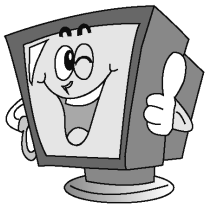

第 3 章　基本数据类型与表达式

本章学习要求

- 掌握三种数值及布尔类型的概念
- 正确使用三种数值类型
- 掌握字符串的概念和表示
- 熟悉基本的运算符
- 结合标准函数进行表达式计算
- 掌握字符串的基本操作和格式化输出

3.1　数据和数据类型的概念

3.1.1　计算机的数据表示

利用计算机处理现实世界的实际问题时，所有有效的信息都将表示成计算机能“理解”和“接受”的形式。通常，程序中需要处理的数据包括数值类、文本类和复合类。

例如，根据输入的半径值计算对应的面积，那么输入的半径值和所求的面积值就是数值类型的数据。如果想要在计算机屏幕上显示“Hello Python!”，那么编程输入的这串文字对应的就是文本类型的数据。有时要表示的对象比较复杂，如教务管理系统中的学生信息，既有“姓名”“学号”“性别”等方面的文本类数据，也有表示各门课程分数的数值类数据，甚至还包括照片等图像类信息，这就要利用复合数据进行处理了。

3.1.2　数据类型概念

各类由程序处理的数据都首先被保存在计算机的内存中，内存以字节为单位进行数据的分配和回收。不同类型的数据占用的字节数不一样，因此所表示的范围是有大小之分的，不仅存储方式不同，每种类型数据的运算也是有区别的。

所以在学习编程语言时，首先要了解和掌握这门设计语言的基本数据类型和基本运算。

3.2 基本数据类型

基本数据类型包括整数、浮点数和复数类型的数值类型，字符串类型的文本类型和布尔类型。

3.2.1 整数类型 int

Python 的整数类型与数学上的整数概念一致，有十进制、二进制、八进制和十六进制 4 种不同的表示方式。除常用的十进制外，其他进制都使用不同的引导符加以区分。整数的不同进制如表 3-1 所示。

表 3-1 整数的不同进制

进制	引导符	说明
十进制	无	默认格式。例如，100，−23
二进制	0b 或 0B	由 0 和 1 两个数码组成。例如，0B1101
八进制	0o 或 0O	由 0 到 7 八个数码组成。例如，0o1354
十六进制	0x 或 0X	由 0 到 9 和 a 到 f，或 A 到 F 共 16 个数码组成。例如，0x9A8B

3.2.2 浮点数类型 float

Python 用浮点数类型表示数学上的实数，有小数和指数两种形式。

3.14129、−123.65 是浮点数的十进制小数表示形式。指数形式采用以字母 e 或 E 表示以 10 为底的幂，格式如下：

```
<a>e 或 E<b>，表示 a*10^b
```

例如，9.2e−3，即 0.0092。−3.6E4，表示−36000。

Python 中浮点数一般以其近似值存储，取值范围和小数精度受不同计算机系统的限制会有所不同，但能满足日常处理的需要。

【例 3-1】

```
>>>23/3.02
7.6158940397351
```

3.2.3 复数类型 complex

Python 提供了数学中复数的表示形式 complex，分成实部和虚部两部分，格式如下：

```
real+imag(J/j)
```

【例 3-2】

```
x=3.5+2.78j
```

```
print(x.real)
print(x.imag)
```

显示结果为：

```
3.5
2.78
```

3.2.4 字符串类型 str

计算机早期主要应用于科学计算，所以程序处理的数据以数值类型为主。如今计算机的服务领域越来越多，处理对象日益丰富，需要进行大量文本数据的存储、编辑等操作。在程序中，文本数据通常以字符串的形式表示。

Python 语言中字符串是以一对单引号、双引号或三引号作为定界符的字符序列，它最基本的单位就是字符，分为可打印字符和不可打印的控制字符。打印字符包括：

①英文的大小写字母 A～Z 和 a～z。

②数字字符 0～9。

③标点符号和一些常用符号。

控制字符包括回车、制表符等，用\开始的转义字符表示，如表 3-2 所示。

表 3-2 转义字符

r	描 述	转义字符	描 述
\\	反斜杠符号	\t	横向制表符
\'	单引号	\n	换行
\"	双引号	\（在行尾）	续行符
\a	响铃	\f	换页
\b	退格	\OOO	八进制数 OOO 代表的字符
\r	回车	\xXX	十六进制数 XX 代表的字符

【例 3-3】

```
>>>print("Hello!")
Hello!
```

“Hello!”就是字符串，效果等同于'Hello!'。如果字符串本身带有单引号或双引号时，可以使用不同的引号嵌套，以示区别。例如：

【例 3-4】

```
>>>print('"Hello!"')
"Hello!"
```

可以使用 3 个单引号进行多行字符串的表示。例如：

【例 3-5】

```
>>>print("'I\'m
a
```

```
student''')

I'm
A
student
```

通常使用下标来区分字符串中某个指定字符。下标必须是整数，第一个字符对应的下标是 0，第二个字符对应的下标是 1，依次类推，最后第 n 个字符的下标就是 n−1。当然，字符串的下标也可以逆向标记，从最后一个字符的下标−1 开始，其他字符的下标往左依次递减，到达第一个字符时下标就是−n。

【例 3-6】示例代码如下：

```
s="Hello world"
print(s[0])
print(s[-1])

H
d
```

3.2.5 布尔类型 bool

布尔类型也称逻辑类型，这种数据只有 True（真）和 False（假）两个值。所有类型的数字 0（整数类型、浮点类型等其他类型）、空序列（字符串、列表、元组）及空字典的值为假，其他值为真。

