行安装 Python 3.x 版本。现以Ubuntu(一款以桌面应用为核心的 Linux 操作系统)为例，简要介绍 Linux 平台下 Python 3 开发环境的搭建过程。

1) 安装 Python 3

Ubuntu 系统默认安装 Python 2，以 Ubuntu 16.04 为例，其预装的就是 Python 2.7。若需在 Python 3 环境下开发软件，则必须另行安装 Python 3。需要注意的是，Ubuntu 系统内许多底层功能依赖 Python 2 运行，因此安装 Python 3 时不可卸载 Python 2。

首先依次执行以下命令：

```
sudo cp/usr/bin/python/usr/bin/python_bak
sudo rm/usr/bin/python
sudo ln -s/usr/bin/python3.9/usr/bin/python
```

命令执行完成后，输入“python”并按回车键，检查版本是否为 Python 3.9，若版本匹配，则说明安装配置成功。

2) 安装 Sublime Text 3

进行 Python 开发前，选择一款合适的编辑器十分重要。Sublime Text 3(ST3)是一款轻量级、跨平台的文本编辑器，可安装在 Ubuntu、Windows 和 Mac OS X 等操作系统上。尽管它采用专有许可证，但用户可以免费使用该程序。如果想拥有更高级的版本，则可付费获取。

打开终端，依次输入下列命令完成安装：

```
sudo add-apt-repository ppa:webupd8team/sublime-text-3
sudo apt-get update
sudo apt-get install sublime-text-installer
```

若需卸载 sublime text，可执行命令：

```
sudo apt-get remove sublime-text-installer
```

安装完成后，桌面可能不会直接显示 Sublime Text 3 的图标，此时可在终端中输入“subl”启动软件，启动后，将其锁定到侧边栏，后续即可快速打开。

3) 配置 Sublime Text 3

为了借助各类插件扩展 Sublime 的功能，需先安装 Package Control 插件管理器(该工具需手动安装)，安装完成后，即可通过它实现 ST3 插件的安装、移除或升级操作。

按 Ctrl +～组合键打开 ST3 的控制台，在控制台中输入以下代码：

```
import urllib.request,os;pf='Package Control.sublime-package';ipp= sublime.installed_packages_path();urllib.request.install_opener(urllib.request.build_opener(urllib.request.ProxyHandler()));open(os.path.join(ipp,pf),'wb').write(urllib.request.urlopen('http://sublime.wbond.net/'+pf.replace(' ','%20')).read())
```

输入完以后，按回车键即可执行。

后续安装其他插件时，按 cmd+Shift+P 组合键打开 Package Control，输入 install，屏幕会显示 Package Control: Install Package 选项，如图 1.14 所示，然后在搜索框中输入要安装的插件名后，单击安装即可。

选择 Package Control:Install Package 选项后，会出现一个文本框，如图 1.15 所示。

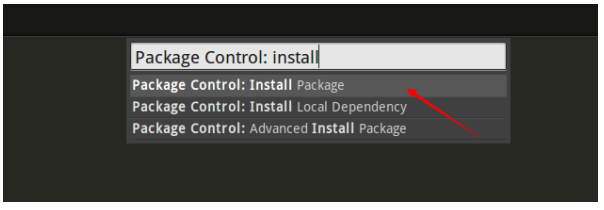

图 1.14　插件的安装

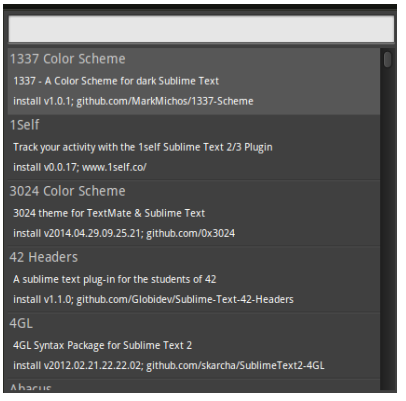

图 1.15　Python 插件的安装

在其中输入 Anaconda 后按回车键，稍等片刻便会出现如图 1.16 所示的界面，这表明 Anaconda 已安装完成。

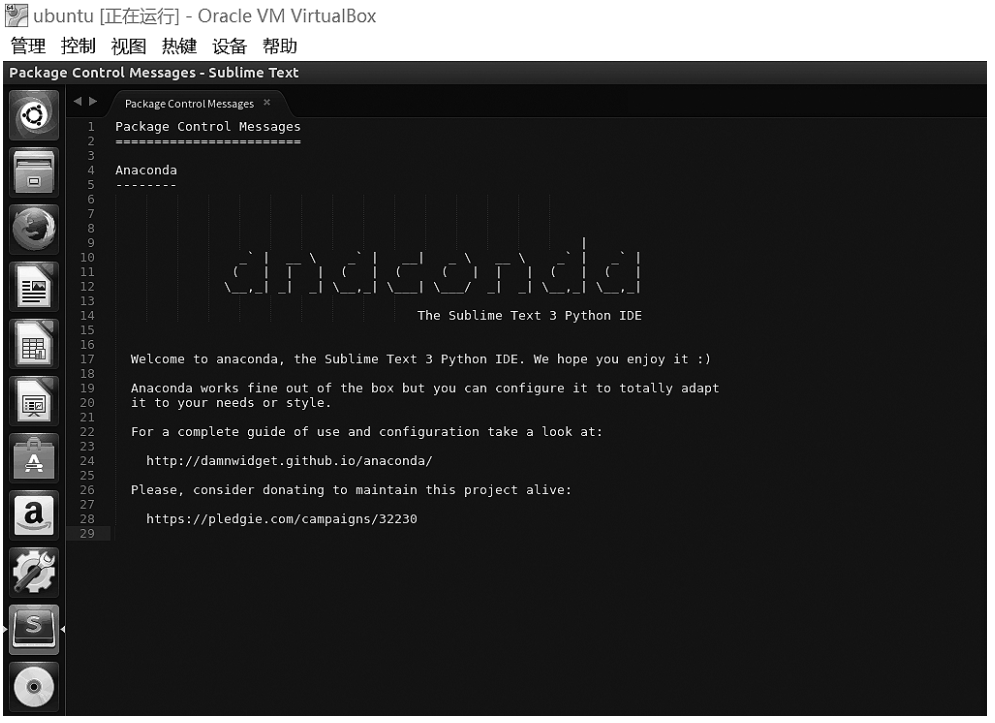

图 1.16　Python 插件安装完成界面

1.3.3　Python 常用的开发工具

除了 IDLE，Python 常用的集成开发环境还有以下几个。

1) PyCharm

PyCharm 是 JetBrains 公司开发的一款跨平台 Python 开发工具，具备代码自动完成、集成 Python 调试器、括号自动匹配及代码折叠等功能。它支持 Windows、Mac OS 及 Linux 等系统，还可实现远程开发、调试和运行。

PyCharm 有以下三个版本。

- PyCharm Professional(专业版)：付费版本，几乎拥有开发桌面、网络和 Web 等程序与系统的所有功能，特别适合项目开发，支持基于 Python 的第三方库(如 Django 和 Flask 等 Web 开发框架)，同时兼容数据库与科学计算工具。
- PyCharm Community(社区版)：免费版本，适合个人或小团队使用，可用于开发桌面和网络程序，但不具备专业版中 Web 开发、Python Web 框架支持、远程开发能力及数据库与 SQL 支持等功能。
- PyCharm Edu(教育版)：免费版本，专为教育场景设计，是面向学生和教师的教学工具，集成了 Python 课程学习平台。教师可借助该版本开展教学活动，学生也能利用它完成作业，满足教学与学习的双重需求。

2) Sublime Text

Sublime Text 是一款适合 Python 开发的工具，虽然它只是一个编辑器，但拥有丰富的插件，对 Python 开发的支持非常到位。

3) Anaconda 与 Jupyter Notebook、Spyder

Anaconda 是一个开源的 Python 发行版本，包含大量用于科学计算的第三方库及其依赖包。Jupyter Notebook 和 Spyder 是非常强大的交互式 Python 开发环境，提供高级的代码编辑、交互测试、调试等特性，支持 Windows、Linux 和 Mac OS X 等系统。Jupyter Notebook 和 Spyder 是 Anaconda 默认的开发工具，也是进行数据分析与机器学习较好的编辑器，特别是 Jupyter Notebook 与 Anaconda 的结合，被认为是进行数据分析的良好基础平台。

4) Eclipse with Pydev

Eclipse 是一款基于 Java 的集成开发环境，历史悠久且广受欢迎。它通过插件机制实现与多种语言无缝集成，支持多样化的编程需求。在 Eclipse 中安装 Pydev 插件后，即可像 PyCharm 一样进行 Python 开发，同时还能支持 Web 开发任务。

