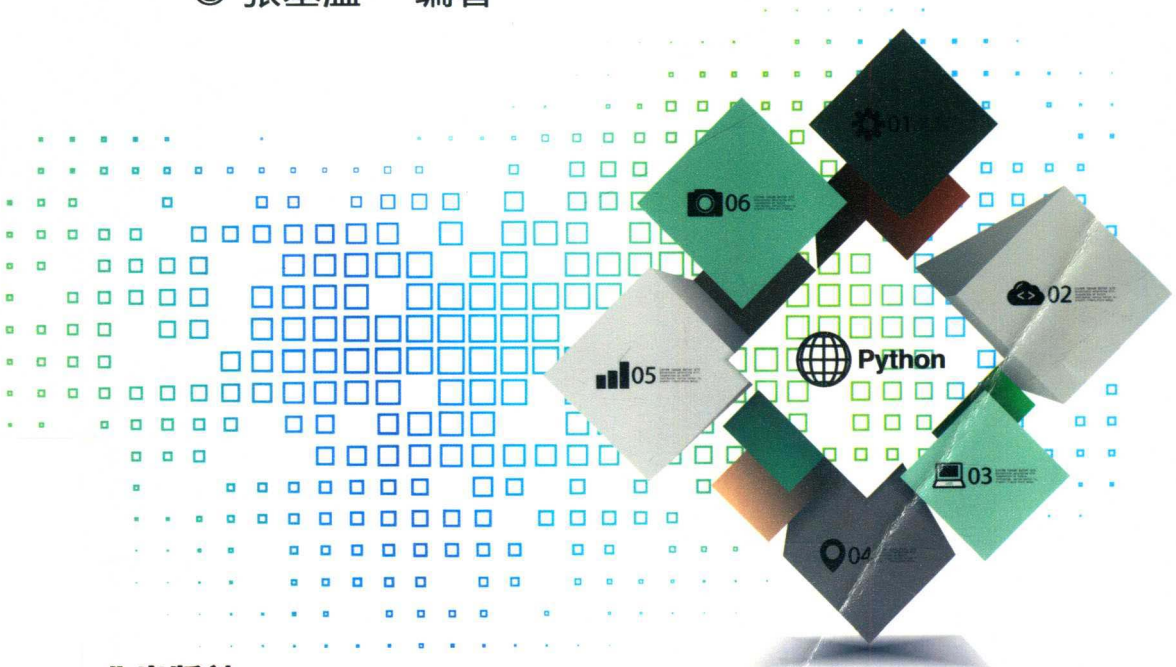
普通高等教育“十三五”计算机类规划教材

新概念

Python 程序设计

New Concept for Python Programming

◎ 张基温　编著



业出版社
HINE PRESS

普通高等教育“十三五”计算机类规划教材

新概念 Python 程序设计

张基温 编著

机械工业出版社

本书是一本 Python 基础教材。全书以 Python 3.6.1 为蓝本，分为五章。

第 1 章介绍如何在 Python 交互模式下模仿计算器，从简单计算，到使用内置函数计算和应用变量进行计算，再到使用选择结构和循环结构进行较复杂计算，并在其中穿插介绍基本数据类型的用法，最后以迭代和穷举收官，带领读者迈进 Python 殿堂。

第 2 章从正常处理和异常处理两个角度介绍 Python 程序过程的两种基本组织形式：函数和异常处理，并介绍与之相关的命名空间和作用域的概念。第 3 章介绍 Python 容器。第 4 章介绍类与对象、类的内置属性、方法与函数、类的继承。通过这三章的介绍，读者可夯实 Python 编程基础。

第 5 章通过数据文件、数据库、Socket 编程、Web 应用和大数据开发的介绍，读者可进一步提升 Python 应用开发的能力。

本书力求内容精练、概念准确、例题典型、代码简洁、习题丰富全面，适合教也容易学。同时，以二维码链接方式提供了知识扩充，为读者创建丰富而友好的学习环境。

本书适合初学 Python 语言的读者使用，也适合作为各类大专院校的教材，同时也可作为对 Python 感兴趣的读者的自学参考书。

图书在版编目（CIP）数据

新概念 Python 程序设计 / 张基温编著 . —北京：
机械工业出版社，2019.5
普通高等教育“十三五”计算机类规划教材
ISBN 978-7-111-62486-8

