高等院校计算机应用系列教材

Python 程序设计语言

(第二版) (微课版)

李美珊　刘　越　陈育德　主　编
韦韫韬　李春洁　王　超　副主编

清華大學出版社
北　京

内 容 简 介

本书致力于培养读者的计算思维能力，重点提升他们运用计算思维解决实际问题的水平。全书以 Python 语言为基础，对计算机程序设计的知识体系展开了全方位且有条理的阐述。在编写过程中，通过程序实例难度呈螺旋式递增的独特设计，巧妙推动知识难度逐步攀升，这种编排模式对于编程领域的初学者而言极为友好，有助于初学者开启编程之旅。

本书共 12 章，内容主要包括计算机基础及 Python 概述，基本数据类型、运算符与表达式，程序控制结构，组合数据类型，字符串操作，函数，文件和文件夹操作，Python 异常处理，中文文本分析，科学计算与数据分析，网络爬虫技术，Python 计算生态等。

本书内容丰富，由浅入深，既可作为普通高等院校 Python 程序设计语言课程的教材，也可供从事相关工作的工程师和爱好者阅读使用。

本书配套的电子课件、知识导图、教学计划、教学大纲、实验大纲、授课方案、习题答案和实例源文件可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以扫描前言中的二维码获取。扫描前言中的视频二维码可以直接观看教学视频。

图书在版编目(CIP)数据

Python 程序设计语言 ：微课版 / 李美珊, 刘越, 陈育德主编. -- 2 版. -- 北京 ：清华大学出版社, 2025. 6.
(高等院校计算机应用系列教材). -- ISBN 978-7-302-69463-2

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2025JY0172 号

责任编辑：胡辰浩
封面设计：高娟妮
版式设计：妙思品位
责任校对：成凤进
责任印制：刘　菲

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：涿州市般润文化传播有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**18.75　　**字　　数：**480 千字
版　　次：2022 年 1 月第 1 版　　2025 年 7 月第 2 版　　**印　　次：**2025 年 7 月第 1 次印刷
定　　价：59.00 元

产品编号：111418-01

前　言

追溯 Python 语言的发展历程，其研制工作始于 1989 年并于 1991 年推出第一个版本，Python 一开始就具备类、函数、异常处理、包含列表和词典在内的核心数据类型以及以模块为基础的拓展生态库等语言特性。Python 一经推出就获得专业人士的喜爱，经过 30 多年的发展，Python 已被广泛应用到系统编程、网络爬虫、人工智能、科学计算、大数据、数据分析、数据挖掘、云计算、图像开发、深度学习、Web 开发、系统运维等众多领域。

近几年，Python 在 TIOBE 语言排行榜上的位次不断上升，2024 年 5 月 Python 已经位列第一，远远超过 C 语言，这主要得益于 Python 语言具备如下一系列卓越优点。首先，Python 的语法简洁明晰且富有优雅气质，堪称一门极为简单易学的编程语言。对于编程初学者而言，其入门的门槛较低，能够让毫无编程基础的人快速上手。并且，随着学习进程的推进与深入，它又具备强大的拓展性与深度，足以支撑学习者去编写一些极为复杂且功能强大的程序。无论是处理大规模数据运算，还是构建复杂的软件架构，Python 语言都能应对自如。

其次，Python 的开发效率令人瞩目。它拥有极为强劲的第三方库体系，这一体系的丰富性与完备性几乎涵盖了人们通过计算机想要达成的所有功能需求。换言之，在 Python 的庞大资源库中，总能找到与之对应的功能模块。开发者仅需将所需模块直接下载并顺利调用，随后便能够在稳固的基础库之上迅速展开开发工作。如此一来，便能够极大幅度地缩短整个项目的开发周期，让创新想法能够更快地转化为实际应用。同时，也巧妙地避免了在开发过程中对已有基础功能进行重复构建的低效行为，使开发者能够将更多的精力与智慧倾注于独特功能的研发与优化之中，从而显著提升开发的整体效率与质量。

Python 语言简单易学，在高校计算机相关专业的教学研究中已取得丰硕成果，但随着跨学科建设不断深入，各学科不断融合，各专业与计算机科学的联系更加紧密，计算机基础课程的 Python 语言教学迫在眉睫。本书是针对基础教学编写的 Python 程序设计语言教材，旨在让零基础的各专业学生在有限的时间中学习编程思想、掌握 Python 的语法特点，并能付诸实践，为所学专业服务，推进各学科与计算机的交叉。本书的编写思想是为教学服务，秉承着零起点、通俗易懂，抛弃复杂、晦涩、脱离现实的案例，融入课程思政教学理念，培养学生的“计算思维”和“逻辑思维”的应用能力。

本书基于 Python 3.8 编写而成，所有内容都已经过反复推敲、讨论。全书包含大量的实例，讲解由浅入深、循序渐进，内容包括计算机基础及 Python 概述，基本数据类型、运算符与表达式，程序控制结构，组合数据类型，字符串操作，函数，文件和文件夹操作，Python 异常处理，中文文本分析，科学计算与数据分析，网络爬虫技术，Python 计算生态等。

本书共 12 章，第 1 章由王超编写，第 2 章由孙志勇编写，第 3、4、8 章由陈育德编写，第 5 章由韦韫韬编写，第 6、11、12 章由李美珊编写，第 7 章由李春洁编写，第 9 章由张竞达编写，第 10 章由刘越编写，最后由李美珊老师统稿、薛佳楣教授主审。

由于作者水平有限，书中难免有不足之处，恳请专家和广大读者批评指正。在编写本书的过程中参考了很多文献和网络素材，在此向这些文献的作者深表感谢。我们的电话是 010-62796045，邮箱是 992116@qq.com。

本书配套的电子课件、知识导图、教学计划、教学大纲、实验大纲、授课方案、习题答案和实例源文件可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以扫描下方左侧的二维码获取。扫码下方右侧的视频二维码可以直接观看教学视频。

编　者

2025 年 3 月

目　录

第 1 章

计算机基础及Python概述

学习目标

- 掌握计算机系统的组成。
- 理解程序设计语言的发展历程。
- 理解 Python 语言的特点及应用领域。
- 掌握 Python 开发环境的安装及配置方法。
- 掌握 Python 语言的编码规范。
- 掌握 Python 扩展库的导入和使用方法。

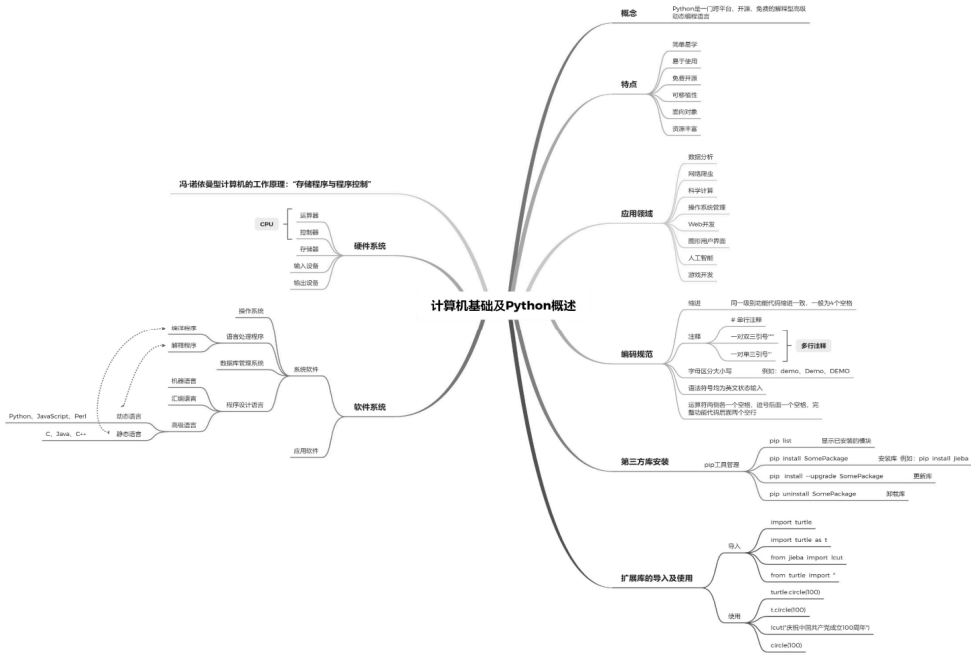
学习重点

掌握计算机系统的组成、冯•诺依曼型计算机的核心思想和特点、计算机硬件系统和软件系统的组成，理解程序设计语言概念及语言处理程序，理解 Python 语言的特点及应用领域，掌握 Python 语言的编码规范，掌握 Python 扩展库的导入和使用方法。

学习难点

pip 工具的使用方法，Python 扩展库的导入方式。

知识导图



1.1 计算机基础概述

计算机无疑是人类社会 20 世纪最伟大的发明之一，并且一直在以令人难以置信的速度快速发展。计算机的出现彻底改变了人类社会的文化和生活，并且对人类的整个历史发展都有着不可估量的影响。随着人类进入物联网时代，计算机已经成为人们社会生活中不可缺少的工具之一。

计算机系统由硬件系统和软件系统两部分组成，如图 1-1 所示。

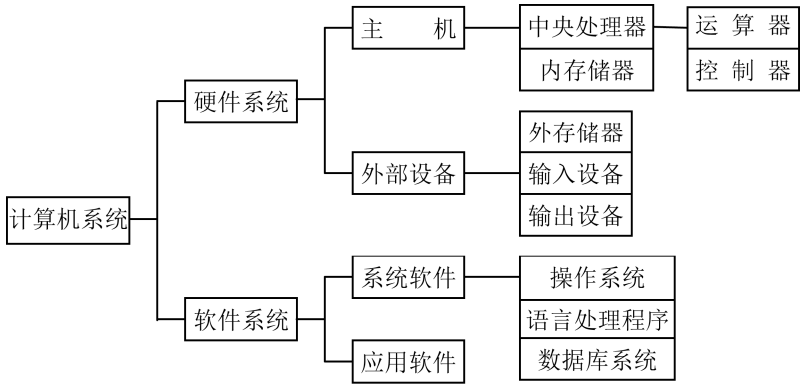


图 1-1　计算机系统的组成

1.1.1　冯 • 诺依曼结构

尽管计算机的发展经历了 4 代，但工作原理都基于冯 • 诺依曼于 1946 年提出的“存储程序与程序控制”思想。存储程序是指人们必须事先把计算机的计算步骤(即程序)及运行中所需的数据，通过一定方式输入并存储在计算机的存储器中。程序控制是指计算机在运行时能自动地逐一取出程序中的一条条指令，加以分析并执行规定的操作。