5) Wing

Wing 是一款专门为 Python 语言设计的 IDE，启动和运行速度都非常快，支持 Windows、Linux 和 Mac OS X 等系统。

由于.py 的源文件实际上就是一个文本文件，因此可以使用任何文本编辑器编写和打开.py 的源代码文件。

1.3.4 编写 Python 简单程序

通过对前面内容的学习，我们来编写一个 Python 小程序：求五个连续整数 11、12、13、14、15 的和并输出结果。

1) 直接在命令提示窗口中实现

执行“Win+R”命令，在“运行”对话框的“打开”文本框中输入“cmd”，打开命令提示窗口，输入“python”，如图 1.17 和图 1.18 所示。

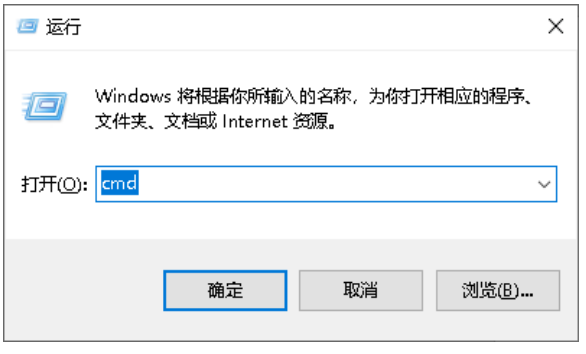

图 1.17 打开“运行”对话框

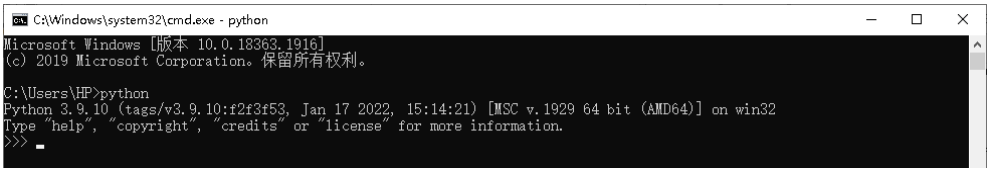

图 1.18　输入“python”命令

在提示符“>>>”后继续输入命令“print("11+12+13+14+15=",11+12+13+14+15)”后，按回车键，即可得到输出结果，如图 1.19 所示。

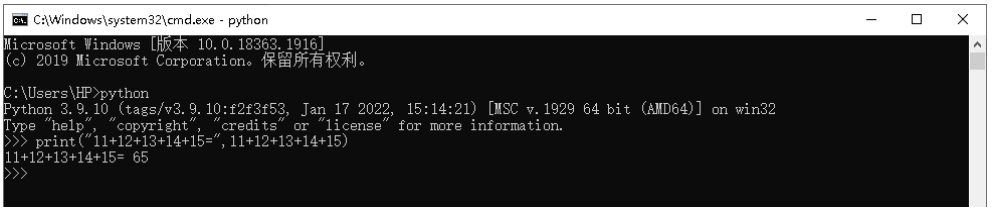

图 1.19　在命令提示窗口中输出结果

2) 使用 Python 程序文件实现

由上可以看出，只用单一的语句就能够在屏幕上输出所需的内容。这种在提示符下进行交互式编程非常方便直观，但都是一次性的，不能保存。相比之下，用户可能更需要编写 Python 程序来实现特定的功能和满足不同的应用需求，同时也方便代码的不断完善和重复利用，毕竟直接使用交互模式不是很方便。而 IDLE 不仅可以进行这种交互式编程，还可以编写、保存和运行 Python 源代码文件(Python 源代码文件的后缀名是.py)。此外，也可以在资源管理器中双击扩展名为.py 的 Python 程序文件来运行源代码文件。源代码编辑窗口如图 1.20 所示。

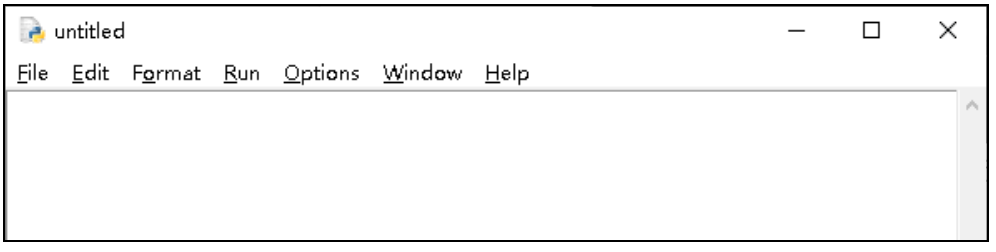

图 1.20　源代码编辑窗口

编辑完成后，我们将程序文件命名为“成都文理学院.py”，程序内容如图 1.21 所示。

图 1.21　编辑和保存程序的窗口

单击 Run(运行)菜单，在出现的下拉菜单中单击 Run Module(运行模块)子菜单或按 F5 键，运行该程序文件后，结果如图 1.22 所示。

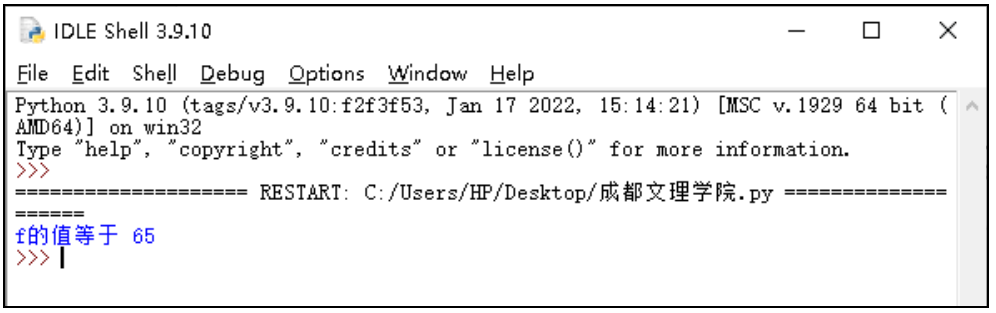

图 1.22　IDLE 中的运行结果

在某些情况下，还可能需要在 DOS 命令提示符环境下运行 Python 程序文件。可通过执行“Win+R”命令打开命令提示窗口后，执行 Python 程序。在 DOS 命令提示窗口中运行该程序后，结果如图 1.23 所示。

图 1.23　DOS 命令提示窗口中的运行结果

1.4　本章小结

本章首先对计算机基础知识做了简单介绍和梳理，重点讲解了计算机基本结构和计算机语言的相关内容。接着，对 Python 语言的发展、特点及应用等方面进行了简要介绍。目前，Python 主要存在两个版本体系：Python 2.*与 Python 3.*，其中 Python 3.*并不向下兼容 Python 2.*。Python 作为一种高级程序设计语言，具备易用易学、跨平台移植性强、可扩展性强等特点，同时拥有庞大的标准库和丰富的第三方库资源，因而在人工智能、Web 开发、网络爬虫、科学计算和数据分析等很多领域都有广泛的应用。

其次，本章介绍了搭建 Python 开发环境的方法。Python 语言属于解释型语言，因此运行 Python 程序时，需使用特定的解释器对其进行解释和执行。同时，本章还讲解了 Python 自带的 IDLE 使用方法，以及常用的第三方开发工具。Python 提供了交互模式用于交互式编程，该模式包含命令行窗口交互模式和基于 IDLE 的 Python Shell 模式。

随后，本章又介绍了编写第一个 Python 程序的两种方法。其中，搭建 Python 开发环境和使用自带的 IDLE 是本章的学习重点。希望读者在学习完本章后，能够成功搭建起学习所需的开发环境，完成第一个 Python 程序，从而迈出 Python 开发的第一步。

第 2 章

Python 语言基础

本章重点内容

- 简单数据类型；
- Python 语法基础；
- 标识符与关键字；
- Python 中的计算；
- 基本输入与输出操作。

2.1 简单数据类型

Python 语言提供了非常丰富的数据类型。Python 3 中，有 6 种标准的数据类型，分别是数字(Numeric)、布尔值(Boolean)、字符串(String)、序列(Sequence)、字典(Dictionary)和集合(Set)。其中，数字类型又分为整数(Integer)、浮点数(Float)和复数(Complex)三种；序列类型又分为列表(List)和元组(Tuple)两种。Python 的数据类型如图 2.1 所示。

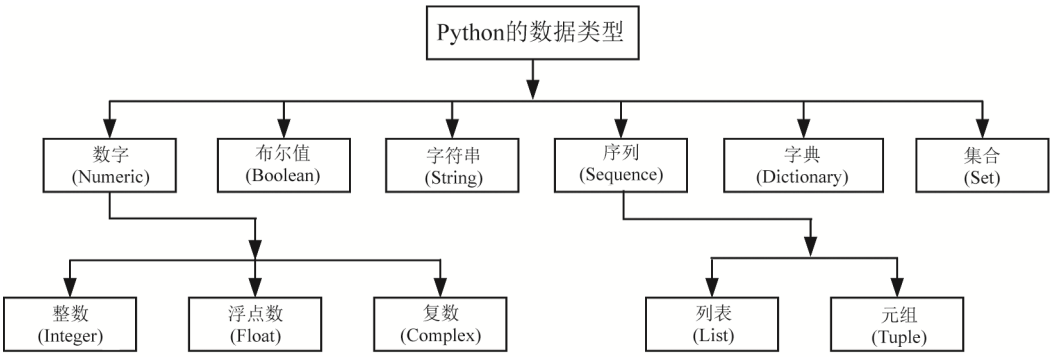

图 2.1　Python 的数据类型

除了以上数据类型，Python 还支持一种特殊的数据类型，即空类型(NoneType)。该类型的对象只有一个 None，用于定义空变量或对象，类似于其他语言中的 null 空值。通常，可以将 None 赋值给任何变量，但不能创建 NoneType 对象。

