第 2 章

成为大牛前的必备知识

本章内容提要

在深入学习一门编程语言之前，需要先学会基本的语法和规范。本章主要讲述 Python 的一些基本语法、保留字、变量、基本数据类型和运算符等知识，还讲述最常用的输入和输出函数。

2.1 Python 的语法特点

微视频

学习 Python 开发之前，首先需要了解 Python 程序的语法特点。

2.1.1 代码注释

Python 中的注释有单行注释和多行注释。Python 中单行注释以#开头，如：

```
#这是一个单行注释
print("茅檐低小，溪上青青草。")
```

单行注释既可以放在代码的前一行，也可以放在代码的右侧。例如：

```
print("茅檐低小，溪上青青草。")    #这是一个单行注释
```

☆**大牛提醒**☆

添加注释的目的是解释代码的功能和用途。注释可以出现在代码的任意位置，但是需要注意的是，注释不能分割关键字和标识符。例如，下面的注释就是错误的。

```
aa=float(#这是一个单行注释 input("请输入商品的价格："))
```

☆**经验之谈**☆

在实际开发的过程中，注释除了可以解释代码的功能和用途以外，还可以用于临时注释不想被执行的代码。这个技巧在代码排错的时候非常有用。

多行注释用 3 个单引号（'''）或 3 个双引号（"""）将注释括起来。

（1）3 个单引号。

```
'''
创作团队：云尚科技
文件名称：2.11.py
功能介绍：主要实现系统安全的检查工作
```

```
'''
```

（2）3 个双引号。

```
"""
创作团队：云尚科技
文件名称：2.11.py
功能介绍：主要实现系统安全的检查工作
"""
```

2.1.2 代码缩进

与其他常见的程序设计语言不同，Python 的代码块不使用大括号（{}）来控制类、函数及其他逻辑判断。Python 语言的主要特色就是采用代码缩进和冒号来区分代码之间的层次结构。

【例 2.1】执行缩进（源代码\ch02\2.1.py）。

```
#严格执行缩进的规则
if 1==2:
    print ("客从远方来，遗我一端绮。")
    print ("相去万余里，故人心尚尔。")
else:
    print ("著以长相思，缘以结不解。")
    print ("以胶投漆中，谁能别离此。")
```

程序运行结果如图 2-1 所示。

```
====================== RESTART: D:/python/ch02/2.1.py
著以长相思，缘以结不解。
以胶投漆中，谁能别离此。
```

图 2-1 例 2.1 的程序运行结果

☆**经验之谈**☆

实现缩进的方法有两种，包括使用空格和<Tab>键。其中，一个 Tab 键作为一个缩进量；使用空格时，通常采用 4 个空格作为一个缩进量。建议采用空格进行缩进。

Python 语言对代码的缩进要求非常严格，同一个级别代码块的缩进量必须相同。如果缩进量不相同，则会抛出 SyntaxError 异常。例如以下错误提示：

```
>>>if 1==2:
    print ("客从远方来，遗我一端绮。")
print ("相去万余里，故人心尚尔。")
SyntaxError: invalid syntax
```

☆**大牛提醒**☆

同一个级别代码块的缩进量，除了保证相同的缩进空白数量，还要保证相同的缩进方式，因为有的使用 Tab 键缩进，有的使用 2 个或 4 个空格缩进，需要改为相同的方式。

2.1.3 编码规范

使用 Python 编写代码，需要遵守如下规范：

（1）不能在行尾加分号，例如以下代码是不规范的。

```
if 1==2:
    print ("客从远方来，遗我一端绮。");
    print ("相去万余里，故人心尚尔。");
```

（2）每行的字符数最多不超过 80 个。如果超过，建议使用小括号将多行的内容隐式连接起来。例如以下代码：

```
a=("客从远方来，遗我一端绮。相去万余里，故人心尚尔。文采双鸳鸯，裁为合欢被。"
"著以长相思，缘以结不解。以胶投漆中，谁能别离此？")
```

（3）每个 import 语句只导入一个模块，尽量避免一次导入多个模块。例如，下面的代码是不规范的。

```
import sys,os
```

推荐使用以下写法：

```
import sys
import os
```

（4）通过必要的空行可以增加代码的可读性。在函数或者类的定义之间空两行，方法定义之间空一行。如果需要分割一些功能，也可以空一行。

（5）尽量避免在循环中使用+和+=运算符进行累加字符串。由于字符串是可变的，这样做会创建临时对象，而这通常是不必要的操作。

2.1.4　换行问题

在 Python 语言中，常见的换行问题如下：

1. 换行符

如果是 Linux/UNTX 操作系统，换行字符为 ASCII LF（linefeed）；如果是 DOS/Windows 操作系统，换行字符为 ASCII CR LF（return + linefeed）；如果是 Mac OS 操作系统，换行字符为 ASCII CR（return）。

例如，在 Windows 操作系统中换行：

```
>>>print ("客从远方来，\n遗我一端绮。")
客从远方来，
遗我一端绮。
```

2. 程序代码超过一行

如果程序代码超过一行，可以在每一行的结尾添加反斜杠（\），继续下一行，这与 C/C++ 的语法相同。例如：

```
if 100 < a < 100 and 1 <=b <=10\
   and 1000 <= c <= 10000 and 0 <= d < 26:   #多个判断条件
```