典型的冯 • 诺依曼计算机是以运算器为中心的。其中，输入设备、输出设备与存储器之间的数据传送都须通过运算器。

现代的计算机已转为以存储器为中心，如图 1-2 所示，图中的实线箭头为控制流、虚线箭头为指令流、双线箭头为数据流。

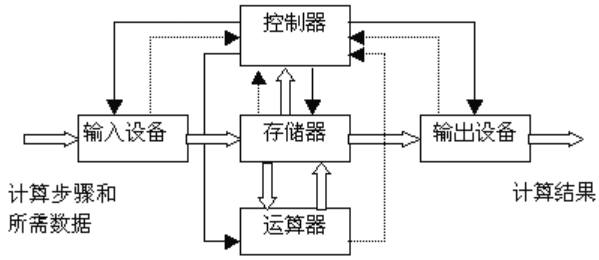


图 1-2　冯 • 诺依曼型计算机的结构

1.1.2　计算机硬件系统

计算机硬件系统是组成计算机系统的各种物理设备的总称，是计算机系统的物质基础，同时也是看得见、摸得着的一些实实在在的有形实体。

1. 存储器

存储器分为主存储器(简称主存)和辅助存储器(简称辅存)。主存可直接与 CPU 交换信息，辅存又叫外存。

2. 运算器

运算器是计算机中处理数据的核心部件，主要由执行算术运算和逻辑运算的算术逻辑单元(arithmetic logic unit，ALU)、存放操作数和中间结果的寄存器组以及连接各部件的数据通路组成，用以完成各种算术运算和逻辑运算。

3. 控制器

控制器是计算机中负责控制管理的核心部件。CPU 和主存储器是信息加工处理的主要部件，我们通常把这两部分合称为主机。

4. 输入输出设备

输入输出设备(简称 I/O 设备)又称为外部设备，作用是与计算机主机进行信息交换以实现人机交互。

1.1.3　计算机软件系统

软件包括可在计算机硬件上运行的各种程序、数据及相关文档。我们通常把计算机软件分为系统软件和应用软件两大类。

1. 系统软件

系统软件是维持计算机系统正常运行并支持应用软件运行的基础软件，包括操作系统、语言处理程序和数据库管理系统等。

2. 应用软件

应用软件也称为应用程序，是由专业的软件公司针对应用领域的需求，为解决某些实际问题而研制开发的程序，此外也可以是由用户根据需要编制的各种实用程序。应用软件通常需要系统软件的支持，才能在计算机硬件上有效运行。例如，文字处理软件、电子表格软件、制图软件、网页制作软件、财务管理软件等均属于应用软件。

1.2　程序设计语言

1.2.1　程序设计语言概述

程序设计语言是计算机理解和识别用户操作意图的一种交互体系，它能按照特定规则组织计算机指令，使计算机自动进行各种运算处理。按照程序设计语言规则组织起来的一组计算机指令称为计算机程序。程序设计语言也叫编程语言。

程序设计语言分为三大类：机器语言、汇编语言和高级语言。

- 机器语言是一种二进制语言，这种语言直接使用 0、1 代码表示指令。机器语言是计算机硬件可以直接识别和执行的程序设计语言。
- 汇编语言采用助记符来代替机器语言中的指令和数据，又称为符号语言。
- 高级语言是一种完全符号化的语言，这种语言采用了自然语言(英语)中的词汇和语法习惯，容易理解和掌握。高级语言完全独立于具体的计算机，具有很强的可移植性。用高级语言编写的程序称为源程序，源程序不能在计算机中直接执行，而必须翻译或解释成目标程序，才能被计算机理解和执行。

1.2.2 编译和解释

高级语言根据计算机执行机制的不同可分成两类：静态语言和动态语言。静态语言采用编译的方式执行，大多数静态语言要求变量在使用之前必须声明数据类型；动态语言一般指脚本语言，采用解释方式执行，运行时才确定数据类型，变量在使用之前无须声明数据类型，变量在使用时，变量的值和数据类型由赋予的值决定。例如，Java、C、C++等都是静态语言，而 Python、PHP、JavaScript、Perl 等都是动态语言。静态语言和动态语言定义变量的方式如图 1-3 所示。

静态语言

```
1  string s="hello world! ";
2  boolean b=true;
3  int i=10;
```

动态语言

```
1  s = "hello world! "
2  b = True
3  i = 10
```

图 1-3　定义变量

编译是将源代码转换成目标代码的过程。通常，源代码是高级语言代码，目标代码是机器语言代码，执行编译的计算机程序称为编译器。图 1-4 展示了程序的编译和执行过程。其中，虚线表示目标代码被计算机运行。

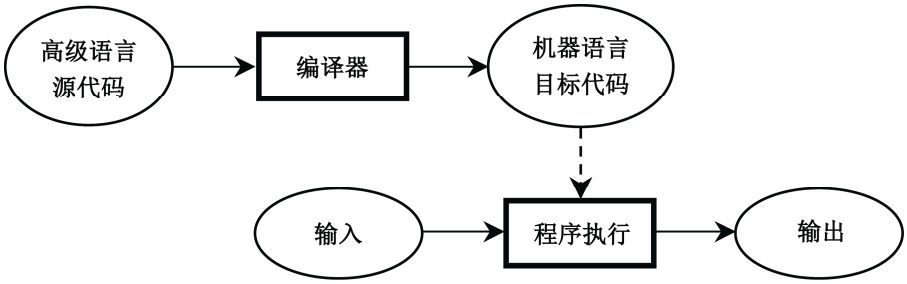


图 1-4　程序的编译和执行过程

解释是将源代码逐条转换成目标代码并同时逐条运行目标代码的过程。执行解释的计算机程序称为解释器。图 1-5 展示了程序的解释和执行过程。其中，高级语言源代码与数据被一同输入解释器，然后输出运行结果。

编译和解释的区别在于编译是一次性翻译，不再需要编译程序或源代码。解释则在程序每次运行时都需要解释程序或源代码。两者的区别类似于外语的完整翻译和同声传译。

Python 是一种被广泛使用的高级动态编程语言，采用解释方式执行，但 Python 解释器保留了编译器的部分功能，随着程序的运行，解释器也会生成完整的目标代码。这种将解释器和编

译器结合在一起的新型解释器是现代脚本语言为了提升计算性能而做的一种有益改进。

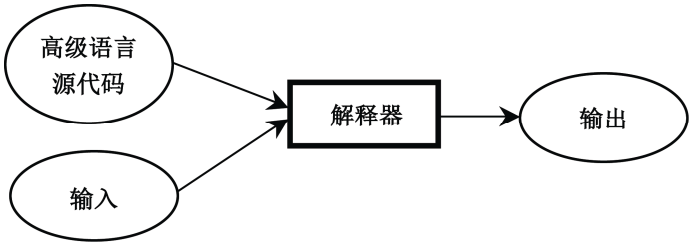


图 1-5　程序的解释和执行过程

1.2.3　计算机编程方法

1. 学习编程的意义

苹果公司创始人史蒂夫 • 乔布斯(Steve Jobs)曾经说过：“在工作中是否要编程未必那么重要，但你可以把它当成一面镜子，一面你思考的镜子。我认为学习如何思考是最有价值的。世界上的每个人都应该学习如何编写程序，因为它能教会你如何思考。正如人们学习法律未必要当律师，但学习法律可以告诉你如何从法律的角度思考问题。同样，编程是一种稍微不同的思考方法。因此，我认为计算机科学是一门基础学科，每个人都应该在一生中花费一年的时间学习如何编程。”

这段话从另外一个角度阐述了学习编程的重要性。学习编程实际上是一种思维训练，学习如何严谨科学地分析问题、寻找解决思路、设计解决方案。这种思维训练对人的一生，无论是工作还是生活，都会带来极大的好处。

2. 编写程序的方法

计算机程序由各种指令组成，一个程序如果具备了解决某个问题的功能，那么实际上体现的是程序设计者对于该问题的分析和解决思路。例如，设计一个计算圆面积的程序，大脑对于如何求圆的面积通常会做如下分析。

(1) 为了求出圆的面积，首先需要知道圆的半径。

(2) 知道半径以后，根据圆面积的计算公式可以计算出圆的面积。

(3) 求出面积后，根据要求输出结果。

这个问题比较简单，所以分析问题和设计解决方案的过程也较简单，把以上解决思路落实到具体的程序代码上，就成了求解圆面积的实际程序：

```
1    r = eval(input("请输入圆的半径："))    # 用户通过键盘输入圆的半径并存放到 r 中
2    area = 3.14 * r * r                    # 根据圆面积公式计算圆的面积并存放到 area 中
3    print(area)                            # 输出圆面积 area 到显示器
```

上述程序用了 3 行 Python 代码来解决圆面积的求解问题，基本上对应了前面的分析过程，看起来非常简单。但是，如果需要解决的问题很复杂，那么相关的分析过程和解决方案通常也会很复杂，与其对应的实现代码也必然不再简单。

由此可见，程序代码只是程序设计者对于某个问题解决方案的计算机实现。问题分析和解决思路必须在编写代码之前确定，这也是真正重要的、十分有价值的一个环节，读者应在后面

的学习过程中牢记这一点。熟练掌握一门编程语言的语法及编程技巧固然重要，但是对于问题的分析和解决方案的设计也同样重要，特别是如何把一些实际问题转换为计算机所能解决的问题，这种转换能力也就是如今经常提到的“计算思维”。