Ⅰ . ①新… Ⅱ . ①张… Ⅲ . ①软件工具—程序设计—
高等学校—教材 Ⅳ . ① TP311.561

中国版本图书馆 CIP 数据核字（2019）第 068500 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：路乙达 王保家 责任编辑：路乙达 王保家
责任校对：郑 婕 封面设计：张 静
责任印制：张 博
三河市宏达印刷有限公司印刷
2019 年 7 月第 1 版第 1 次印刷
184mm × 260mm · 14.25 印张 · 321 千字
标准书号：ISBN 978-7-111-62486-8
定价：37.80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务 网络服务
客服电话：010-88361066 机 工 官 网：www.cmpbook.com
010-88379833 机 工 官 博：weibo.com/cmp1952
010-68326294 金 书 网：www.golden-book.com
封面无防伪标均为盗版 机工教育服务网：www.cmpedu.com

Preface 前　言

近年来，一种程序设计语言日渐粲然，得到人们的青睐。这就是 Python。许多人都想在此时为熊熊燃起的 Python 之火再添上一把柴，本人也有此愿望。

Python 之所以能够成为一颗冉冉升起的新星，是因为其鲜明的特色。

Python 简单、易学。它虽然是用 C 语言写的，但是却摒弃了其中的指针，降低了学习和应用的难度。

Python 代码明确、优雅。Python 代码描述具有伪代码风格，使人容易理解；其强制缩进的规则，使得代码具有极佳的可读性。

Python 自由、开放。Python 是自由/开放源码软件（Free/Libre and Open Source Software，FLOSS）之一，它支持向不同的平台上移植，允许部分程序用 C/C++编写；它可提供脚本功能，允许把 Python 程序嵌入到 C/C++程序之中；它还鼓励编程人员进行创造、改进与扩张。因此，Python 在短短的发展历程中，形成了异常丰富、几乎覆盖一切应用领域的标准库和第三方库，为开发者提供了丰富的可复用资源和便利的开发环境。

为了彰显优势，Python 博采众长、趋利避害，形成了一套独特的语法体系。本书力求正本清源，从基本理论出发，对 Python 的语法给出一个清晰的概念和解释，以此为基础快速地将读者带入 Python 应用开发领域。

本书共分 5 章。第 1 章从简单易懂的计算开始，将读者引进 Python 世界。同时，穿插介绍一些最基本的语法知识，然后通过选择结构和循环结构，让读者在简单算法中试水。

第 2～4 章在第 1 章的基础上从函数、命名空间与作用域、异常处理、序列对象构建与操作、字符串、字典、集合、类与对象、类的内置属性、方法与函数继承等方面，使读者进一步夯实 Python 语法基础，提高语言应用能力。

第 5 章则是在上述 4 章的基础上，从应用广泛的数据文件、数据库、Socket 编程、Web 操作和大数据应用等方面，突出领域知识、模块选择和解题思路三要素，使读者练就 Python 开发的真本领。

除正文的内容之外，本书还有例题、习题和二维码链接。本书例题经典、代码简洁，以便充分发挥举一反三的作用。习题以节为单位进行组织，题型多样、针对性强，便于读

者学习某一节后，立即可以从不同角度检测学习效果。有些在其他同类书中可能有不妥的表述，拿来作为反例，以习题的形式供读者分析。

除此之外，本书中还收集了许多相关知识和较大型的案例，以二维码链接的形式提供，这样可以为初学者提供继续学习或提升能力的途径。

需要说明的是，本书中的程序代码基本上是在 IDLE 环境中运行的。在 IDLE 环境中，关键字、标识符、对象实例、运行结果、异常代码等分别用不同的颜色表示。为减轻读者阅读的难度，本书中操作者键入的代码及数据为正常字体，而程序给出的结果为黑体。

本书就要出版了。它的出版，是我在程序设计教学改革工作中跨上的一个新台阶。本人衷心希望得到有关专家和读者的批评和建议，也希望能多结交一些志同道合者，把本书编写得更好一些。同时，还要向在本书编写过程中参加了部分工作的张秋菊、史林娟、张展赫、戴璐等谨表谢意。