☆**大牛提醒**☆

行末的反斜杠（\）之后不要加注释文字。

如果是以小括号（）、中括号［］或大括号｛｝包含起来的语句，不必使用反斜杠（\）就可以直接分成数行。例如：

```
name = ('苹果', '香蕉',  '橘子',
          '芒果',   '西瓜',     '橙子')
```

3. 将数行表达式写成一行

如果要将数行表达式写成一行，只需在原来除最后一行以外的每一行的结尾添加分号（;）即可。例如：

```
>>>a = '苹果'; b = '香蕉'; c = '橙子'
>>> a
'苹果'
>>> b
```

```
    '香蕉'
    >>> c
    '橙子'
```

2.2 标识符与保留字


微视频

标识符用来识别变量、函数、类、模块及对象的名称。Python 的标识符可以包含英文字母（A～Z、a～z）、数字（0～9）及下画线符号（_），但有以下几个方面的限制：

（1）标识符的第 1 个字符必须是字母表中的字母或下画线（_），并且变量名称之间不能有空格。

（2）Python 的标识符有大小写之分，如 Data 与 data 是不同的标识符。

（3）在 Python 3.x 版本中，非 ASCII 标识符也被允许使用。

（4）保留字不可以当作标识符。

☆**经验之谈**☆

Python 语言支持汉字作为标识符使用。虽然不建议使用汉字作为标识符，但是在实际运行中，程序不会报错。例如以下代码:

```
    >>> 古诗="客从远方来，遗我一端绮。"
    >>> print(古诗)
    客从远方来，遗我一端绮。
```

保留字也叫关键字，不能被用作任何标识符名称。读者可以使用以下命令查看 Python 的保留字：

```
    >>>import keyword
    >>>keyword.kwlist
    ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
```

如果在开发程序时，不小心使用 Python 中的保留字作为了模块、类、函数或者变量的名称，将会提示错误信息：SyntaxError: invalid syntax。例如：

```
    >>> as="客从远方来，遗我一端绮。"
    SyntaxError: invalid syntax
```

由于 Python 语言是区分大小写的，因此 as 和 AS 是不一样的，所以以下代码就不会报错。

```
    >>> AS="客从远方来，遗我一端绮。"
    >>> As="客从远方来，遗我一端绮。"
    >>> aS="客从远方来，遗我一端绮。"
```

2.3 变量


微视频

在 Python 解释器内可以直接声明变量的名称，但不必声明变量的类型，因为 Python 会自动判别变量的类型。

定义 Python 中的变量名称，需要遵循以下规则：

（1）变量名称必须是一个有效的标识符。
（2）变量名不能和 Python 中的保留字冲突。
（3）尽量选择有意义的单词作为变量名。
（4）谨慎使用大小写字母 O 和小写字母 l。
为变量赋值可以通过等于号（=）来实现。语法格式如下：

```
变量名=value
```

例如，创建一个整数变量，并为其赋值为 88。

```
>>>a =88    #创建变量 a 并赋值为 88，该变量为数值型
>>>a
88
```

读者可以在解释器内直接做数值计算，例如：

```
>>>555 + 666
1221
```

内置的 type()函数可以用来查询变量所指的对象类型。
例如：

```
>>> a= 2000                        整数类型的变量
>>> print(type(a))
<class 'int'>
>>> b= "客从远方来，遗我一端绮。"  字符串类型的变量
>>> print(type(b))
<class 'str'>
```

在 Python 中，变量就是变量，没有类型，这里所说的“类型”是变量所指的内存中对象的类型。等号用来给变量赋值。等号运算符左边是一个变量名，等号运算符右边是存储在变量中的值。

Python 中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后才会被创建。如果创建变量时没有赋值，会提示错误，例如：

```
>>> name
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    name
NameError: name 'name' is not defined
```

Python 允许用户同时为多个变量赋同一个值。例如，将 a 和 b 两个变量都赋值为数字 666，通过内置函数 id()可以获取内存的地址，得到的结果是一样的。

```
>>>a =b =666
>>>print(a,b)
666 666
```

上面创建两个变量，值为 666，两个变量被分配到相同的内存空间。
也可以同时为多个对象指定不同的变量值，例如：

```
>>>x, y, z = 666, 888, "明月何时照我还"
>>>print(x,y,z)
666 888 明月何时照我还
```

两个整型对象 666 和 888 分配给变量 x 和 y，字符串对象"明月何时照我还"分配给变量 z。


微视频

2.4 基本数据类型

Python 3.x 版本中有两个简单的数据类型，即数字类型和字符串类型。

2.4.1 数字类型

Python 3.x 版本支持 int、float、bool、complex 4 种数字类型。

☆**大牛提醒**☆

在 Python 2 中是没有 bool 的，用数字 0 表示 False，用 1 表示 True。在 Python 3 中，把 True 和 False 定义成了关键字，但它们的值还是 1 和 0，可以和数字相加。