本章主要介绍较为简单的数字类型和布尔类型。其中，数字类型主要用于存储数值，它支持三种不同的数值类型，分别是整数、浮点数和复数。关于字符串、列表、元组、字典和集合等数据类型，将在第 4 章和第 6 章进行详细讲解。

2.1.1 整数类型

整数类型用来表示整数数值，常指不带小数点的数。在 Python 语言中，整数包括正整数、负整数和 0，并且它的位数是任意的(当超过计算机自身的计算功能时，会自动转用高精度计算)，如果要指定一个非常大的整数，只需要写出其所有位数即可。整数类型包括十进制整数、八进制整数、十六进制整数和二进制整数，如 0、100、-680、-018、-0x233、0x85。整数类型没有大小限制，仅受限于可用内存的大小。

1. 十进制整数

十进制整数由数字 0～9 组成，进位规则为“逢十进一”。在 IDLE 窗口中输入各种十进制整数，结果如图 2.2 所示。在 Python 2.x 中，如果输入的数比较大，Python 会自动在其后面加上字母 L(也可能是小写字母 l)。

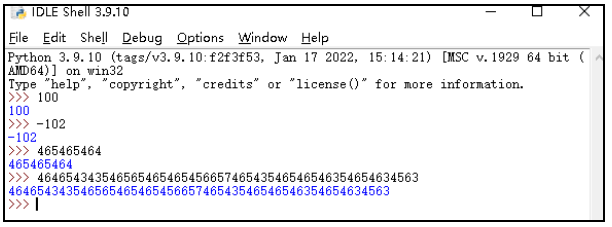

图 2.2 在 IDLE 窗口中输入十进制整数

2. 八进制整数

八进制整数由数字 0～7 组成，进位规则为“逢八进一”，并且是以 0o/0O 开头的数，如 0o77(转换为十进制数后为 63)、-0o77(转换为十进制数后为-63)。在 IDLE 窗口中输入各种八进制整数，结果如图 2.3 所示。在 Python 3.x 中，八进制整数必须以 0o/0O 开头，但在 Python 2.x 中，八进制整数可以 0 开头。

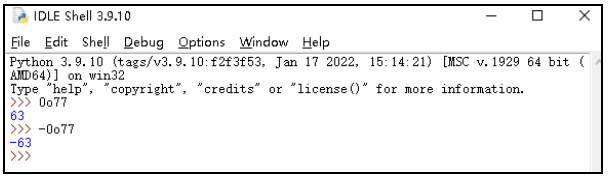

图 2.3 在 IDLE 窗口中输入八进制整数

3. 十六进制整数

十六进制整数由数字 0～9、字母 A(a)～F(f)组成，进位规则为“逢十六进一”，并且是以 0x/0X 开头的数，如 0xAB(转换为十进制数后为 171)、0X525(转换为十进制数后为 1317)。在 IDLE 窗口中输入各种十六进制整数，结果如图 2.4 所示。

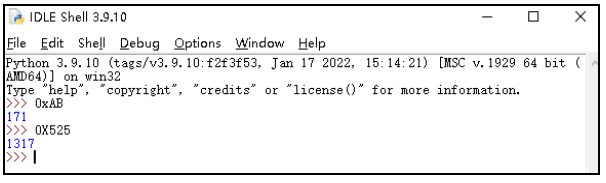

图 2.4 在 IDLE 窗口中输入十六进制整数

4. 二进制整数

二进制整数由数字 0 和 1 组成，进位规则是“逢二进一”，并且是以 0b/0B 开头的数，如 0b1010(转换为十进制数后为 10)、0B1111(转换为十进制数后为 15)。在 IDLE 窗口中输入各种二进制整数，结果如图 2.5 所示。

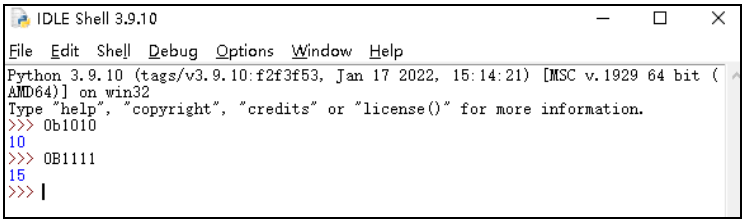

图 2.5　在 IDLE 窗口中输入二进制整数

2.1.2　浮点数

浮点数由整数和小数点组成，主要用于表示和处理包含小数的数值，如 0.1、15.8、−5.9、−3.5e10、1.32e18。浮点数可以用科学计数法表示，例如，2.5e2 = 2.5 × 10^2 = 250。在 IDLE 窗口中输入浮点数，结果如图 2.6 所示。

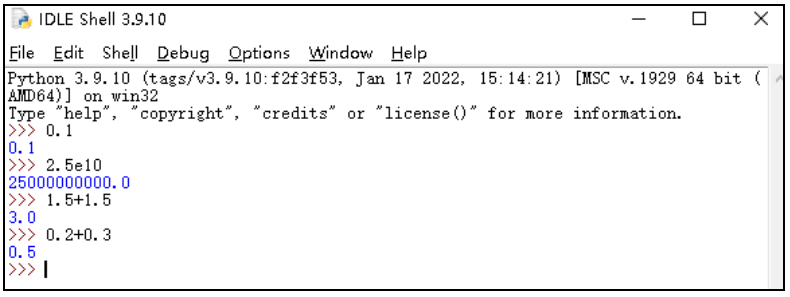

图 2.6　在 IDLE 窗口中输入浮点数

2.1.3　复数

在Python中，复数的形式与数学中的复数完全一致，它由实数和虚数两部分构成，虚部用 j 或 J 表示，既可以写成 a + bj 的形式，也可以用 complex(a,b)表示。当表示一个复数时，可以将其实部和虚部相加，例如，一个复数的实部为 3.5，虚部为 3.5j，则这个复数为 3.5+3.5j。在 IDLE 窗口中输入复数，结果如图 2.7 所示。

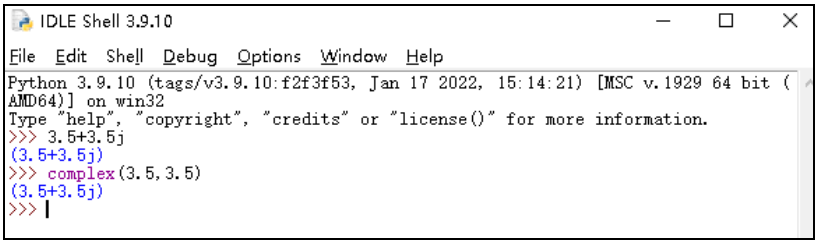

图 2.7　在 IDLE 窗口中输入复数

2.1.4 布尔型

布尔类型主要用来表示真值和假值。在 Python 中，标识符 True 和 False 被视为布尔值。严格来讲，布尔类型实际上是整数类型的子类型，布尔值包含两个常量对象True和False，它们用于表示逻辑上的真与假。当布尔类型用于算术运算符的操作数时，True 和 False 分别相当于整数 1 和 0。需要特别注意的是，Python 中真和假的取值范围很广，下面基本完整地列出了会被视为假值的内置对象。

- False 或 None。
- 数值中的零，包括 0、0.0 和虚数 0。
- 空序列，包括字符串、空元组、空列表、空字典。
- 自定义对象的实例，该对象的_bool_方法返回 False 或_len_方法返回 0。

除了上面这些，其他的值都返回 True。也就是说，在 Python 中，逻辑真包含常量 True、非 None、非零值、非空字符串、非空的列表与元组、非空集合等。通常，我们可以使用 bool()函数来测试值的逻辑真与假。这里以 None、数字 0 和一些非 0 数字为例进行说明，在 IDLE 窗口中使用 bool()函数测试的结果如图 2.8 所示。

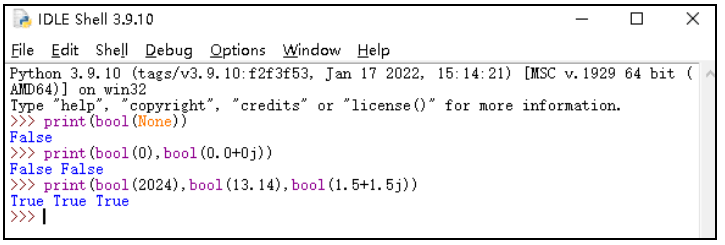

图 2.8 在 IDLE 窗口中使用 bool()函数测试

2.1.5 数字类型转换

数字类型支持类型转换。类型转换是指将一种数据类型(整数、布尔值、浮点数等)的值转换为另一种数据类型的过程。Python 的类型转换分为隐式类型转换(Implicit Type Conversion)和显式类型转换(Explicit Type Conversion)两种。