张基温

己亥正月于穗小海之畔

Contents 目　录

前　言

第 1 章　Python IDLE 作为万能计算器 …… 1

1.1　在 IDLE 中用操作符进行算术计算 …… 1

1.1.1　Python 算术操作符 …… 1

1.1.2　操作符特性 …… 3

1.1.3　注释 …… 4

1.1.4　Python 数据类型 …… 4

1.1.5　数据对象三属性及其获取 …… 7

1.1.6　回显与 print()函数 …… 7

习题 1.1 …… 8

1.2　使用函数计算 …… 9

1.2.1　函数与内置函数 …… 9

1.2.2　Python 内置计算函数对象 …… 10

习题 1.2 …… 10

1.3　利用 math 模块计算 …… 11

1.3.1　导入模块并浏览模块成员 …… 11

1.3.2　math 模块及其应用 …… 12

1.3.3　从一个模块中导入对象 …… 14

习题 1.3 …… 15

1.4　为对象命名——变量的引用 …… 15

1.4.1　Python 变量及其特性 …… 15

1.4.2　Python 变量的赋值 …… 17

1.4.3　Python 标识符与关键字 …… 18

1.4.4　input()函数 …… 19

习题 1.4 …… 20

1.5　选择型计算 …… 22

1.5.1　布尔类型与布尔表达式 …… 23

1.5.2　if-else 型选择结构 …… 26

1.5.3　if-else 嵌套与 if-elif 型选择结构 …… 27

习题 1.5 …… 29

1.6　重复型计算 …… 33

1.6.1　while 语句 …… 33

1.6.2　for 语句 …… 34

1.6.3　循环嵌套 …… 36

1.6.4　在交互环境中执行功能完整的代码段 …… 38

1.6.5　循环中断语句与短路语句 …… 39

1.6.6　for-else 语句与 while-else 语句 …… 40

习题 1.6 …… 40

1.7　迭代与穷举 …… 42

1.7.1　迭代 …… 42

1.7.2　穷举 …… 45

习题 1.7 …… 47

第 2 章　Python 过程组织与管理 …… 49

2.1　Python 函数 …… 49

2.1.1 函数及其基本环节……50
2.1.2 Python 函数参数技术……54
2.1.3 Python 函数的第一类对象特征……57
2.1.4 递归……58
2.1.5 lambda 表达式……60
习题 2.1……61
2.2 Python 命名空间与作用域……64
2.2.1 Python 命名空间……64
2.2.2 Python 作用域……67
2.2.3 Python 名字解析的 LEGB 规则……71
习题 2.2……72
2.3 Python 异常处理……73
2.3.1 异常处理的基本思路与异常类型……73
2.3.2 try…except 语句……75
2.3.3 异常类型的层次结构……76
2.3.4 else 子句与 finally 子句……77
2.3.5 异常的人工触发：raise 与 assert……78
习题 2.3……78

第 3 章 Python 容器……80

3.1 序列对象构建与操作……80
3.1.1 直接书写合法的序列实例对象……81
3.1.2 用构造方法构建序列对象……81
3.1.3 列表推导式与生成器推导式…82
3.1.4 序列对象判定与参数获取……84
3.1.5 序列对象的连接与拆分……85
3.1.6 序列对象的元素索引、切片与排序……87
3.1.7 序列遍历与迭代……89
3.1.8 列表的可变性操作……90
3.1.9 对象赋值、浅复制与深复制……92
习题 3.1……94
3.2 Python 字符串个性化操作……97
3.2.1 字符编码标准……97
3.2.2 字符串测试与搜索……98
3.2.3 字符串修改……99
3.2.4 字符串分割与连接……100
3.2.5 字符串格式化与 format() 方法……100
3.2.6 正则表达式……103
习题 3.2……109
3.3 字典……111
3.3.1 字典与散列函数……111
3.3.2 字典对象的创建……111
3.3.3 可作用于字典的主要操作符……112
3.3.4 用于字典操作的函数和方法……113
习题 3.3……114
3.4 集合……116
3.4.1 创建集合对象……116
3.4.2 Python 集合运算操作符与方法……117
3.4.3 可变集合操作方法……119
3.4.4 面向集合容器的操作函数……119
习题 3.4……120

第 4 章 基于类的程序设计……121

4.1 类与对象……121
4.1.1 类模型与类语法……121
4.1.2 对象创建与__init__()方法……123
4.1.3 最小特权原则与成员访问限制……126

4.1.4 实例方法、静态方法与类方法 …… 128
习题 4.1 …… 130
4.2 类的内置属性、方法与函数 ……132
4.2.1 类的内置属性 …… 132
4.2.2 获取类与对象特征的内置函数 …… 133
4.2.3 操作符重载 …… 137
4.2.4 Python 内置类属性配置与管理方法 …… 139
习题 4.2 …… 146
4.3 继承 ……148
4.3.1 类的继承 …… 148
4.3.2 Python 新式类与 object 类 …… 150
4.3.3 子类访问父类成员的规则 …… 152
4.3.4 子类实例的初始化与 super … 152
习题 4.3 …… 157