1. int（整数）

下面是整数的例子：

```
>>> a = 666688
>>> a
666688
```

可以使用十六进制数值来表示整数，十六进制整数的表示法是在数字之前加上 0x，如 0x80120000、0x100010100L。

例如：

```
>>> a=0x6EEEFFFF
>>> a
1861156863
```

2. float（浮点数）

浮点数的表示法可以使用小数点，也可以使用指数的类型。指数符号可以使用字母 e 或 E 来表示，指数可以使用+/−符号，也可以在指数数值前加上数值 0，还可以在整数前加上数值 0。

例如：

```
6.66    12.     .007     1e100     3.14E-10      1e010    08.1
```

使用 float()内置函数，可以将整数数据类型转换为浮点数数据类型。例如：

```
>>> float(660)
660.0
```

3. bool（布尔值）

Python 的布尔值包括 True 和 False，只与整数中的 1 和 0 有对应关系。例如：

```
>>> True==1
True
>>> True==2
False
>>> False==0
True
>>> False==-1
False
```

这里利用符号==判断左右两边是否绝对相等。

4. complex（复数）

复数的表示法是使用双精度浮点数来表示实数与虚数的部分，复数的符号可以使用字母 j 或 J。例如：

```
1.5 + 0.5j      1J       2 + 1e100j        3.14e-10j
```

数值之间可以通过运算符进行运算操作。例如：

```
>>> 50 + 40                                #加法
90
>>> 5.6 - 2                                #减法
3.6
>>> 30 * 15                                #乘法
450
>>> 1/2                                    #除法，得到一个浮点数
0.5
>>> 1//2                                   #除法，得到一个整数
0
>>> 15 % 2                                 #取余
1
>>> 2 ** 10                                #乘方
1024
```

在数字运算时，需要注意以下问题：

（1）数值的除法（/）总是返回一个浮点数，要获取整数需使用//操作符。

（2）在整数和浮点数混合计算时，Python 会把整数转换为浮点数。

【例 2.2】计算学生的总成绩和平均成绩（源代码\ch02\2.2.py）。

```
name="张小明"                               #保存学生的姓名
print ("该学生的姓名是: "+name)
maths=92.5                                  #保存学生的数学成绩
#使用内置的 str()函数可以将数值转换为字符串
print("该学生的数学成绩是: "+str(maths))
chinese=65.5                                #保存学生的语文成绩
print("该学生的语文成绩是: "+ str(chinese))
english=80.5                                #保存学生的英语成绩
print("该学生的英语成绩是: "+ str(english))
sum= maths+chinese+english                  #保存学生的总成绩
print("该学生的总成绩是: "+str(sum))
avg= sum/3                                  #保存学生的平均成绩
print("该学生的平均成绩是: "+str(avg))
#使用 if 语句判断学生的成绩如何
if avg<65:
   print ("该学生的成绩较差")
if 65<=avg<75:
   print ("该学生的成绩及格")
if  75<=avg<90:
   print ("该学生的成绩良好")
if  avg>=90:
   print ("该学生的成绩优秀")
```

程序运行结果如图 2-2 所示。

```
====================== RESTART: D:/python/ch02/2.2.py
该学生的姓名是：张小明
该学生的数学成绩是：92.5
该学生的语文成绩是：65.5
该学生的英语成绩是：80.5
该学生的总成绩是：238.5
该学生的平均成绩是：79.5
该学生的成绩良好
```

图 2-2　例 2.2 的程序运行结果

2.4.2　字符串类型

Python 将字符串视为一连串的字符组合。在 Python 中，字符串属于不可变序列，通常使用单引号、双引号或者三引号括起来。这三种引号形式在语义上没有区别，只是在形式上有些差别。其中单引号和双引号的字符序列必须在一行上，而三引号内的字符序列可以分布在连续的多行上。

例如下面的代码：

```
>>> a="张小明"           #使用双引号时，字符串的内容必须在一行
>>> b='最喜欢的水果'     #使用单引号时，字符串的内容必须在一行
>>> c='''骤雨东风对远湾，旁然遥接石龙关。 野渡苍松横古木，断桥流水动连环。
客行此去遵何路，坐眺长亭意转闲。'''
>>> print (a)
张小明
>>> print (b)
最喜欢的水果
>>> print (c)
骤雨东风对远湾，旁然遥接石龙关。 野渡苍松横古木，断桥流水动连环。
客行此去遵何路，坐眺长亭意转闲。
```

【例 2.3】输出一个小屋图形（源代码\ch02\2.3.py）。

由于该字符画有多行，所以使用三引号作为定界符比较合适。代码如下：

```
print('''
        @@@@@@@@@@
       @          @
      @            @
     @              @
    @                @
   @@@@@@@@@@@@@@@@@@@@
     @              @
     @      @@@@@   @
     @      @ @ @   @
     @      @@@@@   @
     @              @
     @              @
     @@@@@@@@@@@@@@@@
 ''')
```