在编写程序的过程中，我们可以获得很多快乐。编写程序还是一项具有创造性、可以带来价值的活动，它不仅能解决实实在在的问题，而且为你提供了表现自我的平台。

1.3 Python 语言简介

Python 是一门跨平台、开源、免费的解释型高级动态编程语言。Python 作为动态语言更适合编程初学者，Python 可以让初学者把精力集中在编程对象和思维方法上，而不用担心语法、类型等外在因素。Python 易于学习，应用广泛，拥有大量的扩展库，可以高效地开发各种应用程序，是最受欢迎的程序设计语言之一。

1.3.1 Python 语言的发展及现状

Python 语言的创始人吉多·范罗苏姆(Guido van Rossum)出生于荷兰，是一名计算机程序员，后来成为 Python 语言的最初设计者及主要架构师。

1989 年圣诞节期间，在阿姆斯特丹，吉多为了打发圣诞节的无趣，决心开发一种新的脚本解释程序，作为 ABC 语言的改进版本。之所以选用 Python(蟒蛇的意思)作为该编程语言的名字，是因为吉多是一支名为 Monty Python 的喜剧团体的爱好者。吉多希望 Python 既能够像 C 语言那样全面调用计算机的功能接口，又可以像 shell 那样轻松地进行编程。

1991 年，第一个 Python 编译器(同时也是解释器)诞生了。它是用 C 语言实现的，能够调用 C 库，并且已经具备了类(class)、函数(function)、异常处理(exception)，此外还包括列表(list)和字典(dictionary)等数据类型以及以模块(module)为基础的扩展系统。

2000 年 10 月，Python 2.0 版本发布，这标志着 Python 完成了自身涅槃，开启了 Python 广泛应用的新时代。2010 年，Python 2.x 系列发布了最后一个版本，版本号为 2.7，用于终结 Python 2.x 系列版本的发展，并且不再进行更新。

2008 年 12 月，Python 3.0 版本发布，这个版本的解释器在内部完全采用面向对象的方式来实现，并且在语法层面做了很多重大改进，付出的代价就是 3.x 系列版本的 Python 代码无法向下兼容 2.x 系列版本。

Python 语言经历了一个痛苦但令人期待的版本更迭过程，从 2008 年开始，用 Python 编写的几万个标准库和第三方库开始了版本升级过程，这个过程前后历经 8 年。2016 年，Python 所有重要的标准库和第三方库都已经根据 Python 3.0 版本进行了演进和发展。Python 语言的版本升级过程宣告结束。

提示：

如何直观判断一个 Python 程序是否为 3.x 版本?

最常用的方法是查看 print。Python 3.x 版本用 print()函数替换了 Python 2.x 系列版本中的 print 语句，两者功能一样，但格式不同。

```
>>> print "I Love Python"          # Python 2.x
>>> print("I Love Python")        # Python 3.x
```

随着人工智能和数据挖掘的迅猛发展，Python 语言凭借自身独有的优势，用户使用率呈线性增长。2020 年，Python 赢得了年度 TIOBE 编程语言奖，该奖项用于评选当年最受欢迎的编程语言。Python 语言在 2021 年 8 月实现了 2.17%的正增长，且排名仍在不断上升，如图 1-6 所示。

Aug 2021	Aug 2020	Change	Programming Language	Ratings	Change
1	1		C	12.57%	-4.41%
2	**3**	^	**Python**	**11.86%**	**+2.17%**
3	2	v	Java	10.43%	-4.00%
4	4		C++	7.36%	+0.52%
5	5		C#	5.14%	+0.46%
6	6		Visual Basic	4.67%	+0.01%
7	7		JavaScript	2.95%	+0.07%
8	9	^	PHP	2.19%	-0.05%
9	14	^^	Assembly language	2.03%	+0.99%
10	10		SQL	1.47%	+0.02%

图 1-6　2021 年 8 月的 TIOBE 编程语言排行榜

TIOBE 编程社区索引是编程语言受欢迎程度的风向标，如图 1-7 所示。

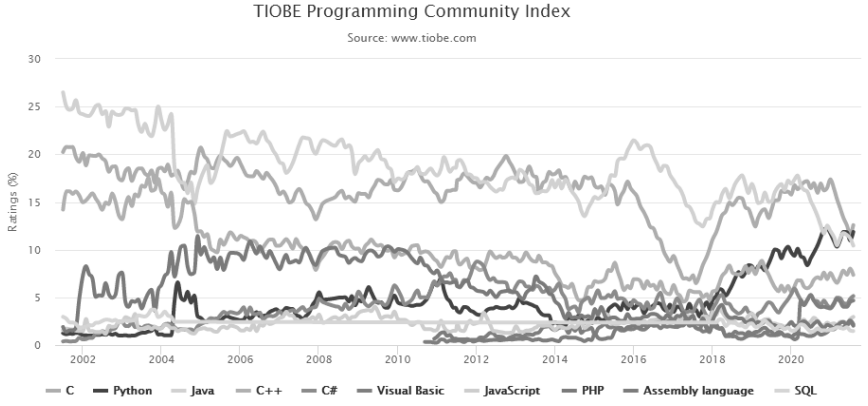


图 1-7　TIOBE 编程社区索引

1.3.2　Python 语言的特点与应用领域

Python 作为一种使用广泛的编程语言，具有下列特点。

- 简单易学：Python 是一种解释型的编程语言，遵循优雅、明确、简单的设计哲学，语法简单，易学、易读、易维护。解释型语言由于需要逐行翻译，因此运行效率不高。
- 易于使用：Python 支持以交互模式运行程序，包含便捷的高级数据类型，并且可以用 C 或 C++语言进行扩展，能快速编写程序并即时满足实际需求。Python 还经常被称为“胶水语言”，能把你用不同语言编写的代码“黏合”在一起。

- 免费和开源：Python 是 FLOSS(自由/开放源码软件)之一，使用者只需要保留版权信息即可任意使用和修改源代码，并将其用于各个领域。
- 可移植性：基于开源本质，Python 已经被移植到许多平台上，包括 Linux/UNIX、Windows、macOS 等。用户编写的 Python 程序，如果未使用依赖于系统的特性，那么无须修改就可以在任何支持 Python 的平台上运行。
- 面向对象：Python 既支持面向过程编程，也支持面向对象编程，Python 还支持继承和重载，这有利于实现代码的复用。
- 资源丰富：Python 语言提供了功能丰富的标准库和第三方库。标准库包括随机库、turtle 库、文档生成、单元测试、数据库、GUI(图形用户界面)等。截至 2021 年 8 月，在 PyPi 网站上，公开发布的第三方库已超过 30 万个，覆盖的信息技术几乎涉及所有领域。

Python 语言具有广泛的应用领域，如图 1-8 所示。下面简单介绍其中常见的几个应用领域。

- 操作系统管理：Python 作为一种解释型的脚本语言，特别适合编写系统管理脚本，使用 Python 编写的系统管理脚本在可读性、源代码复用、扩展性等方面都优于普通的 shell 脚本。
- 科学计算：Python 程序员可以使用 NumPy、SciPy、matplotlib、Pandas 等第三方库编写科学计算程序。众多开源的科学计算软件包均提供了 Python 的调用接口，例如著名的计算机视觉库 OpenCV、三维可视化库 VTK、医学图像处理库 ITK 等。
- Web 开发：Python 语言提供了很多优秀的 Web 开发框架以方便开发者，如常见的 Django、Flask、Pyramid 等。其中，Django 框架的应用范围非常广，学习门槛也很低，能够快速地搭建起可用的 Web 服务。当前，国内外许多知名网站都是使用 Python 语言开发的，如国外的 NASA、CIA、YouTube、Facebook，国内的豆瓣网、知乎网等。
- 图形用户界面(GUI)开发：Python 支持 GUI 开发，使用 Tkinter、wxPython 或 PyQt5 库可以开发跨平台的桌面软件。
- 人工智能：人工智能的迅速发展将深刻改变人类的社会生活，甚至改变世界。Python 语言随此契机迅速发展，并在人工智能领域上坐稳编程语言的第一把交椅，体现了 Python 语言的强势。Python 语言在人工智能领域诞生了很多优秀的第三方库，为人工智能在各个方向的应用提供了极大便利。例如机器学习(TensorFlow、scikit-learn)、自然语言处理(NLTK)、文本处理(python-docx、openpyxl)、人脸识别(OpenCV、PIL)等。
- 游戏开发：在游戏逻辑和功能实现层面，Python 已经成为重要的支撑性语言，如 Pygame、panda3D、cocos2d。网络爬虫是能够自动进行 HTTP 访问并捕获 HTML 页面的程序，Python 语言提供了多个具备网络爬虫功能的第三方库，如 requests、scrapy。

图 1-8　Pyhton 语言的应用领域

1.4 Python 开发环境的安装与配置

在正式开始学习 Python 语言之前，有必要熟悉一下如何安装和配置简单的 Python 语言开发环境。

1.4.1 开发环境的安装

Python 作为一种高级编程语言，计算机是无法直接运行的，而必须使用称为“解释器”的特定程序翻译成机器语言之后才能由计算机执行。Python 解释器可以从 Python 官网下载，如图 1-9 所示。