1. 隐式类型转换

隐式类型转换是指 Python 自动将一种数据类型转换为另一种数据类型，该过程不需要用户参与。下面以整数类型转换为浮点型为例，运行结果如图 2.9 所示。

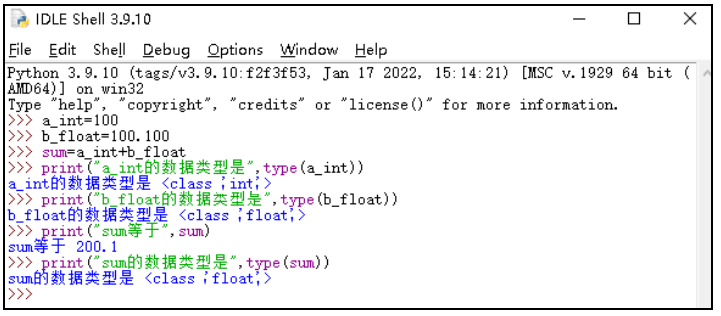

图 2.9 隐式类型转换运行结果

2. 显式类型转换

显式类型转换是指用户主动将对象的数据类型转换为所需的数据类型，它一般通过 Python 提供的函数来实现。常用的显式类型转换函数如表 2.1 所示。

表 2.1　常用的显式类型转换函数

函数	作用
int(x)	将 x 转换为整数类型
float(x)	将 x 转换为浮点数类型
complex(real,[imag])	创建一个复数
str(x)	将 x 转换为字符串
repr(x)	将 x 转换为表达式字符串
eval(str)	计算在字符串中的有效 Python 表达式，并返回一个对象
chr(x)	将整数 x 转换为一个字符
ord(x)	将一个字符 x 转换为它对应的整数值
hex(x)	将一个整数 x 转换为一个十六进制字符串
oct(x)	将一个整数 x 转换为一个八进制字符串

这里以 int(x)和 float(x)函数为例进行显示类型转换，转换结果如图 2.10 所示。

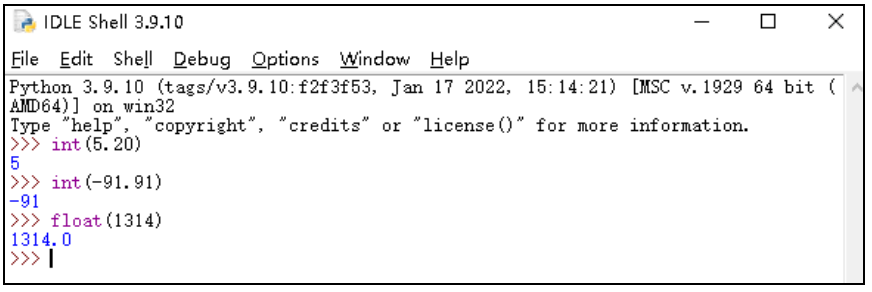

图 2.10　显式类型转换运行结果

2.1.6　变量与常量

程序的功能就是将数据存储在计算机中，通过变量来引用和访问数据，并根据不同需求对数据进行相应的处理与操作。变量是存储在计算机存储器中数据的名称。在 Python 中，变量通过赋值语句创建，且在程序运行期间其值可改变，因此称为变量。此外，Python 中还存在常量，与变量不同，常量是指在内存中用于保存固定值的单元，在程序中其值不可改变。

1. 变量

变量即值可以改变的量，只不过在程序中，变量不仅可以引用数字，还可以引用字符串，甚至可以引用列表、元组、字典、集合等更高级的数据结构。例如，在如图 2.11 所示的例子中，a 就是一个变量，且为它赋值为 100。

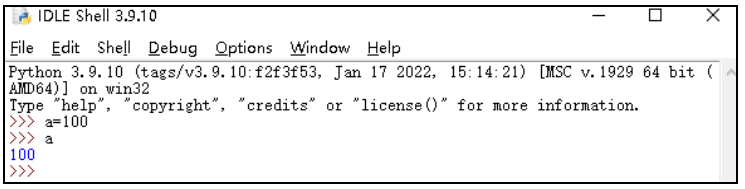

图 2.11　创建变量并赋值

由于 Python 是动态语言，不需要像 C 语言那样提前声明变量的数据类型。例如，C 语言使用变量前需先声明：

```
int a=100;
```

而 Python 直接使用 a=100 即可。这正是动态语言的优势：无须关注变量本身的数据类型，Python 会自行判断。

当多次给同一个变量赋值时，Python 只会记住最后一次所赋的值，在 IDLE 窗口中操作的结果如图 2.12 所示。

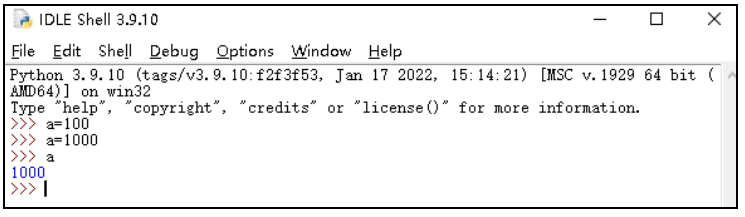

图 2.12　同一变量多次赋值结果

Python 允许同时为多个变量分配同一个值，也支持在同一行给多个变量同时赋不同的值，在 IDLE 窗口中测试的结果如图 2.13 和图 2.14 所示。

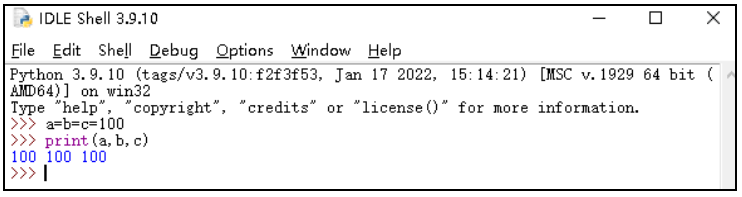

图 2.13　多变量同时赋相同的值

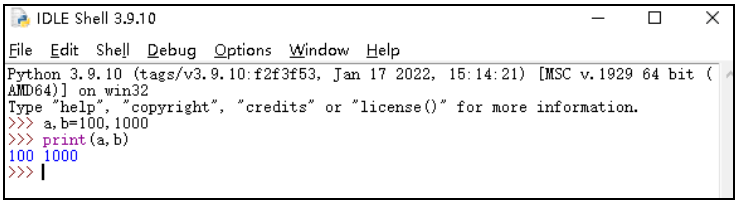

图 2.14　多变量同时赋不同的值

2. 常量

常量与变量相对应，是程序运行过程中值不可改变的量。例如：

```
PI=3.1415926535897
```

这时就认定 PI 是一个常量，也就是说，PI 始终代表 3.1415926535897。实际上，Python 中并没有严格意义上的常量，因为 Python 没有确保常量值不被改变的机制。通常会用全部大写的字母来表示常量，尽管我们将 PI 当作常量，但实际上仍可以为其重新赋值。

2.1.7　运算符与优先级

计算机最根本的用途是处理数据和完成各类运算任务。Python 提供了丰富多样的运算符，这些特殊符号主要用于数学计算、比较大小和逻辑运算等场景。Python 的运算符主要包括算术运算符、赋值运算符、比较(关系)运算符、逻辑运算符和位运算符。使用运算符将不同类型的数据按一定规则连接起来的式子，称为表达式。由一个或多个表达式组成的语句，可满足各种运算需求。例如，用算术运算符连接的式子称为算术表达式，用逻辑运算符连接的式子称为逻辑表达式。表达式是运算符、值和变量等的组合，如 10 + 20、a + b 和 a + b*c -a + d/b 都是表达式，且都有返回值。实际上，单个值本身可视为表达式，变量同样也是表达式。表达式最显著的特点是必须返回一个值，在交互式模式下能直接看到该返回值。

1. 算术运算符

算术运算符是处理四则运算的符号，可以实现加、减、乘、除、求余等数学运算。Python 中的算术运算符及其作用如表 2.2 所示。

表 2.2　Python 中的算术运算符及其作用

算术运算符	作用	说明
+	加法运算	a + b，返回 a、b 的和
-	减法运算	a - b，返回 a、b 的差
*	乘法运算	a * b，返回 a、b 的积
/	除法运算	a / b，返回 a、b 的商
%	求余运算	a % b，返回 a 除以 b 的余数
//	整除运算	a//b，返回 a、b 的商(整数)
**	幂运算	a**b，返回 a 的 b 次幂