程序运行结果如图 2-3 所示。

```
================== RESTART: C:/Program Files/Python37/2.3.py
          @@@@@@@@@@
         @          @
        @            @
       @              @
      @                @
     @@@@@@@@@@@@@@@@@@@@
       @              @
       @      @@@@@   @
       @      @ @ @   @
       @      @@@@@   @
       @              @
       @              @
       @@@@@@@@@@@@@@@@
```

图 2-3　例 2.3 的程序运行结果

☆**大牛提醒**☆

字符串开头与结尾的引号要一致。

下面的案例将字符串开头使用了双引号、结尾使用了单引号。

```
>>> a = "hello world'
Traceback (  File "<interactive input>", line 1
   a = " hello world '
                     ^
SyntaxError: invalid token
```

由此可见，当字符串开头与结尾的引号不一致时，Python 会显示一个 invalid token 的信息。

2.4.3　数据类型的相互转换

有时候，用户需要对数据内置的类型进行转换。数据类型的转换，只需要将数据类型作为函数名即可。以下几个内置的函数可以执行数据类型之间的转换，这些函数返回一个新的对象，表示转换的值。

1. 转换为整数类型

语法格式如下：

```
int(x)
```

将 x 转换为一个整数。例如：

```
>>>int(3.6)
3
```

☆**大牛提醒**☆

int()函数不能转换成非数字类型的数值。例如，使用 int()函数转换字符串时，将会提示 ValueError 错误。例如:

```
>>> int("16个工作日")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    int("16个工作日")
ValueError: invalid literal for int() with base 10: '16个工作日'
```

2. 转换为小数类型

语法格式如下：

```
float(x)
```

将 x 转换为一个浮点数。例如：

```
>>> float （10）
10.0
```

3. 转换为字符串类型

语法格式如下：

```
str(x)
```

将 x 转换为一个字符串。例如：

```
>>>str(123567)
'123567'
```

【例 2.4】模拟出租车的抹零结账行为（源代码\ch02\2.4.py）。

假设出租车司机因为找零钱比较麻烦，所以进行抹零操作。这里 int()函数将浮点型的变量转换为整数类型，从而实现抹零效果。本案例还会用到 str()函数，主要作用是将数字转换为字符串类型。

```
ranges=5.6                          #保存乘客坐车的距离
moneys=8+(ranges-3)*2               #计算总票价
print("本次车费是: "+ str(moneys))
real_moneys=int(moneys)             #进行抹零操作
print("本次实付车费是: "+ str(real_moneys))
```

程序运行结果如图 2-4 所示。

```
========== RESTART: D:/python/ch02/2.4.py
本次车费是：13.2
本次实付车费是：13
```

图 2-4　例 2.4 的程序运行结果


微视频

2.5　运算符和优先级

Python 语言支持的运算符包括算术运算符、比较运算符、赋值运算符、逻辑运算符、位运算符、成员运算符和身份运算符。

2.5.1　算术运算符

Python 语言中常见的算术运算符如表 2-1 所示。

表 2-1　算术运算符

运　算　符	含　　义	举　　例
+	加，两个对象相加	1+2=3
-	减，得到负数或一个数减去另一个数	3-2=1
*	乘，两个数相乘或返回一个被重复若干次的字符串	2*3=6
/	除，返回两个数相除的结果，得到浮点数	4/2=2.0
%	取模，返回除法的余数	21%10=1
**	幂，a**b 表示返回 a 的 b 次幂	10**21=10^{21}
//	取整除，返回相除后结果的整数部分	7/3=2

【例 2.5】计算部门的销售业绩差距和平均值（源代码\ch02\2.5.py）。

这里首先定义两个变量，用于存储各部门的销售额，然后使用减法计算销售业绩差距，最后应用加法和除法计算平均值。

```
branch1=760000
branch2=540000
sub= branch1- branch2
avg= (branch1+ branch2)/2
savg=int(avg)
print("部门 1 和部门 2 的销售业绩差距是: "+ str(sub))
print("两个部门销售业绩的平均值是: "+ str(savg))
```

保存并运行程序，结果如图 2-5 所示。

```
========== RESTART: D:/python/ch02/2.5.py
部门1和部门2的销售业绩差距是：220000
两个部门销售业绩的平均值是：650000
```

图 2-5　例 2.5 的程序运行结果

2.5.2　比较运算符

Python 语言支持的比较运算符如表 2-2 所示。

表 2-2　比较运算符

运　算　符	含　　义	举　　例
==	等于，比较对象是否相等	(1==2) 返回 False
!=	不等于，比较两个对象是否不相等	(1!=2) 返回 True
>	大于，x>y 返回 x 是否大于 y	2>3 返回 False
<	小于，x<y 返回 x 是否小于 y	2<3 返回 True
>=	大于或等于，x>=y 返回 x 是否大于或等于 y	3>=1 返回 True
<=	小于或等于，x<=y 返回 x 是否小于或等于 y	3<=1 返回 False