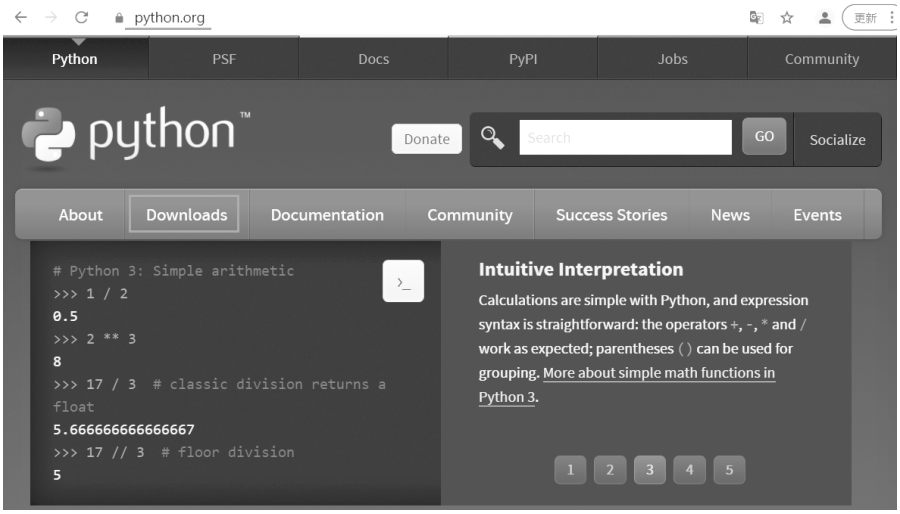


图 1-9　Python 官网

在 Python 官网上，我们可以下载不同版本的 Python 解释器安装程序用于不同的操作系统，如 Windows、Linux、UNIX、macOS 等。

以 64 位的 Windows 10 操作系统和 Python 3.8.7 安装程序为例，下载完成后，双击运行安装程序，在打开的如图 1-10 所示的安装界面中选中 Add Python 3.8 to PATH 复选框。

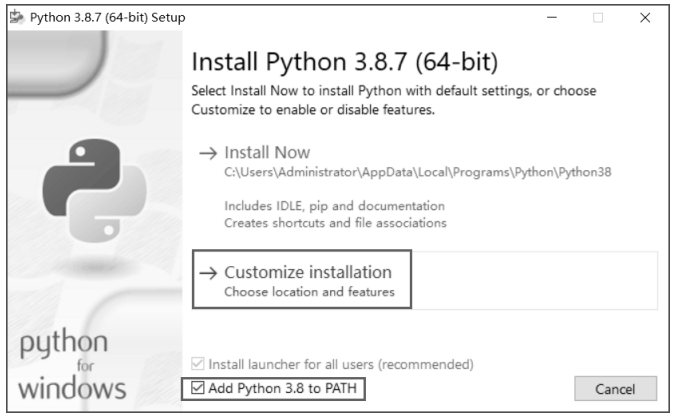


图 1-10　安装界面

第一个安装选项 Install Now 为默认安装，在这种安装方式下，安装组件和安装路径由安装程序自动选择。

第二个安装选项 Customize installation 为用户自定义安装，在这种安装方式下，安装组件和安装路径由用户自己选择。

安装完成后的提示界面如图 1-11 所示。此时，用户的计算机中将会安装与编写和运行 Python 程序相关的若干程序，包括稍后就会用到的 Python 命令行和 Python 集成开发环境(integrated development and learning environment，IDLE)。

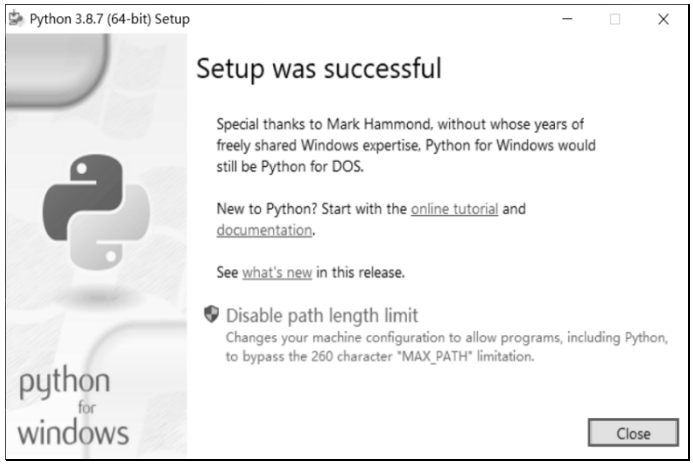


图 1-11　安装完成后的提示界面

注意：Python 3.9 及其以上版本已不再支持 Windows 7。

1.4.2　Python 代码的运行方式

在 Python 集成开发环境中，运行 Python 代码的常用方式有两种：交互方式和文件方式。下面以 Windows 操作系统下的 Python 3.8 为例进行介绍。

1. 交互方式

首先，在 Windows 操作系统的“开始”菜单中找到 Python 3.8 菜单目录并展开，如图 1-12 所示，然后选择 IDLE(Python 3.8 64-bit)选项，就可以打开 IDLE。

图 1-12　“开始”菜单中的 Python 选项

在图 1-13 所示的 IDLE 中，界面上方是 Python 解释器的版本信息，下方是 Python 提示符>>>。

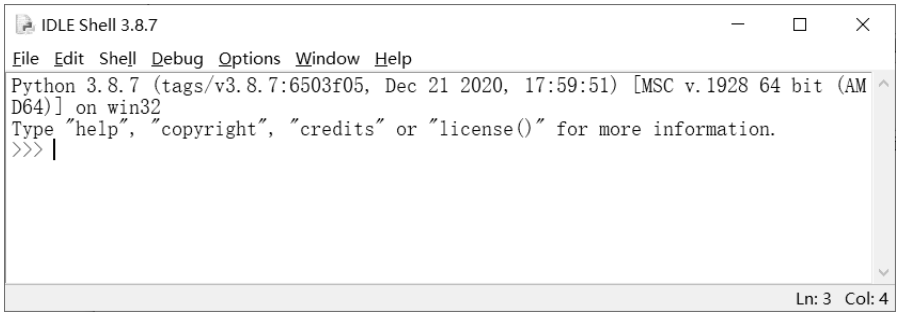


图 1-13　Python 内置的集成开发环境 IDLE

在提示符>>>后尝试输入下列代码，查看是否运行成功。

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

在上述代码中，print()表示将引号内的信息输出到屏幕上，按回车键，将会出现图 1-14 所示的输出。

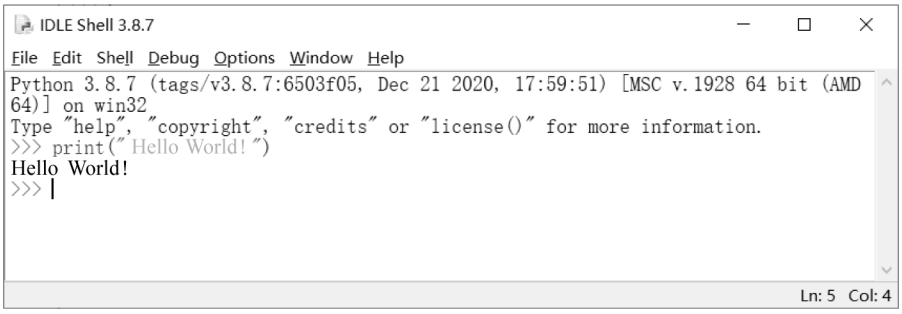


图 1-14　输出结果

Python 语言支持直接使用中文等非西文字符，带中文的 Python 程序的运行结果如图 1-15 所示。

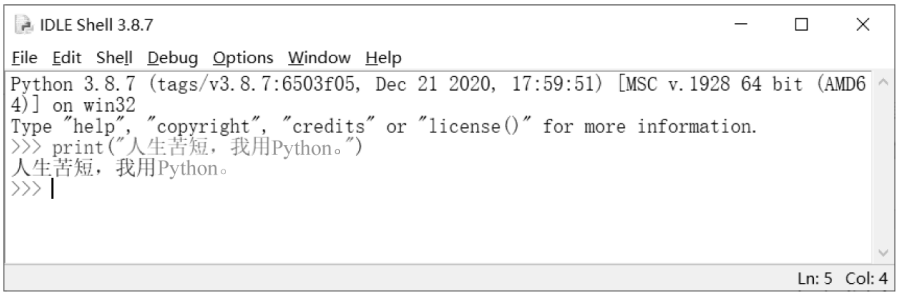


图 1-15　输出中文字符

在提示符>>>后每输入一行代码，都需要按回车键执行代码并给出执行结果，这就是交互的代码执行方式。

2. 文件方式

当需要编写比较复杂的程序时，由于程序中包含多行代码，交互方式对于编写和调试代码十分不便，这时可以采用文件方式来编写代码和执行程序。文件方式是指新建一个扩展名为.py 的文件，然后将程序的所有代码都写在这个文件里，最后由解释器统一执行。

(1) 新建：在 IDLE 的菜单栏中选择 File→New File 选项，如图 1-16 所示，打开图 1-17 所示的文本编辑窗口，输入代码。

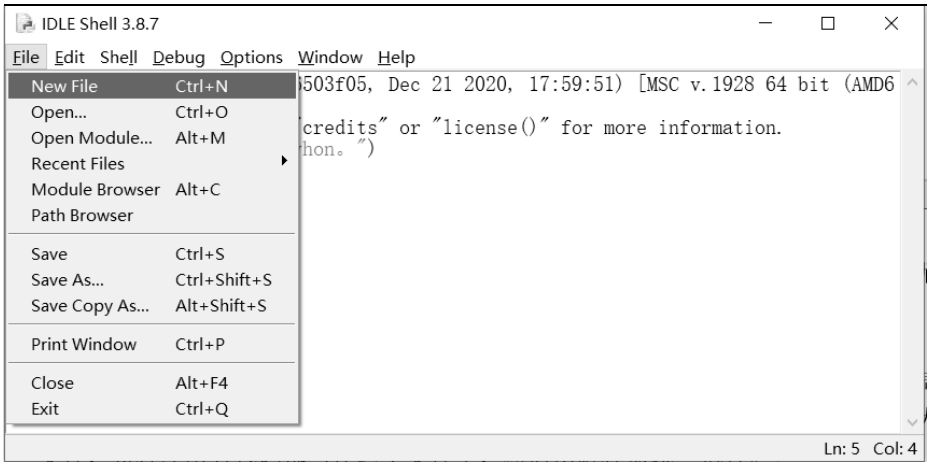


图 1-16　File 菜单

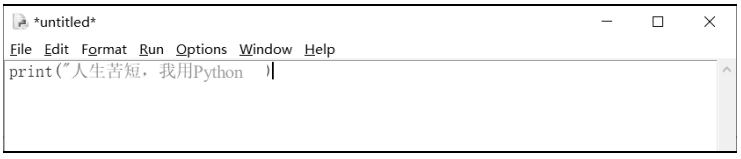


图 1-17　文本编辑窗口

(2) 保存：如图 1-16 所示，选择 File→Save 选项，打开“另存为”对话框，如图 1-18 所示，选择保存路径并设置文件名为 eg1_1.py，单击“保存”按钮。