各种算术运算符的使用举例如图 2.15 所示。

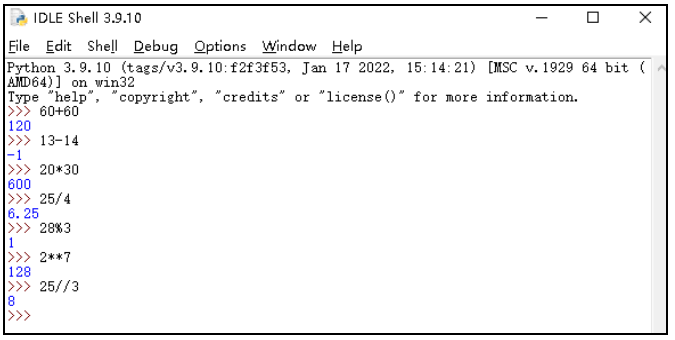

图 2.15　各种算术运算符的使用举例

2. 赋值运算符

赋值运算符主要用于为变量等赋值，能将运算符右侧的常量或变量的值，赋给运算符左侧的变量。Python 中常用的赋值运算符及其作用如表 2.3 所示。

表 2.3　Python 中常用的赋值运算符及其作用

运算符	作用	说明
=	直接赋值	c = a + b 即为 a + b 的运算结果赋值给 c
+=	增强加法赋值	c += a 即为 c = c + a
−=	增强减法赋值	c −= a 即为 c = c − a
*=	增强乘法赋值	c *= a 即为 c = c * a
/=	增强除法赋值	c /= a 即为 c = c / a
%=	增强求余赋值	c %= a 即为 c = c % a
//=	增强整除赋值	c //= a 即为 c = c // a
**=	增强幂赋值	c **= a 即为 c = c ** a

各种赋值运算符的使用举例如图 2.16 所示。

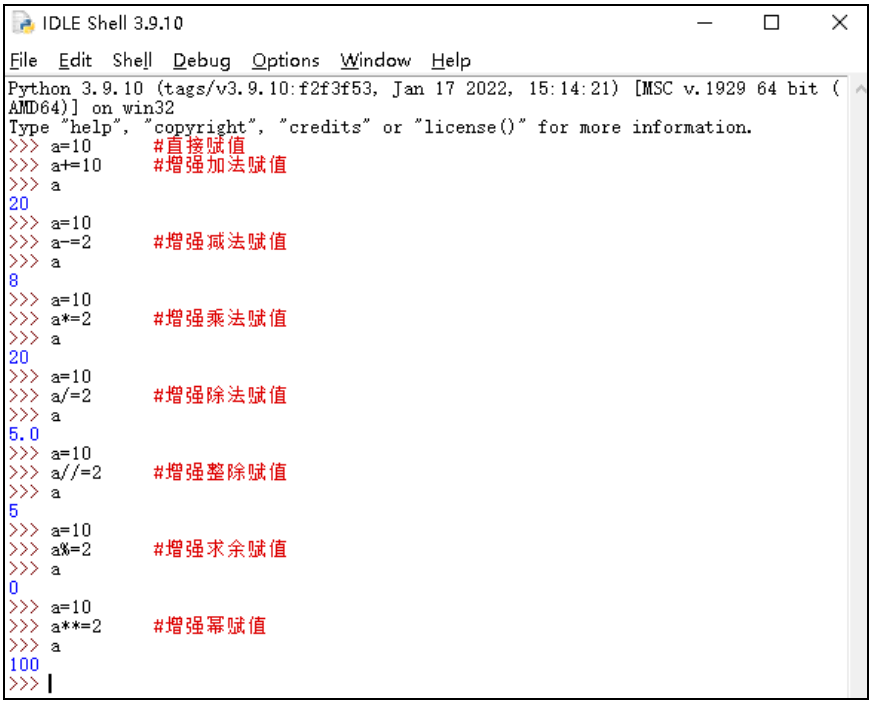

图 2.16　各种赋值运算符的使用举例

3. 比较运算符

比较运算符也称关系运算符，主要用于对数值、变量或表达式的结果进行大小、真假等比较。如果比较结果为真，则返回 True；如果比较结果为假，则返回 False。比较运算符通常用于条件语句中，作为判断依据。Python 中常用的比较运算符及其作用如表 2.4 所示。

表 2.4　Python 中常用的比较运算符及其作用

运算符	作用	说明
==	等于：比较两个对象是否相等	a == b，返回 True 或 False
!=	不等于：比较两个对象是否不相等	a != b，返回 True 或 False
>	大于：比较对象的大小	a > b，返回 True 或 False
<	小于：比较对象的大小	a < b，返回 True 或 False
>=	大于等于：比较对象的大小	a >= b，返回 True 或 False
<=	小于等于：比较对象的大小	a <= b，返回 True 或 False

各种比较运算符的使用举例如图 2.17 所示。

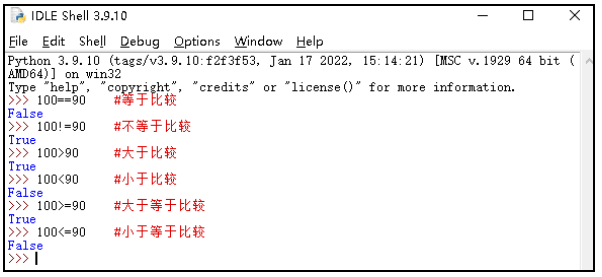

图 2.17　各种比较运算符的使用举例

4. 逻辑运算符

逻辑运算也称布尔运算，运算结果仍是布尔值。逻辑运算包含逻辑与、逻辑或、逻辑非三种基本运算。Python 中对应的逻辑运算符分别是 and(逻辑与)、or(逻辑或)和 not(逻辑非)。Python 中常用的逻辑运算符及其作用如表 2.5 所示。

表 2.5　Python 中常用的逻辑运算符及其作用

运算符	作用	说明
and	逻辑与运算	a and b，如果 a、b 都为真，则返回真
or	逻辑或运算	a or b，如果 a 为真或 b 为真，则返回真
not	逻辑非运算	not(a and b)，如果 a and b 结果为假，则返回真

各种逻辑运算符的使用举例如图 2.18 所示。

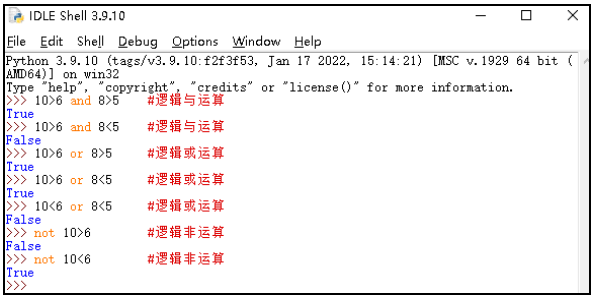

图 2.18　各种逻辑运算符的使用举例

5. 位运算符

计算机程序中的所有数值均以二进制形式存储，为方便计算，Python 提供了位运算符。Python中的位运算符有位与(&)、位或(|)、异或(^)、取反(~)、左移位(<<)和右移位(>>)运算符。Python 中常用的位运算符及其作用如表 2.6 所示。

表 2.6　Python 中常用的位运算符及其作用

运算符	作用	说明
&	位与运算	a&b，参与运算的两个值，如果两个相应位都为 1，则该位的结果为 1，否则为 0
|	位或运算	a|b，只要对应的两个二进制位有一个为 1，结果位就为 1
^	异或运算	a^b ，当两个对应的二进制位相异时，结果为 1
~	取反运算	~a，对数据的每个二进制位取反，即把 1 变为 0，把 0 变为 1
<<	左移位运算	a<<b，运算数的各二进制位全部左移若干位，由<<右边的数指定移动的位数，高位丢弃，低位补 0
>>	右移位运算	a>>b，把>>左边的运算数的各二制进位全部右移若干位，>>右边的数指定移动的位数

各种位运算符的使用举例如图 2.19 所示。

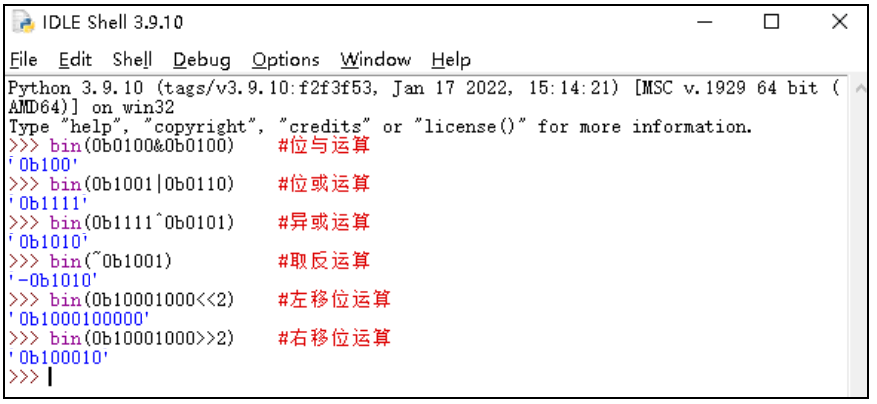

图 2.19　各种位运算符的使用举例

6. 成员运算符

除了上述运算符，Python 还支持成员运算符，用于检查某数据是否存在于包含多个成员的数据类型(如字符串、列表、元祖、字典等)中。如果是成员关系，则 in 返回真，否则返回假；not in 则相反。成员运算符及其作用如表 2.7 所示。