【例 2.6】使用比较运算符（源代码\ch02\2.6.py）。

```
branch1=760000                                          #定义变量，存储部门 1 的销售额
branch2=540000                                          #定义变量，存储部门 2 的销售额
print("部门 1 的销售业绩是: "+str(branch1)+", 部门 2 的销售业绩是: "+str(branch2))
print("部门 1==部门 2 的结果是: "+str(branch1== branch1))     #等于操作
print("部门 1!=部门 2 的结果是: "+str(branch1!= branch1))     #不等于操作
print("部门 1>部门 2 的结果是: "+str(branch1>  branch1))      #大于操作
print("部门 1<部门 2 的结果是: "+str(branch1<  branch1))      #小于操作
print("部门 1<=部门 2 的结果是: "+str(branch1<=branch1))      #小于或等于操作
```

保存并运行程序，结果如图 2-6 所示。

```
========== RESTART: D:\python\ch02\2.6.py
部门1的销售业绩是：760000，部门2的销售业绩是：540000
部门1==部门2的结果是：False
部门1!=部门2的结果是：True
部门1>部门2的结果是：True
部门1<部门2的结果是：False
部门1<=部门2的结果是，False
```

图 2-6　例 2.6 的程序运行结果

2.5.3　赋值运算符

赋值运算符表示将右边变量的值赋给左边变量，常见的赋值运算符的含义如表 2-3 所示。

表 2-3　赋值运算符

运　算　符	含　　义	举　　例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
−=	减法赋值运算符	c −= a 等效于 c = c − a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

【例 2.7】使用赋值运算符（源代码\ch02\2.7.py）。

```
a = 36
```

```
b = 69
c = 60
#简单的赋值运算
c = a + b
print ("c 的值为: ", c)
#加法赋值运算
c += a
print ("c 的值为: ", c)
#乘法赋值运算
c *= a
print ("c 的值为: ", c)
#除法赋值运算
c /= a
print ("c 的值为: ", c)
#取模赋值运算
c = 12
c %= a
print ("c 的值为: ", c)
#幂赋值运算
a=3
c **= a
print ("c 的值为: ", c)
#取整除赋值运算
c //= a
print ("c 的值为: ", c)
```

保存并运行程序，结果如图 2-7 所示。

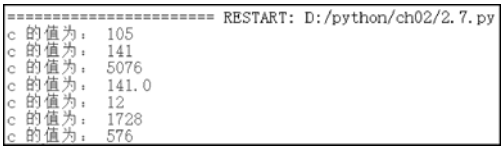

图 2-7　例 2.7 的程序运行结果

2.5.4　逻辑运算符

Python 语言支持的逻辑运算符如表 2-4 所示。

表 2-4　逻辑运算符

运　算　符	含　　义	举　　例
and	布尔"与"，x and y 表示如果 x 为 False，那么 x and y 返回 False，否则返回 y 的计算值	(10＞15 and 10＞16)返回 False
or	布尔"或"，x or y 表示如果 x 是 True，就返回 True，否则返回 y 的计算值	(15＞10 or 10＞15)返回 True
not	布尔"非"，not x 表示如果 x 为 True，就返回 False；如果 x 为 False，它返回 True	not (15＞10) 返回 False

【例 2.8】验证军队的夜间口令和编号（源代码\ch02\2.8.py）。

```
print ("开始验证军队的夜间口令")                    #输出提示消息
password=input("请输入今天夜间的口令: ")    #使用 input()函数接收输入的信息
num=input("请输入军队的编号: ")
number=int(num)                                  #将输入的信息转换为整数类型
if(password=="鸡肋" and (number==1002 or number==1006)):
```

```
    print ("恭喜您，口令正确！")
else:
    print ("对不起，口令错误！")
```

保存并运行程序，结果如图 2-8 所示。

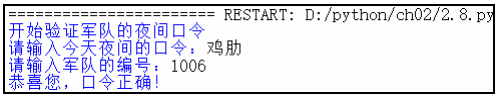

图 2-8　例 2.8 的程序运行结果

2.5.5　位运算符

在 Python 语言中，位运算符把数字看作二进制来进行计算。Python 语言支持的位运算符如表 2-5 所示。

表 2-5　位运算符

运　算　符	含　　义	举　　例
&	按位与，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	(12&6)=4，二进制为：0000 0100
|	按位或，只要对应的两个二进制位有一个为 1，结果位就为 1	(12|6)=14，二进制为：0000 1110
^	按位异或，当两个对应的二进制位相异时，结果为 1，否则为 0	(12^6)=10，二进制为：0000 1010
～	按位取反，对数据的每个二进制位取反，即把 1 变为 0、把 0 变为 1	(～6)=-7，二进制为：1000 0111
<<	左移动，把“<<”左边的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	(12<<2)=48，二进制为：0011 0000
>>	右移动，把“>>”左边的运算数的各二进位全部右移若干位，“>>”右边的数指定移动的位数	(12>>2)=3，二进制为：0000 0011