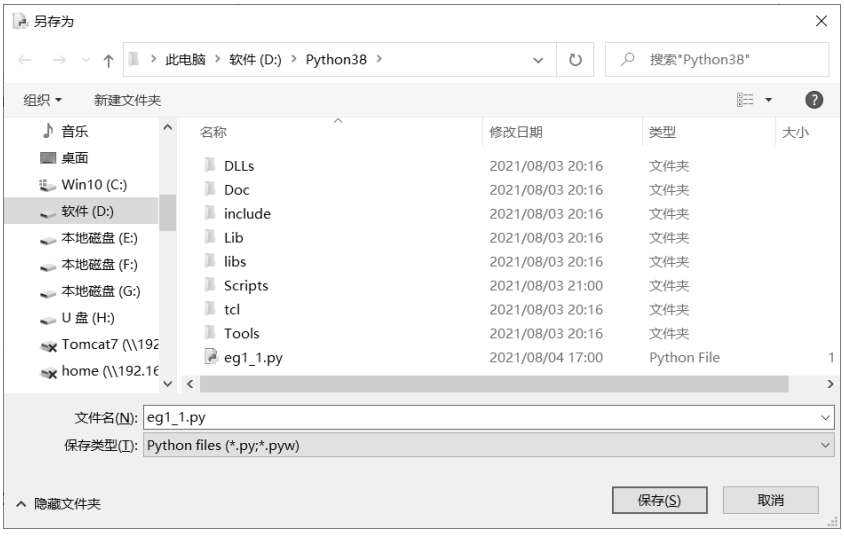


图 1-18　“另存为”对话框

(3) 执行：如图 1-19 所示，选择 Run→Run Module 选项或按 F5 功能键，执行程序，运行结果如图 1-20 所示。

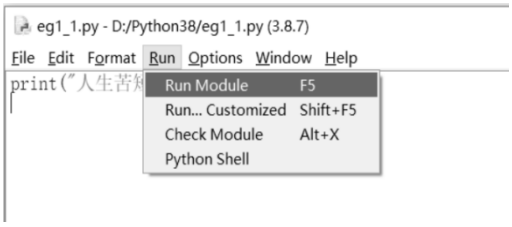


图 1-19　执行文件

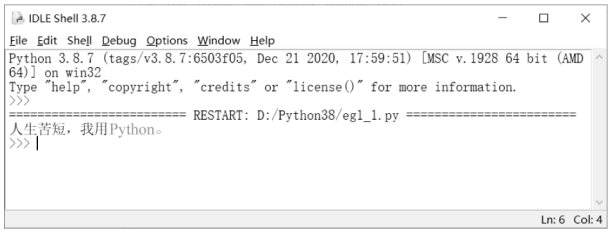


图 1-20　运行结果

此外，也可以通过 Windows 的命令行窗口来执行 Python 程序。例如，包含 Python 程序的文件名为 eg1_1.py，我们首先需要在 eg1_1.py 文件所在文件夹的空白处按住 Shift 键并右击，从弹出的快捷菜单中选择“在此处打开 PowerShell 窗口”选项，然后在打开的命令行窗口中输入 python eg1_1.py 即可执行这个程序，如图 1-21 所示。

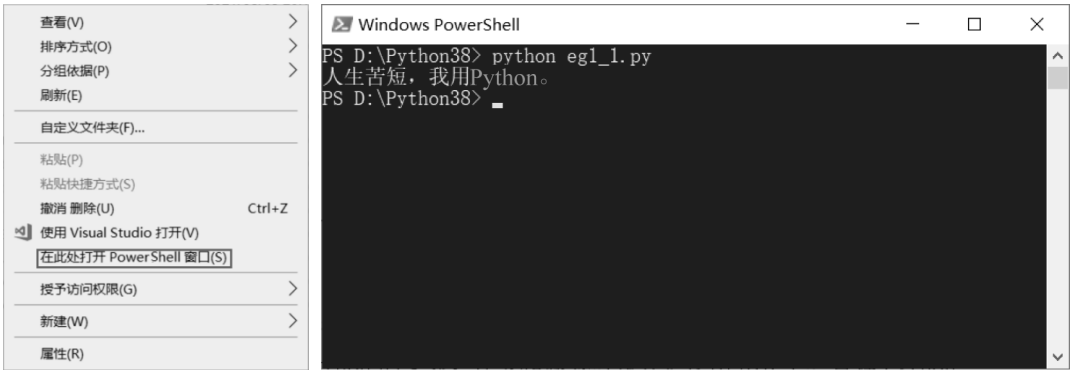


图 1-21　使用 Windows PowerShell 的命令行窗口执行 Python 程序

交互方式和文件方式在本质上是相同的，都是将 Python 代码通过解释器逐行翻译为机器代码并由计算机执行。在学习 Python 语法时，为了了解一些命令的用法，可以选用交互方式；而在编写较复杂的程序时，则优先选用文件方式来编写、调试及运行代码。

1.5　Python 语言的编码规范

“没有规矩，不成方圆。”任何一种编程语言都有自己的约定，该约定又称为编码规范。Python 社区对代码编写有一些共同的要求和规范，最好在开始编写第一段代码时就遵循这些要求和规范，养成良好习惯，只有这样，我们才能具有编写优雅代码的功底和能力。

1. 缩进

Python 程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示对应的代码块也结束了。在函数定义、类定义、选择结构、循环结构、异常处理和 with 语句等结构中，行尾的冒号表示缩进的开始。同一级别代码块的缩进量必须相同，通常以 4 个空格为基本缩进单位。

例如，以下循环语句用于输出 0～9 的 2 次幂，分隔符为空格，代码如图 1-22 所示。

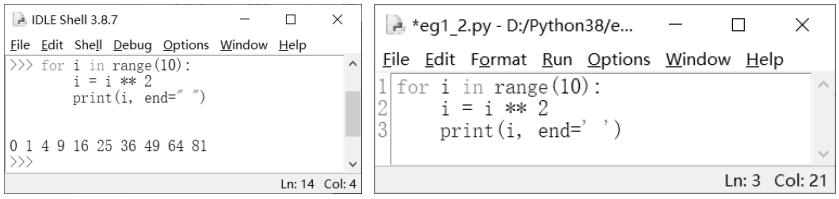


图 1-22　循环结构中的缩进

2. 注释

可读性强的程序一般都会包含 20%以上的注释。常用的注释方法主要有以下两种。

1) 以#开始，表示本行中#之后的内容为注释，主要用于单行注释。

```
# 循环输出 0～9 的 2 次幂
for i in range(10):
    i = i ** 2          # 幂运算
    print(i, end=' ')   # 打印 i 的值
```

2) 包含在一对三引号'''或"""之间且不属于任何语句的内容将被解释器识别为注释，主要用于多行注释。

```
'''
程序功能：循环输出 0～9 的 2 次幂
运行结果：
0 1 4 9 16 25 36 49 64 81
'''
for i in range(10):
    i = i ** 2          # 幂运算
    print(i, end=' ')   # 打印 i 的值
```

除了缩进和注释，读者还需要注意以下 Python 编码规范。

- Python 区分大小写，Num、num、NUM 是完全不同的三个名称。
- Python 语言中的所有语法符号，如冒号、单引号、双引号和圆括号等，都必须在英文输入法下输入，字符串中的符号除外。
- 一行语句如果太长，可以在尾部添加反斜杠“\”来通过换行分成多行，但是建议使用括号来包含多行内容，如图 1-23 所示。
- 最好在函数定义和一段完整的功能代码之后增加两个空行，而在运算符的两侧各增加一个空格，并在逗号的后面增加一个空格。

```
x = 1 + 2 + 3\
    + 4 + 5\
    + 6

y = (1 + 2 + 3
     + 4 + 5
     + 6)
```

图 1-23　使用反斜杠和括号

1.6 第三方库的安装

pip 工具已经成为管理 Python 第三方库的主要方式，pip 工具不仅支持查看本机上已安装的 Python 第三方库列表，而且支持 Python 的安装、升级和卸载等操作。使用 pip 工具管理 Python 第三方库时，只需要保证计算机联网，然后在命令行窗口中输入几个命令即可，这极大方便了用户。常用 pip 命令的使用方法如表 1-1 所示。

表 1-1　常用 pip 命令的使用方法

pip 命令示例	说明
pip list	显示当前已安装的所有模块
pip install SomePackage[==version]	在线安装 SomePackage 模块的指定版本
pip install --upgrade SomePackage	升级 SomePackage 模块
pip uninstall SomePackage[==version]	卸载 SomePackage 模块

在 https://pypi.org/网站上可以获得 Python 第三方库的综合列表。用户既可以根据需要下载源码并进行安装，也可以使用 pip 工具进行在线安装，此外，另有一些扩展库提供了.whl 文件和.exe 文件，从而大幅简化了扩展库的安装过程。

例如：安装科学计算用的扩展库 NumPy，如图 1-24 所示。

```
C:\Users\Administrator>pip install numpy  安装numpy第三方库命令
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting numpy
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/0d/fc/2a55c4d690437e09e44f40fe7a8458a72aef9da01719
bd8746002999f782/numpy-1.21.1-cp38-cp38-win_amd64.whl (14.0 MB)
     |                                | 14.0 MB 6.8 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.21.1   安装成功信息

C:\Users\Administrator>
```