表 2.7　成员运算符及其作用

运算符	作用	说明
in	在指定的序列中找到值，返回 True，否则返回 False	a in b，其中 b 为字符串
not in	在指定的序列中没有找到值，返回 True，否则返回 False	a not in b，其中 b 为字符串

成员运算符的使用举例如图 2.20 所示。

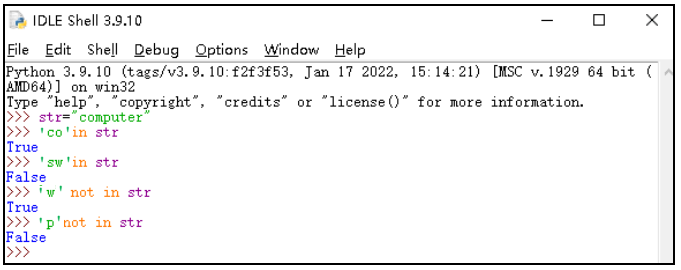

图 2.20　成员运算符的使用举例

在一个运算中，若存在多种运算符，会先执行优先级高的运算，再执行优先级低的运算，而对于同一优先级的操作，则按从左到右的顺序进行。Python 为所有支持的运算符都设定了优先级，各运算符从高到低的优先级顺序如表 2.8 所示。

表 2.8　各运算符的优先级顺序

运算符	作用
**	指数或乘方(最高优先级)
*、/、%、//	乘、除、取余、取整
+、−	加法、减法
>>、<<	右移、左移运算符
&	位与运算
^	异或运算
|	位或运算
in、not in、<、<=、>、>=、!=、==	比较运算，包括成员检测
not	逻辑非运算
and	逻辑与运算
or	逻辑或运算
:=	赋值运算符(最低优先级)

2.2　Python 语法基础

语法基础规定了 Python 语言使用的最根本规则，是编写程序必须遵循的基本规范。在学习 Python 时，需要了解其语法特点，包括代码注释、代码缩进、编码规范等内容。下面将详细介绍 Python 的语法特点。

2.2.1　代码注释

通常，商品都有自己的标签，用来说明规格、大小、型号等信息，方便购买者更全面、更清晰地了解商品；语文课本中的古诗词、文言文也都有一定的标注，用来对句子进行说明，方便读者更好地理解内容。代码中的注释作用与此类似，用于对代码语句进行说明，从而提高代

码的可读性，方便程序员理解代码功能。在代码执行过程中，Python 解释器会忽略注释内容，不会对其进行执行。在 Python 中，注释主要有两种类型，分别是单行注释和多行注释。

1. 单行注释

在 Python 中，使用#作为单行注释的符号。从#开始到该行末尾的所有内容都被视为注释，不会被执行。单行注释可以放在要注释代码的前一行，也可以放在代码的右侧。添加注释时，应确保其有意义，即能够充分解释代码的功能及用途。单行注释的使用如图 2.21 所示。

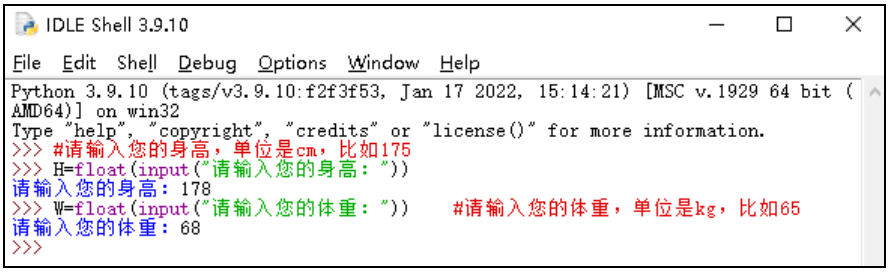

图 2.21　单行注释的使用

2. 多行注释

在 Python 中，并没有专门的多行注释标记，而是将包含在一对三个单引号('''.....''')或三个双引号("""....."""")之间且不属于任何语句的内容视为注释，这样的内容将被解释器忽略。由于这些内容可以跨多行编写，因此称为多行注释。多行注释通常用于为 Python 文件、模块、类或函数等添加版权、功能等说明信息。多行注释的使用如图 2.22 所示。

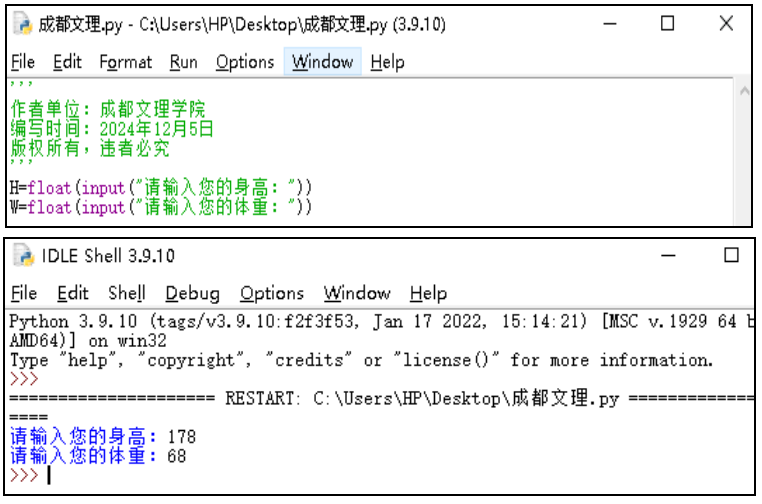

图 2.22　多行注释的使用

2.2.2　代码缩进

Python 不像其他程序设计语言(如 Java 或 C 语言)那样使用大括号({})来分隔代码块，而是采用代码缩进和冒号(:)来区分代码之间的层次。

1. 代码缩进的表示

在编写代码过程中，缩进可以使用 Tab 键或空格来实现。当使用 Tab 键时，一个 Tab 键表示一个缩进量，若需要两个缩进量，则可以按两次 Tab 键；当使用空格时，通常采用 4 个空格作为一个缩进量，若需要两个缩进量，则使用 8 个空格，以此类推。Python 的编程规范建议：缩进最好采用空格形式，每一层向右缩进 4 个空格，一般不建议使用 Tab 键进行缩进。

2. 代码缩进的使用

在编写 Python 代码时，并不是所有代码都需要缩进。缩进主要用于表示代码之间的所属关系，常见于类定义、函数定义、流程控制语句、异常处理语句等。通常，行尾的冒号和下一行的缩进表示一个代码块的开始，而缩进结束则表示该代码块的结束。代码缩进的使用如图 2.23 和图 2.24 所示。

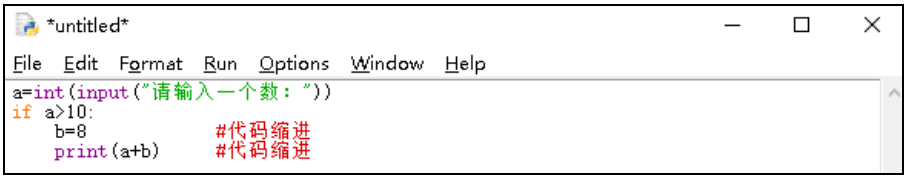

图 2.23　代码缩进的使用(1)

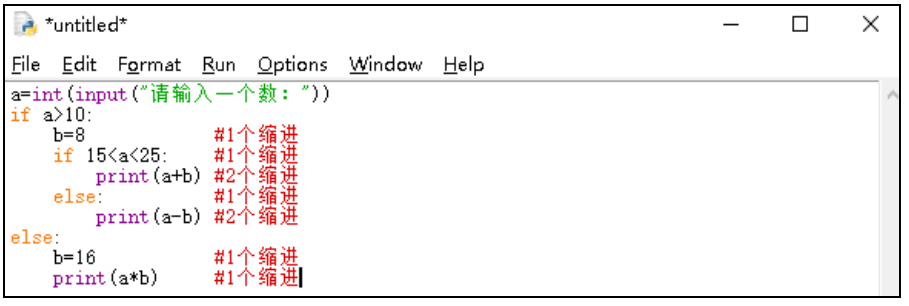

图 2.24　代码缩进的使用(2)

需要注意的是，Python 对代码缩进的要求非常严格，同一级别的代码块必须使用相同的缩进量，且缩进符号(Tab 键、空格键等)也应保持一致，这样才能保证嵌套正确。如果缩进不规范，例如，有的地方用 4 个空格，有的地方用 3 个空格，Python 解释器会抛出 SyntaxError 异常。代码缩进不一致导致的程序错误如图 2.25 所示。