【例 2.9】使用位运算符（源代码\ch02\2.9.py）。

```
a = 12              #12 =0000 1100
b = 6               #6= 0000 0110
c = 0
#按位与运算
c = a & b;          #4 = 0000 0100
print ("c 的值为: ", c)
#按位或运算
c = a | b;          #14 = 0000 1110
print ("c 的值为: ", c)
#按位异或运算
c = a ^ b;          #10 = 0000 1010
print ("c 的值为: ", c)
#按位取反运算
c = ~a;             #-13 = 1000 1101
print ("c 的值为: ", c)
#左移动运算
c = a << 2;         #48 = 0011 0000
print ("c 的值为: ", c)
#右移动运算
c = a >> 2;         #3 = 0000 0011
print ("c 的值为: ", c)
```

保存并运行程序，结果如图 2-9 所示。

```
====================== RESTART: D:\python\ch02\2.9.py =======
c 的值为： 4
c 的值为： 14
c 的值为： 10
c 的值为： -13
c 的值为： 48
c 的值为： 3
```

图 2-9　例 2.9 的程序运行结果

2.5.6　成员运算符

Python 语言还支持成员运算符，其测试例中包含了一系列的成员，如字符串、列表、元组。成员运算符包括 in 和 not in。例如，x in y 表示：若 x 在 y 序列中，则返回 True；x not in y 表示：若 x 不在 y 序列中，则返回 True。

【例 2.10】使用成员运算符（源代码\ch02\2.10.py）。

```
a ='苹果'
b = '香蕉'
fruit = ['苹果', '荔枝', '橘子', '橙子', '柚子' ];
#使用 in 成员运算符
if ( a in fruit ):
   print ("变量 a 在给定的列表 fruit 中")
else:
   print ("变量 a 不在给定的列表 fruit 中")
#使用 not in 成员运算符
if ( b not in fruit ):
   print ("变量 b 不在给定的列表 fruit 中")
else:
   print ("变量 b 在给定的列表 fruit 中")
#修改变量 a 的值
a = '哈密瓜'
if ( a in fruit ):
   print ("变量 a 在给定的列表 fruit 中")
else:
   print ("变量 a 不在给定的列表 fruit 中")
```

保存并运行程序，结果如图 2-10 所示。

```
====================== RESTART: D:/python/ch02/2.10.py
变量a在给定的列表fruit中
变量b不在给定的列表fruit中
变量a不在给定的列表fruit中
```

图 2-10　例 2.10 的程序运行结果

2.5.7　身份运算符

Python 语言支持的身份运算符为 is 和 not is。其中，is 判断两个标识符是不是引用自一个对象；is not 判断两个标识符是不是引用自不同对象。

【例 2.11】使用身份运算符（源代码\ch02\2.11.py）。

```
a ='苹果'
b ='苹果'
#使用 is 身份运算符
if ( a is b):
```

```
    print ("a 和 b 有相同的标识")
else:
    print ("a 和 b 没有相同的标识")
#使用 not is 身份运算符
if ( a not in b ):
    print ("a 和 b 没有相同的标识")
else:
    print ("a 和 b 有相同的标识")
#修改变量 a 的值
a = '香蕉'
if ( a is b):
    print ("修改后的 a 和 b 有相同的标识")
else:
    print ("修改后的 a 和 b 没有相同的标识")
```

保存并运行程序，结果如图 2-11 所示。

```
================= RESTART: C:/Program Files/Python37/2.11.py
a和b有相同的标识
a和b有相同的标识
修改后的a和b没有相同的标识
```

图 2-11　例 2.11 的程序运行结果

2.5.8　运算符的优先级

下面是 Python 语言的运算符，以处理顺序的先后排列。

（1）()、[]、{}。

（2）object。

（3）object[i]、object[1:r]、object.attribute、function()。

“.”符号用来存取对象的属性与方法。下面的案例调用对象 t 的 append()方法，在对象 t 的结尾添加一个字符"t"：

```
>>> t = ["P","a","r","r","o"]
>>> t.append("t")
>>> t
['P', 'a', 'r', 'r', 'o', 't']
```

（4）+x、−x、～x。

（5）x**y：x 的 y 次方。

（6）x * y、x / y、x % y：x 乘以 y、x 除以 y、x 除以 y 的余数。

（7）x + y、x – y：x 加 y、x 减 y。

（8）x << y、x >> y：x 左移 y 位、x 右移 y 位。例如：

```
>>> x = 4
>>> x << 2
16
```

（9）x & y：位 AND 运算符。

（10）x ^ y：位 XOR 运算符。

（11）x | y：位 OR 运算符。

（12）<、<=、>、>=、==、!=、<>、is、is not、in、not in。

in 与 not in 运算符应用于列表（list）。is 运算符用于检查两个变量是否属于相同的对象；is not 运算符则用于检查两个变量是否不属于相同的对象。

!=与<>运算符是相同功能的运算符，都用来测试两个变量是否不相等。Python 建议使用!=运算符，而不使用<>运算符。

（13）not。

（14）and。

（15）or、lambda args：expr。

使用运算符时需要注意以下事项：

（1）除法应用在整数时，其结果会是一个浮点数。例如，8/4 会等于 2.0，而不是 2。余数运算会将 x / y 所得的余数返回来，如 7%4 =3。

（2）如果将两个浮点数相除取余数，那么返回值也会是一个浮点数，计算方式是 x – int(x / y) * y。例如：

```
>>>7.0 % 4.0
3.0
```