第 5 章 Python 应用开发 ……160

5.1 Python 文件 ……160
5.1.1 Python 文件概述 …… 160
5.1.2 打开文件与文件属性 …… 161
5.1.3 文件可靠关闭与上下文管理器 …… 163
5.1.4 文件对象内置属性 …… 164
5.1.5 文本文件读写 …… 165
5.1.6 二进制文件的序列化读写 …… 166
5.1.7 文件指针位置获取与移动 …… 169
5.1.8 文件和目录管理 …… 169
习题 5.1 …… 171
5.2 Python 数据库操作 ……173
5.2.1 数据库与 SQL …… 173
5.2.2 借助 ODBC 模块操作数据库 …… 174
5.2.3 用 SQLite 引擎操作数据库 …… 176
习题 5.2 …… 179
5.3 Python Socket 编程 ……180
5.3.1 TCP/IP 与 Socket …… 180
5.3.2 Socket 模块与 Socket 对象 …… 182
5.3.3 TCP 的 Python Socket 编程 …… 185
5.3.4 UDP 的 Python Socket 编程 …… 187
习题 5.3 …… 189
5.4 Python WWW 应用开发 ……190
5.4.1 WWW 及其关键技术 …… 190
5.4.2 用 urllib 模块库访问网页 …… 194
5.4.3 爬虫框架 scrapy …… 202
习题 5.4 …… 207
5.5 Python 大数据处理 ……208
5.5.1 大数据及其特征 …… 208
5.5.2 大数据计算特点 …… 210
5.5.3 大数据处理过程 …… 211
5.5.4 大数据处理模块 …… 213
5.5.5 大数据开发案例鉴赏 …… 213
习题 5.5 ……213

附录 二维码链接目录 ……215

参考文献 ……217

第 1 章 Chapter 1

Python IDLE 作为万能计算器

如图 1.1 所示，在 Python 交互编程环境中，在提示符>>>后面输入一个 Python 命令，按 Enter 键，便可以启动 Python 解释器对这条命令进行解释执行，给出结果。这种对代码及时反馈的交互模式非常适合初学者验证语法规则，对有经验的 Python 人员也可用于尝试新的 API、库以及函数。对于一般人来说，它可以被当作是一个计算器。但是，Python IDLE 的计算功能要比一般计算器功能强大得多，可计算的内容也广泛得多，可以称得上一个万能计算器。

链 1-1 Python 程序的运行与 IDLE

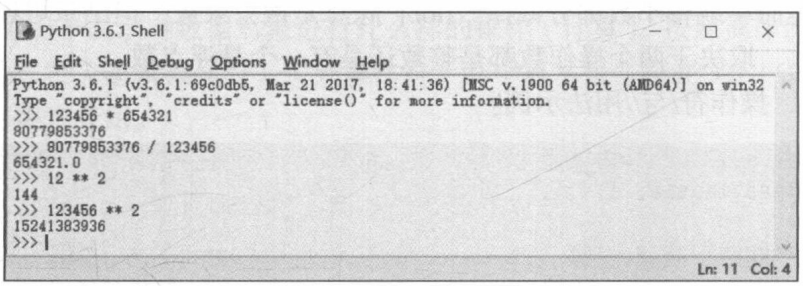


图 1.1 Python IDLE 用作计算器示例

1.1 在 IDLE 中用操作符进行算术计算

计算器最基本的应用是进行算术计算。本节介绍如何在 IDLE 中直接用操作符进行算术计算。

1.1.1 Python 算术操作符

在程序中，操作符（operators，也称运算符）是计算机操作命令的简洁表示。为了

便于用户计算，各种计算机程序设计语言中都会将一些常用计算机操作定义为内置操作符供用户使用，并按照操作的性质，将操作符分为不同的种类。表 1.1 给出了 Python 内置的算术操作符。

表 1.1 Python 内置的算术操作符（假定 a = 10，b = 3）

优先级	操作符	数学意义	操作对象数目	实　例	结合方向
高	**	幂	双目	a ** b 为 10 的 3 次方，返回 1000	右先
	+、–	正负号	单目	+10，–3	
中	//	floor 除	双目	a // b 返回 3	左先
	%	取模	双目	a % b 返回 1	
	/	真除	双目	a / b 返回 3.3333333333333335	
	*	相乘	双目	a * b 返回 30	
低	+	相加	双目	a + b 返回 13	
	–	相减	双目	a – b 返回 7	

说明：

1）注意由两个字符组成的操作符，一定不能在两个字符之间加入空格。

2）程序设计语言中的算术操作符与普通数学中的算术操作符有些相同，也有些不同，如在 Python 中用“*”表示乘，用“/”和“//”表示除，用“**”表示幂。这主要是为了便于用键盘进行操作。