图 1-24　安装扩展库

再如：显示已安装的所有扩展库，如图 1-25 所示。

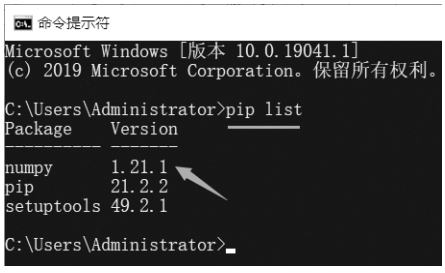


图 1-25　显示已安装的所有扩展库

提示：

当安装或升级扩展库时，超时了怎么办？可以选择国内的镜像源进行安装，国内的镜像源如表 1-2 所示。

表 1-2　国内的镜像源

镜像名称	镜像地址
阿里云	https://mirrors.aliyun.com/pypi/simple/
中国科学技术大学	https://mirrors.ustc.edu.cn/pypi/simple/
清华大学	https://pypi.tuna.tsinghua.edu.cn/simple/

以清华大学的镜像源为例，设置方法如下：

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple xxxxxxx          # 临时使用
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple  # 永久设置
```

1.7 扩展库的导入与使用

Python 在默认安装时仅包含基本或核心模块，启动时也仅加载基本模块。我们可以在需要时才显式地导入和加载标准库及第三方库(须正确安装)，这样除了能够减小程序运行的压力，还能够使程序具有很强的可扩展性。

Python 库有三种，分别是标准库、扩展库和用户自定义库。通常情况下，可以首先使用 import 命令按需将标准库、扩展库、用户自定义库导入，然后即可使用模块中的方法。导入方式有如下三种。

1. import 模块名 [as 别名]

使用这种方式导入以后，方法在使用时，需要在方法名前加上模块名作为前缀，也就是必须以“模块名.方法名()”的形式进行访问。如果模块名比较长的话，可以为导入的模块设置别名，然后使用“别名.方法名()”的形式调用方法。

```
>>> import random                # 导入 random 标准库
>>> random.randint(10, 20)       # 随机产生一个取值区间为[10,20]的整数
15
>>> import random as r           # 导入 random 标准库，别名为 r
>>> r.uniform(0,1)               # 随机产生一个取值区间为[0,1)的小数
0.8147119193923934
>>>
```

2. from 模块名 import 方法名 [as 别名]

使用这种导入方式只能导入指定的方法，但可以为导入的方法设置别名。这种导入方式不仅可以减少查询次数、提高访问速度，而且可以减少程序员需要导入的代码量，此处还不需要使用模块名作为前缀。

```
>>> from math import sin
>>> sin(1)
0.8414709848078965
>>> from math import sin, pi
>>> sin(pi/2)
1.0
>>> from math import sin as s
>>> s(1)
0.8414709848078965
```

3. from 模块名 import *

使用这种导入方式可以一次性导入模块中通过__all__变量指定的所有方法。

```
>>> from math import *           # 导入 math 标准库中的所有方法
>>> gcd(36,18)                   # 最大公约数
18
```

最后一种导入方式简单粗暴，写起来比较省事，虽然适合初学者使用，但我们并不推荐，原因是这种导入方式会降低代码的可读性，增加代码的运行时间，我们将很难区分自定义函数和那些从模块中导入的函数，这会导致命名空间的混乱，不利于代码的理解和维护。

1.8 习题

一、选择题

1. 关于 Python 语言的特点，以下选项中描述错误的是(　　)。
   A. Python 语言是非开源语言　　B. Python 语言是跨平台语言
   C. Python 语言是解释型语言　　D. Python 语言是脚本语言
2. 关于 import 引用，以下选项中描述错误的是(　　)。
   A. 可以使用 import turtle 导入 turtle 库
   B. 可以使用 from turtle import setup 导入 turtle 库
   C. 可以使用 import turtle as t 导入 turtle 库并取别名为 t
   D. import 保留字用于导入模块或模块中的对象
3. IDLE 环境的退出命令是(　　)。
   A. esc()　　B. close()　　C. 回车键　　D. exit()
4. 关于 Python 语言的编码规范，以下选项中描述错误的是(　　)。
   A. Python 允许把多条语句写在同一行
   B. 在 Python 语句中，增加缩进表示语句块的开始，减少缩进表示语句块的结束
   C. Python 可以将一条长语句分成多行来显示，方法是使用反斜杠“\”
   D. Python 不允许把多条语句写在同一行
5. Python 语言属于(　　)。
   A. 机器语言　　B. 汇编语言　　C. 高级语言　　D. 以上都不是
6. Python 内置的集成开发环境是(　　)。
   A. PythonWin　　B. Pydev　　C. IDE　　D. IDLE
7. Python 解释器的提示符是(　　)。
   A. >　　B. >>　　C. >>>　　D. #
8. Python 官方的扩展索引库是(　　)，所有人都可以下载第三方库或上传自己开发的库到其中。
   A. PyPI　　B. PyPy　　C. pip　　D. PyPe
9. 在 Python 解释器环境中，用于表示上一次运算结果的特殊变量为(　　)。
   A. :　　B. _　　C. >　　D. #
10. 关于 Python 程序的缩进格式，以下选项中描述错误的是(　　)。
   A. 不需要缩进的代码顶行写，前面不能留空白
   B. 缩进既可以用 Tab 键来实现，也可以用多个空格来实现
   C. 严格的缩进可以约束程序结构，Python 允许多层缩进
   D. 缩进是用来美化 Python 程序的

二、判断题

1. 我们在 Windows 平台上编写的 Python 程序无法在 UNIX 平台上运行。 ( )

2. Python 程序只能使用源代码来运行，不能打包成可执行文件。 ( )

3. 对于 Python 代码来说，缩进是硬性要求，如果缩进错了，就可能导致程序无法运行或运行结果错误。 ( )

4. pip 命令支持使用扩展名为.whl 的文件直接安装 Python 扩展库。 ( )

5. Python 扩展库需要导入以后才能使用其中的对象，而 Python 标准库不需要导入即可使用其中所有的对象和方法。 ( )

6. Python 使用缩进来体现代码之间的逻辑关系。 ( )

7. Python 中的代码注释只有一种方式，那就是使用#符号。 ( )

8. 放在一对三引号之间的任何内容都将被认为是注释。 ( )

9. 为了让代码更加紧凑，在编写 Python 程序时应尽量避免加入空格和空行。 ( )

10. 如果只需要 math 模块中的 sin()函数，建议使用 from math import sin 进行导入，而不要使用 import math 导入整个模块。 ( )

三、填空题

1. 计算机程序设计语言分为三大类，分别是_____、_____、_____。

2. Python 集成开发环境 IDLE 支持两种运行方式：一种是_____，另一种是_____。

3. 在 IDLE 的 shell 环境下，输入_____后，按回车键即可进入交互式帮助系统。

4. 用于显示已安装的所有扩展库的 pip 命令是_____。

5. 使用 Python 语言内置的输入函数_____和输出函数 print()可以实现程序和用户的交互。

6. 下列代码中 print()函数的输出结果是_____。

```
a = 1
b = a
a = 2
print(b)
```

四、操作题

1. 下载并安装 Python 3.8.7。

2. 下载并安装 Anaconda 3 个人版或 PyCharm 社区版，任选一个。

3. 安装扩展库 NumPy、Pandas、matplotlib、openpyxl、jieba、python-docx。如果已经安装，可尝试进行升级。

第 2 章

基本数据类型、运算符与表达式

学习目标

- 掌握 Python 语言的基本数据类型。
- 掌握 Python 语言的保留字以及变量的命名和赋值。
- 掌握 Python 语言的基本输入输出函数。
- 掌握 Python 表达式的运算次序并能够计算出结果。

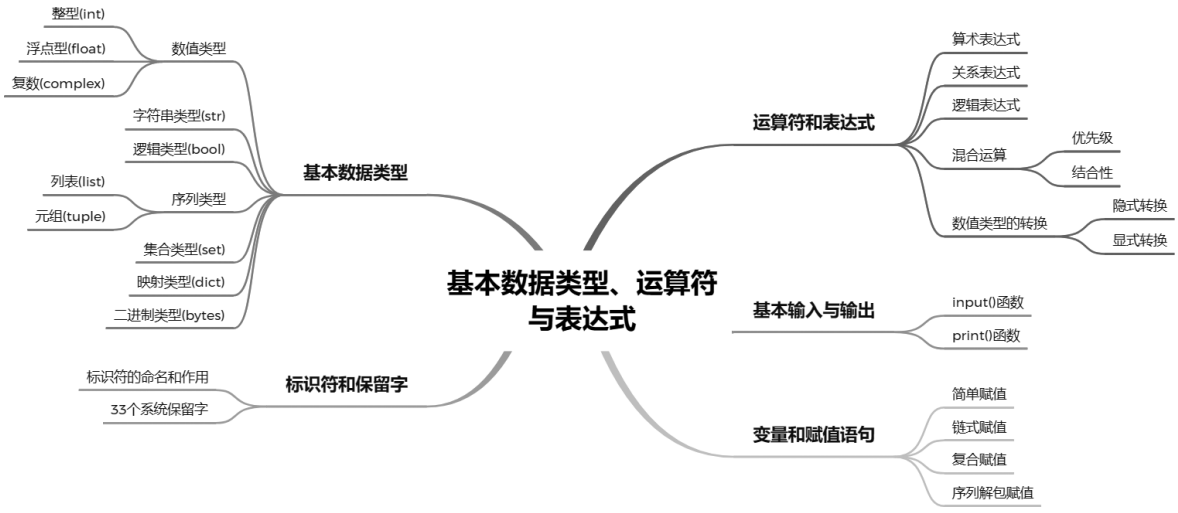
学习重点

掌握标识符的命名规则，掌握变量的命名并能够使用不同的赋值方法对变量进行赋值，掌握输入输出函数的格式及用法，掌握运算符的优先级和结合性。

学习难点

整数与浮点数，输入输出函数，混合运算。

知识导图



2.1 引例

本节将通过讨论一个温度转换的问题，简单介绍程序设计的基本过程，并编写一个 Python 程序。

在温度的表述上，世界上绝大多数国家都使用国际公制单位——摄氏度(℃)来标识温度。摄氏度是由瑞典天文学家安德斯·摄尔修斯于 1742 年提出的，其后历经改进。摄氏度的含义是指在 1 标准大气压下，纯净的冰水混合物的温度为 0 摄氏度，水的沸点为 100 摄氏度，中间 100 等分，每等分为 1 摄氏度。温度体系的另外一个单位是华氏度(℉)，使用华氏度的国家较少，其中大家比较熟知的国家是美国。华氏度也是温度的一种度量单位，以其发明者德国人华伦海特命名。华氏度的含义是指在 1 标准大气压下，冰水混合物的温度为 32 华氏度，水的沸点为 212 华氏度，中间 180 等分，每等分为 1 华氏度。

1. 分析需求

中国人如果去美国工作或旅游，那就需要把当地使用的华氏度转换成自己熟悉的摄氏度；而美国人如果来中国工作或旅游，就需要把摄氏度转换成自己熟悉的华氏度。温度转换有很多种方法，我们这里讨论利用程序进行温度转换，用户输入温度值，程序则进行温度转换并输出。这里需要注意的是，如果输入的是摄氏度，那么程序需要转换为华氏度输出；如果输入的是华氏度，那么程序需要转换成摄氏度输出。

2. 设计算法

输入：用户需要输入一个数字形式的温度值，另外在输入时还要体现这个数字是摄氏度还是华氏度，为此，可在数字的前面加上一个字母标识。例如，用 C24 表示 24 摄氏度，用 F72 表示 72 华氏度。也就是说，每次输入的内容中既有温度体系标识，也有需要转换的温度值。

计算：根据输入的温度体系标识，也就是输入内容中的首字符，决定进行何种温度转换。如果首字符是 C，则使用公式 F = C×1.8 + 32 计算得到华氏度；如果首字符是 F，则使用公式 C = (F – 32) / 1.8 计算得到摄氏度。

输出：输出温度体系标识和温度值。

3. 编写程序

根据上述算法，编写 Python 程序代码。