（3）比较运算符可以连在一起处理，如 a < b < c < d，Python 会将这个式子解释成 a < b and b < c and c < d。像 x < y > z 也是有效的表达式。

（4）如果运算符（operator）两端的运算数（operand），其数据类型不相同，那么 Python 就会将其中一个运算数的数据类型转换为与另一个运算数一样的数据类型。转换顺序为：若有一个运算数是复数，则另一个运算数也会被转换为复数；若有一个运算数是浮点数，则另一个运算数也会被转换为浮点数。

（5）Python 有一个特殊的运算符——lambda。利用 lambda 运算符能够以表达式的方式创建一个匿名函数。lambda 运算符的语法如下：

```
lambda args : expression
```

args 是以逗号（,）隔开的参数列表 list，而 expression 则是对这些参数进行运算的表达式。

例如：

```
>>>a=lambda x,y:x + y
>>>print (a(3,4))
7
```

x 与 y 是 a()函数的参数，而 a()函数的表达式是 x+y。lambda 运算符后只允许有一个表达式。如要达到相同的功能也可以使用函数来定义 a，如下所示：

```
>>> def a(x,y):              #定义一个函数
 return x + y               #返回参数的和
>>> print (a(3,4))
7
```

【例 2.12】运算符的优先级（源代码\ch02\2.12.py）。

```
a = 5
b = 8
c = 4
d = 2
e = 0
e = (a + b) * c / d        #(13 *4 ) / 2
print ("(a + b) * c / d 运算结果为: ",  e)
e = ((a + b) * c) / d      #(13 *4 ) /2
print ("((a + b) * c) / d 运算结果为: ",  e)
e = (a + b) * (c / d);     #（13）* (4/2)
print ("(a + b) * (c / d) 运算结果为: ",  e)
e = a + (b * c) / d;       #5 + (32/2)
print ("a + (b * c) / d 运算结果为: ",  e)
```

保存并运行程序，结果如图 2-12 所示。

```
====================== RESTART: D:/python/ch02/2.12.py
(a + b) * c / d 运算结果为：  26.0
((a + b) * c) / d 运算结果为：  26.0
(a + b) * (c / d) 运算结果为：  26.0
a + (b * c) / d 运算结果为：  21.0
```

图 2-12　例 2.12 的程序运行结果

2.6　Python 的输入和输出


微视频

Python 语言的内置函数 input()和 print()用于输入和输出数据。本节将讲述这两个函数的使用方法。

2.6.1　input()函数

Python 语言提供的 input() 函数从标准输入读入一行文本，默认的标准输入是键盘。input()函数的基本语法格式如下：

```
input([prompt])
```

其中，prompt 是可选参数，用来显示用户输入的提示信息字符串。用户输入程序所需要的数据时，就会以字符串的形式返回。

☆**经验之谈**☆

添加提示用户输入信息是比较友好的，对于编程时所需要的友好界面非常有帮助。

【例 2.13】使用 input()函数。

```
>>> a= input("请输入最喜欢的编程语言: ")
请输入最喜欢的编程语言: Python
>>> print(a)
Python
```

上述代码用于提示用户输入最喜欢的编程语言的名称，然后将名称以字符串的形式返回并保存在 a 变量中，以后可以随时调用这个变量。

当运行此句代码时，会立即显示提示信息“请输入最喜欢的编程语言：”，之后等待用户输入信息。当用户输入 Python 并按下 Enter 键时，程序就接收到用户的输入。最后调用 a 变量，就会显示变量所引用的对象——用户输入的编程语言名称。

☆**大牛提醒**☆

用户输入的数据全部以字符串形式返回，如果需要输入数值，就必须进行类型转换。

2.6.2　print ()函数

print()函数可以输出格式化的数据，与 C/C++语言的 printf()函数功能和格式相似。print()函数的基本语法格式如下：

```
print(value,…,sep=' ' ,end='\n')    #此处只说明了部分参数
```

上述参数的含义如下：

（1）value 是用户要输出的信息，后面的省略号表示可以有多个要输出的信息。

（2）sep 用于设置多个要输出信息之间的分隔符，其默认的分隔符为一个空格。

（3）end 是一个 print()函数中所有要输出的信息之后添加的符号，默认值为换行符。

【例 2.14】测试处理结果的输出（源代码\ch02\2.13.py）。

```
print("庄周梦蝴蝶","，蝴蝶为庄周")                #输出测试的内容
print("庄周梦蝴蝶","，蝴蝶为庄周",sep='*')        #将默认分隔符修改为'*'
print("庄周梦蝴蝶","，蝴蝶为庄周",end='>')        #将默认的结束符修改为'>'
print("庄周梦蝴蝶","，蝴蝶为庄周")                #再次输出测试的内容
```