3）在 Python 3.x 中，将除分为两种：真除（/）和 floor 除（//）。真除也称浮点除，即无论是两个整数相除，还是带小数的浮点数相除，结果都是要保留小数部分的浮点数。floor 除则是一种整除，其结果是真除结果向下舍入（或称向–∞舍入）小数部分得到的整数，而不是简单地将小数部分截掉。floor 除总是截去余数，但结果可以是整数，也可以是浮点数，取决于两个操作数都是整数还是有一个是浮点数。

代码 1-1 操作符/与//用法示例。

```
>>> 9 / 7, 9 // 7
(1.2857142857142858, 1)
>>> -9 / 7, -9 // 7
(-1.2857142857142858, -2)
>>> 9 / -7, 9 // -7
(-1.2857142857142858, -2)
>>> 5 /2.3, 5 // 2.3
(2.173913043478261, 2.0)
>>> 5 / -2.3, 5 // -2.3
(-2.173913043478261, -3.0)
```

4）取模操作符%是取 floor 除后的余数，即先计算出 floor 除值，再按公式：被除数–（整除值*除数）计算得到其值。

代码 1-2 操作符%用法示例。

```
>>> 9 % 7,9 % -7, -9 % 7, -9 % -7
```

```
(2, -5, 5, -2)
>>> 5 % 2.3, 5 % -2.3, -5 % 2.3
(0.40000000000000036, -1.8999999999999995, 1.8999999999999995)
```

注意：在 Python 中，括在圆括号中的几个数据对象，组成了一个元组（tuple，程序中简写为 tup）。元组是由多个数据对象作为元素所组成的数据对象序列，在语法上每个元组都相当于一个数据对象。显然，多个用逗号分隔的表达式被一次性解释执行时，生成的是一个元组。在这个元组中，数据对象作为元素形成用逗号分隔的序列。

1.1.2　操作符特性

表达式是数据和操作符按照一定规则排列的求值计算表示。在一个表达式中可能不包含操作符，也可能包含一个操作符或包含多个操作符。掌握操作符的特性，关系到对表达式计算过程以及取值的正确理解。从表 1.1 中可以看出，每个操作符都具有四个方面的特性。结合下面的例子说明这四个特性的用法。

代码 1-3　操作符特性示例。

```
>>> + 2                          #求正
2
>>> * 2                          #非法表达式
SyntaxError: can't use starred expression here
>>> 5 % + 3                      #求正优先，合法
2
>>> 5 + % 3                      #求正优先，非法
SyntaxError: invalid syntax
```

说明：

1）操作符用于表示对数据对象进行什么样的操作（计算）。按照操作的性质，操作符分为算术操作符、逻辑操作符、关系操作符。其中算术操作符在 1.1.1 节已经介绍，其他操作符将逐步介绍。

2）操作符需要的操作对象数目。例如，表达式+2 是正确的，而表达式*2 就是错误的，因为操作符*是一个双目操作符，需要两个操作对象。

3）操作符的优先级（precedence）。如表 1.1 所示，Python 算术操作符分为三个不同的优先级。当一个表达式中含有不同级别的操作符时，高优先级的操作符先与数据对象结合。例如，表达式 5% + 3 执行的顺序相当于 5% (+ 3)，即先对 3 取正，再用+3 对 5 求模，得 2。而表达式 5 + %3 就是错误的，因为按照优先级，应当先进行求模操作，但%要求两个数据对象，而+不是数据对象。

4）操作符的结合性（associativity）。操作符的结合性规定了在一个表达式中两个同级别的操作符相邻时哪个操作符先与数据对象结合，或两个子表达式相邻时先进行哪个子表达式的计算。“左先”就是操作符左面的数据对象先与之结合，“右先”就是操作符右面的数据对象先与之结合。例如操作符**和–都是右先，所以在表达式–10 ** –2 中，先执行–2 中的–，再执行**，最后执行最前面的–；而不是先执行–10 中的–，因为那样

的结果是 0.01，而不是–0.01。

除了上述四个特性，Python 还有一个特殊的操作符——一对圆括号。在表达式中，圆括号分组可以超越 Python 的优先级和结合性规则，即它可以强制先执行圆括号中的子表达式。当有多个圆括号形成嵌套结构时，内层圆括号内的表达式优先级别高于外层；当有多个分组并列时左优先，即分组具有左先结合性。这一点与普通数学相同，不再赘述。但需要指出的是，在复杂表达式中，人们往往会显式使用圆括号来提高子表达式优先级别的可读性。

1.1.3 注释

在 Python 程序中，以字符#开头的字符序列称为注释。注释是不被解释器解释的部分，仅用于如下两种情况。

1）对程序的某些部分进行适当的说明和解释，以提高程序的可读性，如代码 1-3 中的注释。

2）在程序调试时，为了分析某些的作用，可以在其前加一个注释符号#，看看这条语句不存在时，程序的执行情况有什么变化。当还需要这条语句时，只要删除添加的注释符号#即可。