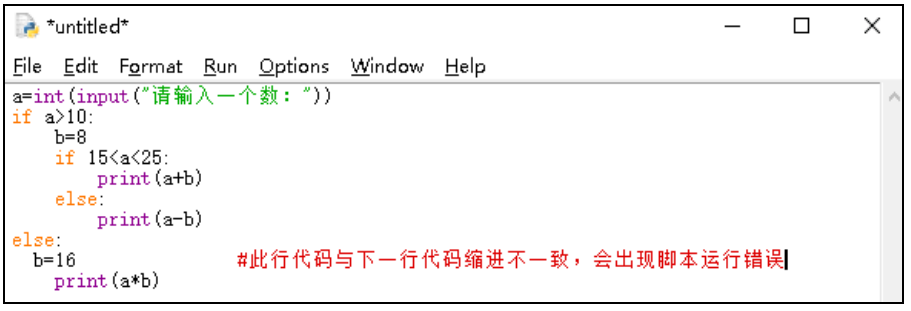

图 2.25　代码缩进不一致导致的程序错误

2.2.3 编码规范

在编写代码的过程中，遵循一定的编写规则和命名规范可以使代码更加规范化，对代码的理解与维护都起到至关重要的作用。Python 采用 PEP8 作为编码规范，其中 PEP 是 Python Enhancement Proposal(Python 增强建议书)的缩写，而“PEP8”中的“8”表示版本号。下面以 PEP8 为基础，介绍编写规则和命名规范的相关知识。

1. 编写规则

(1) 每个 import 语句只导入一个模块，应尽量避免一次导入多个模块。

(2) 不要在行尾添加分号“;”，也不要用分号将两条命令放在同一行。

(3) 建议每行不超过 80 个字符，如果超过，建议使用小括号“()”将多行内容隐式连接，不推荐使用反斜杠“\”进行连接。以下两种情况除外。

① 导入模块的语句过长。

② 注释里的 URL。

(4) 使用必要的空行可以增加代码的可读性。一般在顶级定义(如函数或类的定义)之间空两行，方法定义之间空一行。另外，在用于分隔某些功能的位置也可以空一行。

(5) 通常情况下，运算符两侧、函数参数之间、逗号“,”两侧建议使用空格进行分隔。

(6) 应避免在循环中使用“+”和“+=”运算符累加字符串。因为字符串是不可变的，这样做会创建不必要的临时对象。推荐将每个子字符串加入列表，然后在循环结束后使用 join()方法连接列表。

(7) 适当使用异常处理结构可以提高程序的容错性，但不能过多依赖异常处理，适当的显式判断也是必要的。

2. 命名规范

命名规范在编写代码过程中至关重要。尽管不遵循命名规范程序也可以运行，但规范的命名可以让人更直观地理解代码所代表的含义。Python 中常用的命名规范如下。

- 模块名：尽量短小，全部使用小写字母，可以用下画线分隔多个单词。
- 包名：尽量短小，全部使用小写字母，不推荐使用下画线。
- 类名：采用驼峰式(pascal)命名法，所有单词的首字母都必须大写，单词之间需连在一起不能空格，如 BorrowBook、LastName。
- 变量名：小写，单词之间用下画线“_”连接。尽量不要使用字符“i”(小写字母)、“O”(大写字母)或“I”(i 的大写字母)作为单个字符变量名，因为这些字符与数字“1”和“0”可能难以区分。
- 函数名：小写，单词之间用下画线“_”连接。
- 常量名：所有字母大写，单词之间用下画线“_”连接。
- 模块内部的类：采用“下画线＋驼峰式”命名，例如，在 BorrowBook 类中的内部类，可以使用_BorrowBook。
- 私有成员：使用双下画线“__”开头的实例变量或方法是类私有的。

2.3 标识符与关键字

Python 程序代码中出现的如 User_Name、math 和 print 等英文单词，是 Python 为变量、模块和函数等对象赋予的名称，这些名称称为标识符。标识符用于标识程序中不同的对象，并访问这些对象所指向的数据。为程序中不同对象取名字的过程，称为命名。另外，还有一些具有特殊语法含义的标识符，称为关键字或保留字。

2.3.1 标识符

标识符可以简单地理解为一个名字(比如每种水果都有自己的名字)，主要用来标识变量、函数、类、模块和其他对象。在计算机语言中，标识符是允许作为名称使用的有效字符串集合。在 Python 中，程序员可以使用大写字母、小写字母、数字、下画线和汉字等字符组合进行命名，这些组合起来的字符串就是标识符。

Python 语言标识符命名规则如下。

- 由字母、下画线“_”和数字组成。在 Python 中，标识符的第一个字符不能是数字，且只允许使用 ISO-Latin 字符集中的 A～Z 和 a～z 作为字母。
- 不能使用 Python 中的保留字。
- 区分字母大小写。在 Python 中，标识符中的字母是严格区分大小写的，如果两个同样单词的大小写格式不一样，则所代表的意义是完全不同的。
- 在 Python 中，以下画线开头的标识符有特殊意义，一般应避免使用相似的标识符。
  - 以单下画线开头的标识符(如_length)表示不能直接访问的类属性，另外，也不能通过“from xxx import *”导入；
  - 以双下画线开头的标识符(如__add)表示类的私有成员；
  - 以双下画线开头和结尾的是 Python 中专用的标识，如__init__()表示构造函数。

2.3.2 关键字

关键字(keyword)，也称为保留字，是指在 Python 语言中已被赋予特定意义的单词。在命名过程中，不能将它们作为标识符为变量、函数、类、模板及其他对象命名。在 Python 交互式模式下，可使用>>>import keyword 和>>>keyword.kwlist 命令查看 Python 所提供的关键字，如下所示。

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

Python 中的关键字如表 2.9 所示。

表 2.9　Python 中的关键字

and	as	assert	break	class	continue
def	del	elif	else	except	finally

(续表)

for	from	False	global	if	import
in	is	lambda	nonlocal	not	None
or	pass	raise	return	try	True
while	with	yield			

2.3.3 命名错误抛出异常

在编写程序时，如果没有按照上述规则命名，就会抛出异常并显示错误信息。例如，命名时若用数字开头，就会显示 SyntaxError：invalid syntax 信息(表示语法错误)，如图 2.26 所示。需要注意的是，这类命名错误虽然会抛出异常，但并不会明确指出错误产生的具体原因，需要开发者自行检查。

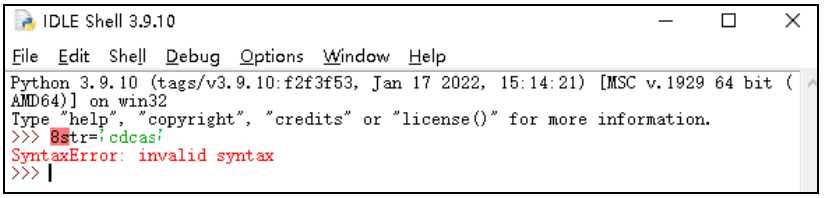

图 2.26 命名异常

2.4 Python 中的计算

在交互模式下，用户可将 Python 当作计算器使用，直接输入需要计算的表达式，即可快速得出结果。此外，通过 Python 提供的丰富函数，还可以进行各类数学计算。

2.4.1 直接算术运算

在 Python 交互式命令行中，用户可以直接进行基本的数学计算。只需在命令行状态下输入算式，即可计算并输出结果。直接算术运算的示例如图 2.27 所示。

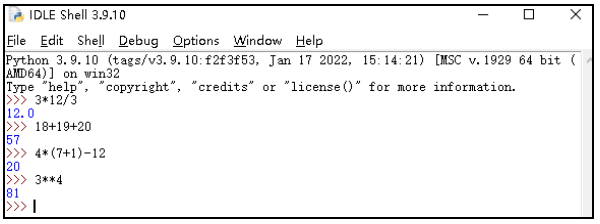

图 2.27 直接算术运算的示例

2.4.2 math 模块中丰富的数学函数

Python 提供了许多内置模块，其中 math 模块包含丰富的数学函数。在进行相关计算时，可以借助这些函数来完成计算，从而提高计算效率。math 模块中的常用数学函数及其作用如表 2.10 所示。

表 2.10　math 模块中的常用数学函数及其作用

函数	作用
sin(x)	求 x 的正弦值
cos(x)	求 x 的余弦值
asin(x)	求 x 的反正弦值
acos(x)	求 x 的反余弦值
tan(x)	求 x 的正切值
atan(x)	求 x 的反正切值
hypot(x,y)	求直角三角形斜边的长度
fmod(x,y)	求 x/y 的余数
ceil(x)	取不小于 x 的最小整数
floor(x)	取不大于 x 的最大整数
fabs(x)	求绝对值
exp(x)	求 e 的 x 次幂
pow(x,y)	求 x 的 y 次幂
log10(x)	求 x 的以 10 为底的对数
sqrt(x)	求 x 的平方根