保存并运行程序，结果如图 2-13 所示。这里调用了 4 次 print()函数。其中，第 1 次为默认输出，第 2 次将默认分隔符修改为'*'，第 3 次将默认的结束符修改为'>'，第 4 次再次调用默认的输出。

```
====================== RESTART: D:/python/ch02/2.13.py
庄周梦蝴蝶 ，蝴蝶为庄周
庄周梦蝴蝶*，蝴蝶为庄周
庄周梦蝴蝶 ，蝴蝶为庄周>庄周梦蝴蝶 ，蝴蝶为庄周
```

图 2-13　例 2.14 的程序运行结果

从运行结果可以看出，第一行为默认输出方式，数据之间用空格分开，结束后添加了一个换行符；第二行输出的数据项之间以'*'分开；第三行输出结束后添加了一个'>'，与第 4 条语句的输出放在了同一行中。

☆**大牛提醒**☆

从 Python 3.x 版本开始，将不再支持 print 输出语句，例如，print "Hello Python"，解释器将会报错。

如果输出的内容既包含字符串，又包含变量值，就需要将变量值格式化处理。

例如：

```
>>> x = 66
>>> print ("x = %d" % x)
x = 66
>>> print ("x = %d" , x)
x = %d 66
```

这里要将字符串与变量之间以%符号隔开。如果没有使用%符号将字符串与变量隔开，Python 就会输出字符串的完整内容，而不会输出格式化字符串。

【例 2.15】实现不换行输出（源代码\ch02\2.14.py）。

```
a="碧空溶溶月华静，"
b="月里愁人吊孤影。"
#换行输出
print( a )
print( b )

print('---------')
#不换行输出
print( a, end=" " )
print( b, end=" " )
print()
```

保存并运行程序，结果如图 2-14 所示。

```
====================== RESTART: D:/python/ch02/2.14.py
碧空溶溶月华静，
月里愁人吊孤影。
---------
碧空溶溶月华静， 月里愁人吊孤影。
```

图 2-14　例 2.15 的程序运行结果

本例中，通过在变量末尾添加 end=" "，可以实现不换行输出的效果。读者从结果可以看出换行和不换行的不同之处。

2.7　新手疑难问题解答

疑问 1：如何使用一条 print()语句输出多个内容，而且不换行？

解答：在 Python 语言编程中，默认情况下，一条 print()语句输出后会自动换行，如果想一次性输出多个内容，而且不换行，可以将要输出的内容使用英文半角逗号分隔。例如，下面的代码：

```
>>> x=1010
>>> y=2020
>>> z=3030
>>> print(x,y,z)
1010 2020 3030
```

疑问 2：input()函数在 Python 2.x 版本和 Python 3.x 版本中有什么不一样吗？

解答：在 Python 2.x 版本中，input()函数接收内容时，数值直接输入即可，并且接收后的内容作为数字类型。如果输入的类型是字符串，需要将对应的字符串使用括号括起来，否则会报错。

在 Python 3.x 版本中，输入的任何字符，都将作为字符串读取。如果想要转换为数值，需要将接收到的字符串进行类型转换。这里需要使用 int()函数与 float()函数进行转换。

例如，下面的代码：

```
>>> x= int(input("请输入整数: "))
请输入整数: 2020
>>> y = float(input("请输入浮点数: "))
请输入浮点数: 12.12
>>> x
2020
>>> y
12.12
```

2.8　实战训练


解题思路

实战 1：模拟银行的自助取款机。

编写 Python 程序，模拟银行的自助取款机。

（1）计算机输出信息：欢迎使用 XXX 取款机系统，请输入要取款的金额。

（2）用户输入：2000。

（3）计算机输出：取款成功，您本次取款金额为 2 000 元。

程序运行结果如图 2-15 所示。

```
====================== RESTART: D:/python/ch02/2.15.py
欢迎使用XXX取款机系统
请输入要取款的金额，2000
取款成功，您本次取款金额为2000元
```

图 2-15　实战 1 的程序运行结果

实战 2：根据体重和身高计算 BMI（身体质量指数）。

编写 Python 程序，实现根据体重和身高计算 BMI。BMI 的计算公式为：体重/身高的二次方。

（1）用户输入体重和身高。

（2）计算机输出 BMI。

（3）如果 BMI＜18.5，则输出“您的体重太轻了！”；如果 18.5≤BMI＜24.9，则输出“您的体重很完美！”；如果 24.9≤BMI＜29.9，则输出“您的体重偏高！”；如果 29.9≤BMI，则输出“您的体重太胖了！”。程序运行结果如图 2-16 所示。

实战 3：绘制 008 号坦克战车。

编写 Python 程序，使用键盘上的各种符号绘制一辆 008 号坦克战车，程序运行结果如图 2-17 所示。

```
====================== RESTART: D:/python/ch02/2.16.py
请输入您的体重：(千克)65
请输入您的身高：(米)1.73
您的BMI指数是：21.71806608974573
您的体重很完美！
```

图 2-16　实战 2 的程序运行结果

```
====================== RESTART: D:/python/ch02/2.17.py
                                  |
                          \--     |
                       --  ____ --
|__|==============@@@@@[/#008_____|
                        ____\_____]/
                    /__坦克战车_____|
                    \@@@@@@@@@@@@@@@/
```

图 2-17　实战 3 的程序运行结果