1.1.4 Python 数据类型

在 Python 中，一切皆对象。数据对象是程序处理与加工的对象。每一个数据对象属于特定的类型，是特定类型的实例。类型规定了实例的操作和存储属性，了解类型的属性，也就了解了这类实例对象的属性。为了便于用户开发，Python 提供了丰富的数据类型。表 1.2 给出了 Python 3.x 主要的内置数据类型，可以分为标量类型和容器类型两大类。

表 1.2 Python 3.x 主要的内置数据类型

类型分类		类型名称	描　述	可变类型	示　例
标量类型	数值类型	整数	int	否	123
		浮点数	float	否	12.3、1.2345e+5
		复数	complex	否	（1.23, 5.67j）
	布尔类型	布尔值	bool	否	True、False
容器类型	序列	字符串	str	否	'abc' "abc" '"abc"' "123"
		列表	list	是	[1, 2, 3]、['abc', 'efg', 'ijklm']、list[1, 2, 3]
		元组	tup	否	(1, 2, 3, '4', '5')、tuple("1234")
	字典	字典	dict	是	{'name': 'wuyuan', 'blog': 'wuyuans.com', 'age': 23}
	集合	可变集合	set	是	set([1, 2, 3])
		不可变集合	frozenset	否	frozenset([1, 2, 3])

说明：

1）标量类型也称原子类型，是不可再分的数据对象，主要包括数值类型和布尔类型。容器类型也称组合类型，主要包括序列、字典和集合。在 Python 中比较特殊的是字符串，它具有容器的性质——可以容纳字符。但是在 Python 中，字符不是原子类型，也常把字符串看成基本类型。

2）Python 数据对象分为可变与不可变两大类。不可变对象是指对象的值不能被改变。若要改变，就称为另一个对象，会被另外存储。或者说，不可变对象是按值存储的。可变对象不受此限制，其值可以在原来的存储位置被改变。

3）除此之外，Python 还内置了一些其他特殊对象类型，如 None、模块、类、函数、文件等。这些内容将穿插在有关章节中介绍。

下面先介绍其中几种主要类型。

1. 整数和浮点数

整数和浮点数都属于数值类型，它们都是用数字表示，所以也称数字类型。但它们具有如下不同。

1）整数的类型名用 int 表示，浮点数的类型名用 float 表示。

2）整数类型数据对象可以用下列 4 种形式表示。

二进制：0、1，并加前缀 0b 或 0B，如 0b1001。

八进制：数字 0、1、2、3、4、5、6、7，加前缀 0o，或 0O，如 0o3567810。

十六进制：数字 0～9、A～F（或 a～f），加前缀 0x 或 0X，如 0x3579acf。

链 1-2　机器数的浮点格式与定点格式

十进制：数字 0～9，不加任何前缀。

而浮点数类型的数据对象仅能表现为一个 0～9 和小数点组成的数字序列。

3）整数类型数据对象表示的整数数值是精确的，而浮点数类型数据对象表示的大部分实数是近似的。因为许多二进制小数换算成十进制小数时，得到的是一个无穷小数值，其精度受计算机字长限制。也就是说，浮点数类型并不能精确地表示任何实数。在程序设计语言中，之所以称为浮点数，是因为在计算机内，它们采用了浮点表示格式，而整数类型采用了定点格式。此外，也不提倡在计算机中对两个浮点数进行相等比较。

4）在 Python 中，整数类型数据对象的取值范围可以任意大，仅受限于所用计算机系统的可用内存大小。

代码 1-4　Python 的整数类型对象的取值范围可以任意大。

```
>>> 99999 ** 99
9990104849431886366088059804028029154004345369796553866540094908138594575986162068371798214123026924209118782383921146943538722057054699799977580008475440744973980716030344675748108624929563154031641451813178503640103763002134537524379822655184645683257905595574017252764130455649272583256993531300425482080649240092160935475461012317475259759993157354372605902542813411030155819338431910097687712154624952723282098019352901532659298074019199496336368611420338714592310635625568489514900009899999
```

2. 元组和列表

元组（tuple）和列表（list）都是由数据对象组成的序列。所谓序列，是指它们的组成元素与元素在容器中的位置顺序有对应关系。二者的不同，首先在于元组是不可变对象，而列表是可变对象。另外，元组用圆括号作为边界符，例如：

```
(1,7,'a',5,3)
```