```
1   # eg02-01.py
2   # 温度转换程序
3   temperature = input('请输入温度体系标识和温度值：')    # 输入要转换的温度
4   if temperature[0] in ['c', 'C']:                      # 判断首字符是否为字母 C 或 c
5       f = eval(temperature[1:]) * 1.8 + 32              # 计算华氏度
6       print("华氏温度为：{:.2f}".format(f))              # 输出华氏度
7   elif temperature[0] in ['f', 'F']:                    # 判断首字符是否为字母 F 或 f
8       c = ((eval(temperature[1:]) - 32) / 1.8)          # 计算摄氏度
9       print("摄氏温度为：{:.2f}".format(c))              # 输出摄氏度
10  else:                                                 # 当首字符不是上述 4 个字母时
11      print("输入错误！请重新运行。")                    # 提示错误信息
```

4. 输入与编辑程序

启动 IDLE 后，在菜单栏中选择 File→New File 选项，在弹出的文本编辑窗口中输入以上程序，保存为 eg02-01.py。

5. 运行与调试程序

运行时输入带摄氏温度体系标识的温度值，运行结果如下：

```
请输入温度体系标识和温度值：C24          # 输入值
华氏温度为：75.20                         # 运行结果
```

运行时输入带华氏温度体系标识的温度值，运行结果如下：

```
请输入温度体系标识和温度值：f98
摄氏温度为：36.67
```

运行时输入错误标识的温度值，运行结果如下：

```
请输入温度体系标识和温度值：a100
输入错误！请重新运行。
```

【思考题】假设 1 人民币=16.9 日元、1 日元=0.059 人民币，尝试编写人民币和日元兑换的 Python 程序。

2.2 基本数据类型

计算机所能处理的远不止数值，计算机还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据需要定义成不同的数据类型。在 Python 中，能够直接处理的数据类型如表 2-1 所示。

表 2-1　Python 内置的数据类型

数值类型	int、float、complex
字符串类型	str
逻辑类型	bool
序列类型	list、tuple、range
集合类型	set、frozenset
字典类型	dict
二进制类型	bytes、bytearray、memoryview

2.2.1 数值类型

数值类型用于存储数值。Python 支持三种不同的数值类型：整型(int)、浮点型(float)和复数类型(complex)。

1. 整数

整数就是不带小数点的数字，Python 中的整数包括正整数、0 和负整数。Python 整数的取值范围可以说是无限的(仅受限于运行 Python 的计算机的硬件)，不管多大或多小的数字，Python 都能轻松处理。当所用数值超出计算机自身的能力限制时，Python 会自动转用高精度计算(极大数运算)。相对于大多数高级语言，Python 语言对于计算极大数非常方便，而且精度高。下面我们使用 pow()函数来计算和测试一下整数的取值范围。

```
>>> pow(2, 64)                          # 计算 2 的 64 次方
18446744073709551616
>>> pow(2, 1000)                        # 计算 2 的 1000 次方
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788
15695858127594672917553146825187145285692314043598457757469857480393456777482423098542107460506
23711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243886837205668069376
```

在 Python 中，可以使用以下进制来表示整数。

1) 十进制形式

我们平时所见的整数就是十进制形式，十进制整数由 0～9 十个数字组成，例如 789、−35 。

2) 二进制形式

由 0 和 1 两个数字组成，书写时以 0b 或 0B 开头，例如 0b1001、−0B111。

3) 八进制形式

八进制整数由 0～7 八个数字组成，以 0o 或 0O 开头。注意，第一个符号是数字 0，第二个符号是大写或小写的字母 O，例如 0o61、－0O25。

4) 十六进制形式

由 0～9 十个数字和 A～F(或 a～f)六个字母组成，书写时以 0x 或 0X 开头，例如 0x2f、0X2E。

数字分隔符：当数值很大时，为了提高数字的可读性，Python 允许使用下画线作为数字分隔符。数字分隔符既可用于整数中，也可用于浮点数中。通常的形式为，每隔三个数字添加一个下画线，类似于数字中的逗号。下画线不影响数字本身值的大小。

```
>>> print("地球到月球的平均距离：",  384_401_000)
地球到月球的平均距离：384401000
```

2. 浮点数

浮点数与数学中的实数基本类似，它们都用来表示带有小数的数值。Python 中的浮点数必须带有小数部分，小数部分可以是 0。浮点数有两种表示形式：十进制形式和指数形式。

1) 十进制形式

十进制形式就是我们平时看到的小数形式，例如 12.3、123.0、0.123、.3。

注意：

十进制形式的浮点数在书写时必须包含小数点，以 123.0 为例，如果写成 123，这个数字就会被 Python 当作整数处理。

2) 指数形式

指数形式的写法为 *a*E*b* 或 *a*e*b*。

a 为尾数部分，书写为十进制形式；*b* 为指数部分，书写为十进制整数；E 或 e 是固定的字符，用于分隔尾数部分和指数部分。整个表达式等效于 $a\times10^{b}$。例如：3.84E8 = 3.84×10^{8}，其中 3.84 是尾数，8 是指数。一个数字只要能写成指数形式，它就是浮点数，即使它的最终值看起来像整数。例如，2E3 等效于 2000.0，因此 2E3 是一个浮点数。

Python 中的浮点数需要使用 8 字节的存储空间，其中 52 位用来存储浮点数的有效数字，11 位用来存储指数，1 位用来存储正负号。浮点数约有 16 位有效数字，可使用以下代码来验证浮点数的取值范围：

```
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308,
               min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53,
               psilon=2.220446049250313e-16, radix=2, rounds=1)
```

浮点数看似有效数字很多，但二进制与十进制的转换往往存在一些误差，因此浮点数之间应尽量避免直接进行相等比较，一般以差的绝对值足够小来比较浮点数是否相等。

```
>>> (0.1 + 0.6) == 0.7                    # 比较结果为 True
True
>>> (0.1 + 0.7) == 0.8                    # 因误差导致比较结果为 False
False
>>> 0.1 + 0.7
0.7999999999999999
```

3. 复数

复数是 Python 内置类型，直接书写即可使用。换句话说，Python 语言本身就支持复数，而不需要依赖于标准库或第三方库。

复数由实部(real)和虚部(imag)构成。在 Python 中，复数的虚部以 j 或 J 作为后缀，具体格式为：*a* + *b*j。*a* 表示实部，*b* 表示虚部。例如，5 + 0.6j 就是一个复数。

2.2.2　字符串类型

若干字符的集合就是字符串。Python 中的字符串必须由单引号(')、双引号(")、三引号('''或""")包围，字符串的开头和结尾必须使用相同类型的引号。

1. 使用单引号和双引号

格式为："字符串内容"或'字符串内容'。

字符串的内容可以包含英文字母、标点、特殊符号、中文等计算机系统支持的所有文字，例如"12345"、'123abc'、"佳木斯大学"。

Python 中的字符串是由 Unicode 字符组成的序列。因此，字符串中的中文字符和英文字符都算 1 个字符。字符串有两种取值顺序：从左至右索引，默认从 0 开始，最大值是“字符串长度减 1”；从右至左索引，默认从-1 开始，逐个递减到负的字符串长度。如果需要从字符串中

获得子字符串，可以使用 [头下标:尾下标]的格式来截取相应的字符串。其中下标从 0 起始，可以使用正数和负数；下标还可以为空，表示取到头或尾。

```
s = 'abcdefg'
>>> print( s )                  # 输出整个字符串
abcdefg
>>> print( s[0] )               # 输出字符串中从左边开始的第 1 个字符
a
>>> print( s[1:3] )             # 输出字符串中从左边开始的第 2 和第 3 个字符
bc
>>> print( s[2:] )              # 输出字符串中从左边第 3 个字符开始到末尾的所有字符
cdefg
>>> print( s[-5:-3] )           # 输出字符串中从右边开始的第 4 和第 5 个字符
cd
>>> print( s * 2 )              # 重复输出字符串两次
abcdefgabcdefg
```

对于 Python 字符串而言，使用单引号和双引号没有区别。当字符串中出现引号时，我们需要进行特殊处理，否则 Python 会解析出错。以字符串'I'm all ears.'为例，由于其中包含了单引号，因此 Python 会将字符串中的单引号与第一个单引号配对，这样就会把'I'当成字符串，而后面的 m all ears.'就变成了多余的内容，从而导致语法错误。