使用上述数学函数前，需先用 import math 命令导入模块。使用 math 模块函数实现的计算如图 2.28 所示。

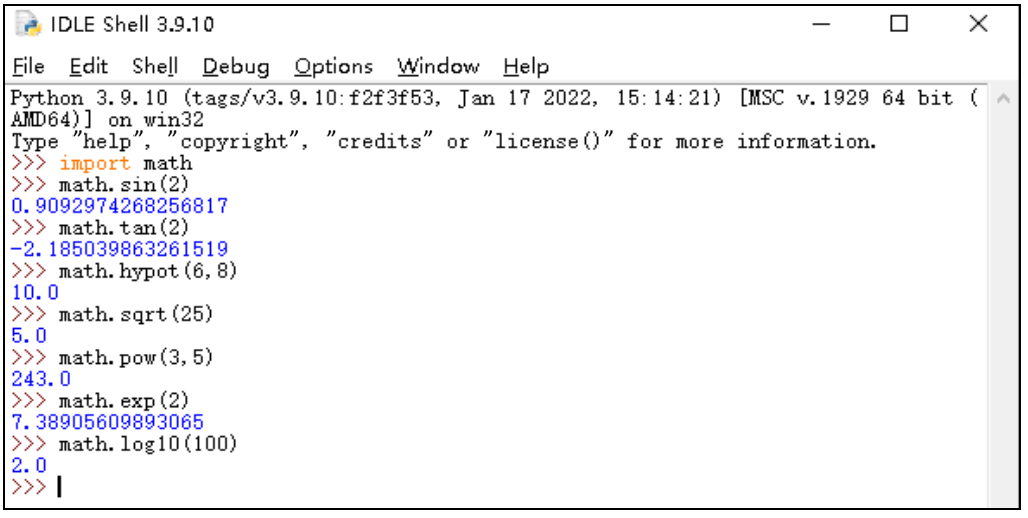

图 2.28　使用 math 函数模块实现的计算

2.5 基本输入与输出操作

2.5.1 输入函数 input()

在 Python 中，input()是内置的基本输入函数，用户可直接调用。它从计算机的输入设备(默认是键盘)上读取数据，其语法格式如下。

```
input(<提示性文字>)
```

其中，<提示性文字>为可选参数(类型为字符串)，用于提示用户输入信息。无论用户输入什么内容，该函数返回的结果均为字符串。若用户需要的是数值，则必须使用 int()函数或 float()函数对返回值进行类型转换。输入函数 input()的使用示例如图 2.29 所示。

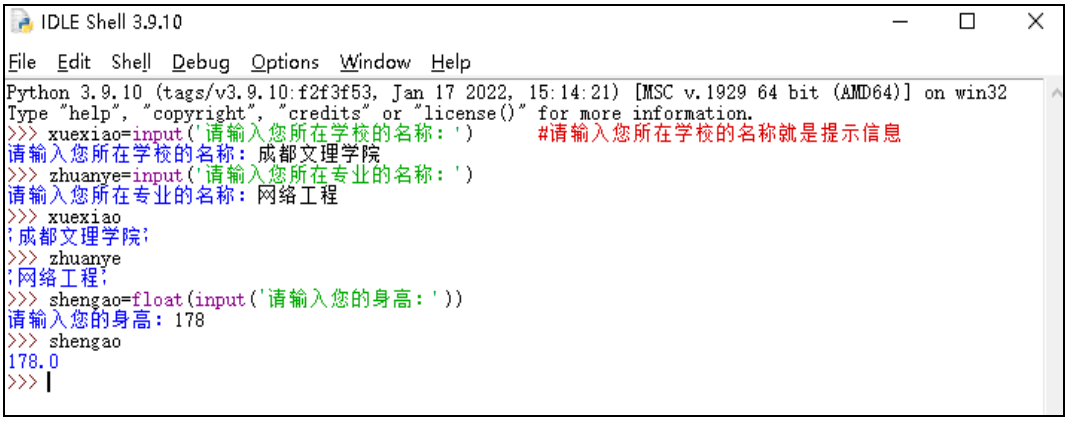

图 2.29　输入函数 input()的使用示例

2.5.2 输出函数 print()

在 Python 中，print()是内置的基本输出函数，用户可以直接调用。print()函数可以输出 Python 中的所有数据类型的值，并且不需要事先指定要输出的数据类型。根据输出个数的不同，该函数的语法格式分为以下两种情况。

1. 输出单个信息

输出单个信息的语法格式如下。

```
print(待输出的信息)
```

其中，“待输出的信息”可以是多种数据类型，如输出信息本身或输出计算结果。单个信息输出的示例如图 2.30 所示。

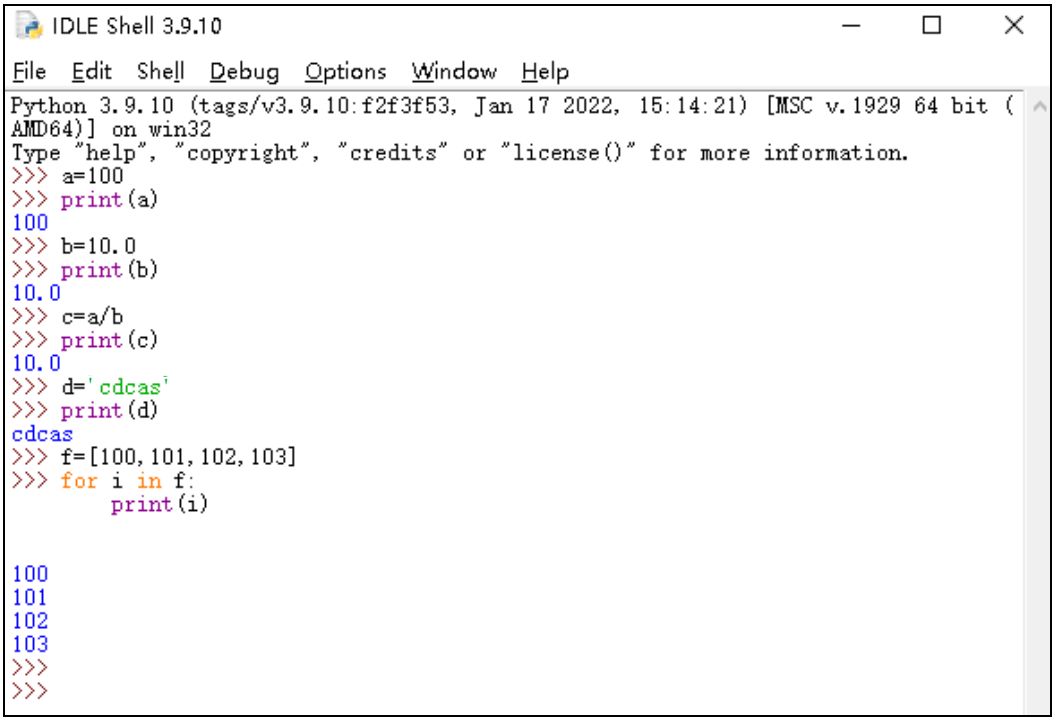

图 2.30　输出函数 print()的使用(单个信息输出)

2. 输出多个信息

输出多个信息的语法格式如下。

```
print(待输出的信息 1, 待输出的信息 2, 待输出的信息 3,…, 待输出的信息 n)
```

其中，多个输出信息之间用逗号进行分隔。输出时，每个待输出的信息之间会自动用空格进行分隔。多个信息输出的示例如图 2.31 所示。

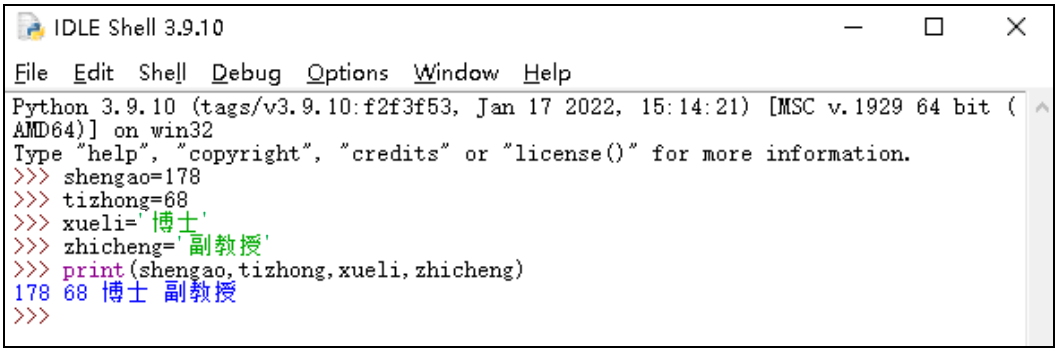

图 2.31　输出函数 print()的使用(多个信息输出)

2.6 本章小结

本章首先介绍了 Python 中较为简单的数字型和布尔型两种数据类型，详细介绍了数字类型中的整数类型、浮点型和复数类型，并举例说明了这些数据类型的使用方法及注意事项。随后介绍了针对这些数据的运算符及其优先级等内容，以帮助初学者对 Python 的数据及相关运算有更深刻的理解。

其次介绍了 Python 的语法基础。语法基础是 Python 语言使用的根本规则，是编写程序的基础规范。初学者需要了解它的语法特点，如代码注释、代码缩进、编码规范等。在掌握语法基础后，本章又详细介绍了编写代码过程中常用的标识符和关键字。通过这部分内容的学习，初学者能够了解标识符和关键字的作用及使用规则。

最后介绍了 Python 中的计算方法和常用到的 math 模块函数，以及 Python 中常用的输入函数 input()和输出函数 print()，并详细讲解了这两个函数的使用方法及注意事项。

本章的每一节内容都以相关案例的方式做了详细讲解。相信通过本章的学习，读者能够对 Python 的编程规则有更全面、更深入的理解，同时也为更好地使用 Python 编写程序打下坚实基础。