而列表用方括号作为边界符，例如：

```
[1,7,'a',5,3]
```

3. 字典和集合

字典和集合都是以花括号为边界符，但是字典的元素为键值对，键与值之间用冒号连接，例如：

```
{'A':90,'B':80,'C':70,'D':60}
```

集合的元素可以为任何对象，例如：

```
{'B',6,9,3,'A'}
```

4. 字符串类型

在 Python 中，用一对单撇号（'）、一对双撇号（"）以及一对三撇号（'''）作为起止符的字符序列称为字符串（string）。从用法上可以将字符串分为如下两种。

（1）单行字符串

用单撇号和双撇号作为起止符的字符串只能是单行字符串——不可跨行。二者的区别在于，双撇号作为起止符的字符串中，可以包含用单撇号作为起止符的字符串。

（2）多行字符串

用三撇号作为起止符的字符串是一种特殊字符串，其特殊性主要表现在如下方面：

1）可以是多行字符串。

2）多行字符串中可以包含格式信息，包括撇号、制表符以及回车符等。

代码 1-5 字符串应用示例。

```
>>> "abc'def'gh"
"abc'def'gh"
>>> '''a"bb'ccc'dd"ee'''
'a"bb\'ccc\'dd"ee'
>>> '''abcdefg''hijk'lmn'op''qrst uvw'''
"abcdefg''hijk'lmn'op''qrst\nuvw"
```

其中，最后一行输出中的\n 称为转义字符，表示换行。

5. 转义字符

转义字符就是赋予某些字符以特殊的意义。它们都以反斜杠为前缀，目的是告诉计算机，后面的字符是转义字符。转义字符中，大部分是用一个字符来代表一些常见的计算机操作，如换行、回车、制表、响铃、换页、退格、续行、终止等。八进制、十六进制标识的转义字符也是这个意义。还有些是避免与其他字符已经赋予的意义冲突、混淆

而变义的。例如，要在字符序列中增加一个反斜杠，但是反斜杠已经被定义为转义字符前缀，为了避免这个意义上可能的冲突，就在其前再加一个反斜杠，告诉计算机后面的斜杠有特殊意义——不再是转义字符前缀。表 1.3 列出了一些常用转义字符。

表 1.3　转义字符

转义字符	描　述	转义字符	描　述	转义字符	描　述	转义字符	描　述
\（行尾）	续行符	\a	响铃	\n	换行	\f	换页
\\	反斜杠符号	\b	退格（Backspace）	\v	纵向制表符	\o	八进制，后为八进制字符
\'	单引号	\e	转义	\t	横向制表符	\x	十六进制，后为十六进制字符
\"	双引号	\000	空	\r	回车	\000	终止，忽略之后的字符串

1.1.5　数据对象三属性及其获取

在 Python 中，每个数据对象都有三个基本属性：值、ID 和类型。其中，ID 是该数据对象的身份码。在 Python 程序中，一个对象一旦被创建，就会得到一个系统分配的唯一标识码（identity），也称对象的身份码，并且这个身份码将伴随这个对象一生，即一旦对象被创建，它的身份码就不允许更改。不同的身份码表示不同的对象。

数据对象的值可以通过回显（echo）或者用 print()函数获取，而数据对象的类型和身份码可以分别用内置函数 type()和 id()获取。

代码 1-6　数据对象的类型以及 ID 的获取示例。

```
>>> type(123),id(123)
(<class 'int'>, 1405666176)
>>> type(123.456),id(123.456)
(<class 'float'>, 2753616288336)
>>> type('abcdef'),id('abcdef')
(<class 'str'>, 2753627142160)
```

注意：尽管 Python 有 int、float、str 等一系列的类型声明符，但在使用一个数据对象时，并不需要先声明其类型，一个对象一经创建，Python 就会根据其存在形式自动判定出其类型，并在内部为其添加一个类型标志。

1.1.6　回显与 print()函数

1. 回显与 print()函数的异同

在交互模式下输入一个表达式，就会自动返回该表达式的值，这种“输出”称为“回显”（echo）。回显使用简便，但往往会受某些限制。因此，Python 输出使用的输出指令是内置函数 print()。

代码 1-7　回显与 print()函数的用法比较。

```
>>> 1/7
0.14285714285714285
>>> print(1/7)
0.14285714285714285
```

```
>>> '#*'*10                                    #回显 10 个'#*'组成的新字符串
'#*#*#*#*#*#*#*#*#*#*'
>>> print('#*'*10)                             #输出 10 个'#*'组成的新字符串的值
#*#*#*#*#*#*#*#*#*#*
>>> 3,5,8                                      #三个表达式一起解释执行的回显
(3, 5, 8)
>>> print(3,5,8)                               #输出三个表达式的值
3 5 8
>>> print('abc',123)
abc 123                                        #输出两个不同类型的值
```