```
>>> print( 'I'm all ears' )      # 执行后，系统提示有语法错误
SyntaxError: invalid syntax
```

该问题的解决方法有两种。

1) 对引号进行转义

只需要在引号的前面添加反斜杠“\”就可以对其进行转义，让 Python 把它当作普通文本对待。

```
>>> str1 = 'I\'m all ears.'                          # 将第二个单引号转义为普通字符
>>> str2 = "英文双引号是\"，中文双引号是“"          # 将第二个双引号转义为普通字符
>>> print( str1 )
I'm all ears.
>>> print( str2 )
英文双引号是"，中文双引号是“
```

2) 使用不同的引号包围字符串

如果字符串中出现了单引号，那么可以使用双引号包围字符串，反之亦然。

```
>>> str1 = "I'm all ears."                           # 字符串中出现单引号，用双引号包围字符串
>>> str2 = '英文双引号是"，中文双引号是“'           # 字符串中出现双引号，用单引号包围字符串
>>> print( str1 )
I'm all ears.
>>> print( str2 )
英文双引号是"，中文双引号是“
```

2. 使用三引号

格式为：'''字符串内容'''或"""字符串内容"""。

Python 语言中的三引号通常用来处理长字符串，比如当程序中有大段文本内容需要定义成字符串时，此外也可用于对多行内容进行注释。例如：

```
>>> longstr = '''Jiamusi University, the cradle of talents in the east pole of China, located in the
earliest zone to welcome sunrise in China --Jiamusi city of Heilongjiang Province,
which is also the best ecological environment charming city in China.'''
>>> print(longstr)
Jiamusi University, the cradle of talents in the east pole of China, located in the
earliest zone to welcome sunrise in China --Jiamusi city of Heilongjiang Province,
which is also the best ecological environment charming city in China.
```

2.2.3　逻辑类型

Python 语言提供了逻辑(bool)类型来表示真(对)或假(错)，比如表达式 3>2，这个表达式是正确的，在计算机程序中称为真(对)，Python 语言使用 True 来表示；再比如表达式 2>3，这个表达式是错误的，在计算机程序中称为假(错)，Python 语言使用 False 来表示。True 和 False 都是 Python 语言中的关键字，输入时一定要注意字母的大小写，否则解释器会报错。

```
>>> 3 > 2              # 表达式的结果为真
True
>>> 2 > 3              # 表达式的结果为假
False
```

2.2.4　其他常用数据类型

1. 列表

在列表中，元素的数据类型可以不相同，甚至可以包含列表。

列表的格式为：[元素 1，元素 2，…]。

和字符串一样，列表也是有序的，也可以被截取。列表在被截取后，返回的将是一个包含所截取元素的新列表，列表截取方法和字符串截取方法相同。与字符串不同的是，列表中的元素是可以改变的。

```
>>> listone=[36, 3.14, 'xyz', "hello", 98.2]
>>> listtwo=['大学英语', '高等数学', '大学计算机']
>>> print(listone)                          # 输出 listone 的所有元素
[36, 3.14, 'xyz', 'hello', 98.2]
>>> print(listone[0])                       # 输出 listone 左边的第 1 个元素
36
>>> print(listone[2:4])                     # 输出 listone 左边的第 3 和第 4 个元素
['xyz', 'hello']
>>> print(listone[3:])                      # 输出从 listone 左边的第 4 个元素开始到末尾的所有元素
['hello', 98.2]
>>> print(listtwo * 2)                      # 输出 listtwo 的所有元素两次
['大学英语', '高等数学', '大学计算机', '大学英语', '高等数学', '大学计算机']
>>> print(listone + listtwo)                # 输出 listone 和 listtwo 的所有元素
[36, 3.14, 'xyz', 'hello', 98.2, '大学英语', '高等数学', '大学计算机']
```

2. 元组

元组与列表类似，元组中元素的数据类型可以不相同，元组也是有序的，也可以被截取。不同之处在于，元组中的元素不能修改。

元组的格式为：(元素 1，元素 2，…)。

```
>>> tupleone=(36, 3.14, 'xyz', "hello", 98.2)
>>> tupletwo=('大学英语', '高等数学', '大学计算机')
>>> print(tupleone)
(36, 3.14, 'xyz', 'hello', 98.2)
>>> print(tupleone[0])
36
>>> print(tupleone[2:4])
('xyz', 'hello')
>>> print(tupleone[3:])
('hello', 98.2)
>>> print(tupletwo * 2)
('大学英语', '高等数学', '大学计算机', '大学英语', '高等数学', '大学计算机')
>>> print(tupleone + tupletwo)
(36, 3.14, 'xyz', 'hello', 98.2, '大学英语', '高等数学', '大学计算机')
```

3. 集合

Python 中的集合与数学中集合的概念类似，它们都是零个或多个数据项的无索引形式的无序组合。集合中的元素不可重复，并且数据类型不能是可变的。

集合的格式为：{元素 1,元素 2,…}。

可以使用大括号{元素 1,元素 2,…}或 set()函数创建集合。

```
thisset={'baidu', 'tencent', 'alibaba'}
>>> print(thisset)
{'alibaba', 'tencent', 'baidu'}
```

4. 字典

字典是一种映射类型，使用一对大括号{ }标识，元素的形式是“键-值”对，键必须使用不可变数据类型，可以通过引用键名来访问元素，字典中的元素是无序的。

字典的格式为：{键:值, 键:值, 键:值,…}。

可以使用 dict()函数创建字典，也可以使用{}创建空字典。

```
>>> scores = {'高数': 89, '英语': 84, '计算机': 91}
>>> print( scores )
{'高数': 89, '英语': 84, '计算机': 91}
```

注意：

在同一个字典中，键必须是唯一的。

5. 字节符

字节符以二进制形式来存储数据，和字符串类似，字节符也使用单引号、双引号或三引号作为定界符。如果字符串的内容都是 ASCII 字符，那么直接在字符串的前面添加前缀 b 就可

以将其转换成字节符。

```
>>> print( b'hello!' )
b'hello!'
```

2.3 标识符和保留字

2.3.1 标识符

在 Python 中，标识符的主要作用就是作为变量、函数、类、模块以及其他对象的名称。标识符的命名必须遵循以下规则：

- 标识符由大小写字母、下画线、数字和汉字组成，但首字符不能是数字，长度没有限制。
- 标识符不能和 Python 中的保留字相同。

例如，合法的标识符有 name、UserID、x123、python_good，不合法的标识符如下。

```
$money      # $不在指定的字符范围内
5pen        #  不能以数字开头
is          # is 是保留字，不能作为标识符
```

在 Python 语言中，标识符中的字母是严格区分大小写的，NAME、Name 和 name 是三个不同的标识符，它们之间没有任何关系。

另外，以下画线开头的标识符都有特殊含义。例如，以单下画线开头的标识符表示不能直接访问的类的属性，以双下画线开头的标识符表示类的私有成员，以双下画线开头且结尾的标识符是专用标识符。因此，读者应避免定义以下画线开头的标识符。

2.3.2 保留字

保留字是 Python 语言内部定义的标识符，它们具有特定的含义，因此不能作为变量名、函数名或任何其他标识符使用。

Python 从 3.10 版本开始有 35 个保留字。表 2-2 列出了 35 个系统保留字。这些保留字分别用于控制程序流转、定义函数、处理异常等功能。保留字不能由程序员自定义，否则会导致语法错误。

在编写 Python 代码时，应避免使用保留字作为标识符。如果你尝试使用保留字命名变量或函数，Python 解释器将会抛出语法错误。为了避免冲突，要使用有意义且不与保留字重复的名称作为变量、函数、类等的名称。

表 2-2　Python 中的 35 个系统保留字

and	elif	is	with
as	else	lambda	yield
assert	except	nonlocal	False
async	finally	not	None
await	for	or	True

(续表)

break	from	pass	
class	global	raise	
continue	if	return	
def	import	try	
del	in	while	

注意：只有 False、None、True 的首字母为大写，其余保留字的首字母均为小写。

2.4 变量和赋值语句

2.4.1 变量

变量与数学中方程变量的概念类似，只不过在计算机程序中，变量不仅可以是数字，而且可以是任意数据类型。变量名必须遵循标识符的命名规则，变量值是变量中保存的数据，可以被多次修改。

2.4.2 简单赋值

在编程语言中，将数据放入变量的过程叫作赋值。Python 使用等号作为赋值运算符，格式为：变量 = 表达式。

需要注意的是，变量除了要遵循 Python 标识符的命名规范，还要避免和 Python 内置函数以及 Python 保留字重名。

```
>>> pi = 3.1415926                    # 将圆周率赋值给变量 pi
>>> str = "佳木斯大学"                 # 将地址赋值给变量 str
>>> flag = True                       # 将布尔值赋值给变量 flag
>>> print( pi, str, flag )
3.1415926 佳木斯大学 True
```

变量的值可以随时修改，只需要重新赋值即可。可以将不同类型的数据赋值给同一个变量，而不需要关心数据的类型。例如：

```
>>> name = "Python 语言"              # 将字符串赋值给变量 name
>>> name = 365                        # 将整数赋值给变量 name
>>> name = [20, 10, 30]               # 将列表赋值给变量 name
>>> print( name )
[20, 10, 30]
```

注意，变量一旦再次赋值，之前的值就被覆盖了，变量中只能容纳一个值。

除了赋值单个数据，也可以将表达式的运算结果赋值给变量，例如：

```
>>> sum = 100 + 20                    # 将加法的运算结果赋值给变量
>>> rem = 25 * 30 % 7                 # 将余数赋值给变量
>>> str = "佳木斯大学" + "信息电子技术学院"   # 将字符串的拼接结果赋值给变量
>>> print( sum, rem, str )
```