说明：回显与 print()函数的输出在多数情况下略有差别——回显字符串或元组这些容器对象时，也表明它们是容器，而 print()函数仅输出值。此外，print()函数可以进行输出格式控制，而回显无法实现这个功能。

2. print()函数中分隔符与终结符的控制

一般说来，print()函数的输出具有如下默认格式：

1）一个 print()函数可以输出多个表达式的值。这时，作为参数的表达式之间用逗号分隔，而所输出的值之间默认用空格作为分隔符。

2）print()函数执行时，默认最后添加一个换行操作，即默认最后添加一个终结符。然而，print()函数允许使用参数 sep 改变分隔字符，并允许采用参数 end 改变终结符。

代码 1-8 print()函数中分隔符与终结符的控制示例。

```
>>> print(1,2,3)                                #用默认格式输出 3 个表达式值
1 2 3
>>> print(1,2,3,sep = '**')                     #用指定分隔符'**'输出 3 个表达式值
1**2**3
>>> print(1);print(2)                           #用默认终结符执行 2 个 print()函数
1
2
>>> print(1,end = '##');print(2,end = '##')    #用指定终结符'##'执行 2 个 print()函数
1##2##
>>> print(1,end = ' ');print(2,end = ' ')      #用指定终结符' '执行 2 个 print()函数
1 2
```

习题 1.1

1. 选择题

（1）表达式-5 // 3 的输出值为_____。

A. -1.0　　B. -1.6666666666666666

C. -1.6666666666666667　　D. -2

（2）表达式 5 / 3 的输出值为_____。

A. 1　　B. 1.6666666666666666

C. 1.6666666666666667　　D. 2

（3）表达式-5 % 3 的输出值为_____。

A. 1　　B. -1.0　　C. -2　　D. 2

（4）表达式 2 ** 3 ** 2 的输出值为_____。

A. 512　　B. 64　　C. 32　　D. 36

（5）表达式.1** −2 ** 2 的值近似于_____。

A. 0.0001　　B. 10000　　C. −10000　　D. −0.0001

（6）语句 world = "world"; print ("hello" + world)的执行结果是_____。

A. helloworld　　B. "hello" world　　C. hello world　　D. 语法错

（7）代码 print (type(('China', 'Us', 'Africa'))的输出为_____。

A. <clsss, 'set'>　　B. <clsss, 'list'>　　C. <clsss, 'dict'>　　D. <clsss, 'tuple'>

（8）代码 print (type({'China', 'Us', 'Africa'})的输出为_____。

A. <clsss, 'set'>　　B. <clsss, 'list'>　　C. <clsss, 'dict'>　　D. <clsss, 'tuple'>

2. 填空题

（1）表达式 5 % 3 + 3 // 5 * 2 的运算结果是_____。

（2）表达式 int (1234.5678 * 10 + 0.5) % 100 的运算结果是_____。

3. 实践题

在交互编程模式下，计算下列各题。

（1）从今天开始，100 天后是星期几？共经过多少个完整的星期？

（2）从今天开始，倒退 50 天是星期几？共经过多少个完整的星期？

（3）从当前时刻开始，经过 200 小时后是几点（按 24 时制）？共经过了几天？

1.2　使用函数计算

函数是操作符的扩展，也是程序中组织与管理过程的手段。Python 函数分为内置函数（built-in function）、模块函数对象和自定义函数对象三个层次。

链 1-3　Python3 内置函数

1.2.1　函数与内置函数

函数是实现一个功能的计算代码段的封装，它用一个名字定义一个代码段之后，就可以用这个名字代表其所定义的代码段——称为函数调用，形成一次定义、多次被调用的机制。通常，这个被定义的代码段是为处理某些数据而编写的，这些被处理的数据可能会因这个代码段在程序中的运行环境不同而异。为此，函数定义时，还需要为其指定需要处理并且因上下文而异的数据。这就形成了函数参数。函数参数也称函数的形式参数，它们仅作为函数所定义代码的处理角色存在，在调用时需要将具体使用的参数——实际参数的值传递给这些形式参数。例如一个用于计算 x^y 的函数，需要告诉函数计算的 x 和 y 各是多少。函数被调用时，用给定的参数运行所定义的代码段，就会给出计算结果，这称为函数返回。

函数可以自己设计，也可以使用经过验证的别人设计的函数。为了方便应用，Python 把一些仅次于算术操作符的常用计算定义成函数集成在自己的核心部分，供人们直接使用而不需任何额外定义。这类函数被称为内置函数。