【例 3-7】示例代码如下：

```
s=bool("Hello world")
t=bool(0)
x=bool("Hello world")+bool(0)
print(s)
print(t)
print(x)

True
False
1
```


3.3 运算符与表达式

3.3.1 运算符与表达式概念

对数据进行不同运算处理时运用的符号叫运算符。根据参与运算对象的数目，可以分为一元运算符、二元运算符。Python 常用的运算符包括算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、成员运算符、身份运算符等。

使用运算符和括号将运算对象组合在一起的式子就是表达式。单独的一个常量或变量都可以视为一个表达式。表达式的运算结果与所参与的运算、运算的目数和运算的优先级有关。

3.3.2 算术运算符

常用的算术运算符如表 3-3 所示，假设变量 a 取值 2，变量 b 取值 5。

表 3-3 算术运算符

运算符	描 述	举 例
-a	取负	结果为−2
a+b	相加	结果为 7
a-b	相减	结果为−3
a*b	相乘	结果为 10
a/b	相除	结果为 0.4
a//b	整除	结果为 0
a%b	取余	结果为 2
a**b	求幂	结果为 32

Python 中的算术运算符跟常规的算术概念基本一致。不过个别运算符有所区别，乘法的运算符用*表示。除法运算分为普通除法（/）和整除（//）两种，前者，无论参与运算的数据是整数还是浮点数，计算结果都是浮点数；后者，如果参与运算的是整数，则结果为整数，如果参与的有浮点数，则结果就是浮点数。因此，在编程时应根据实际需要选择合适的除法运算。算术运算符的优先顺序是先进行单目取反运算，再进行乘除之类的运算，最后是加减运算。

【例 3-8】示例代码如下：

```
>>>5/2                    #输出：2.5
>>>5//2                   #输出：2
>>>5%2                    #输出：1
>>>3.7//3                 #输出：1.0
```

3.3.3 关系运算符

关系运算符是进行两个对象大于、小于或相等与否关系比较的运算符，关系成立的运算结果为真，反之为假，所以关系运算的值是布尔类型数据：True 或 False。变量 a 和 b 参与的关系运算如表 3-4 所示。

表 3-4 关系运算符

运算符	描 述	举 例
a>b	比较 a 是否大于 b	若 a 大于 b，则返回 True，否则返回 False
a>=b	比较 a 是否大于等于 b	若 a 大于等于 b，则返回 True，否则返回 False
a<b	比较 a 是否小于 b	若 a 小于 b，则返回 True，否则返回 False
a<=b	比较 a 是否小于等于 b	若 a 小于等于 b，则返回 True，否则返回 False
a==b	比较 a 是否等于 b	若 a 等于 b，则返回 True，否则返回 False
a!=b	比较 a 是否不等于 b	若 a 不等于 b，则返回 True，否则返回 False

关系比较运算类似数学的不等式表示，但有区别。其中的“大于等于”和“小于等于”的运算符用两个并列的符号>=和<=表示，不同于数学上的≥和≤；用==两个等号表示相等关系，与数学上习惯用=一个等号表示相等概念不同；用!=表示不相等关系，而非用数学符号“≠”表示。这些是大家学习 Python 中需要注意的方面。

【例 3-9】示例代码如下：

```
>>>a=10
>>>b=20
>>>a>b                    #输出：False
>>>a!=b                   #输出：True
```

3.3.4 逻辑运算符

关系运算进行的是较简单的布尔判断，复杂的布尔表达式就需要逻辑运算了。逻辑运算包括逻辑非、逻辑与和逻辑或运算，分别用 not、and 和 or 三个运算符表示。逻辑运算的结果可以是 True 和 False 两个值，具体如表 3-5 所示。

表 3-5 逻辑运算符

a	b	a and b	a or b	not a
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True
True	True	True	True	False

逻辑运算和关系运算的结果非真即假的特点，往往被用作判断条件。因此在选择和循环结构中，逻辑运算和关系运算用作条件判断表达式的情况比较普遍。

【例 3-10】判断一个整数能否被 3 和 5 同时整除。

```
>>>x=100
>>>x%3==0 and x%5==0                    #输出：False
>>>y=150
>>>y%3==0 and y%5==0                    #输出：True
```

3.3.5 赋值运算符

在 Python 语言中，赋值运算是一种常用的运算，用一个=表示。赋值运算符不再意味左右相等关系，而表示将右边表达式的结果赋予左边的变量。赋值运算时，可以一次只给一个变量赋值，也可以同时给多个变量赋值。例如：

```
>>>x=100                          #一次赋值一个变量
>>>x,y,z=100,200,300             #一次赋值多个变量
>>>x,y=y,x                        #x，y 两个数值互换
```

同时给多个变量赋值属于同步赋值，表示将右边的表达式的每个结果按从左到右的顺序依次赋予左边的每个变量，接收赋值的变量顺序也是从左到右，并且左边变量与右边表达式的个数要匹配。Python 中同时给多个变量赋值的操作可以很方便地实现将两个变量值互换的操作。

另外，还有一种复合赋值的运算，其结构是：

```
变量  运算符=表达式
```

等同于：

```
变量=变量  运算符  表达式
```

例如：

```
>>>x=100
>>>x/=5                          #运算后 x 的值就是 20
```

3.3.6 位运算符

位运算符是针对二进制数按每一位进行运算的符号。

1. 按位取反（ ~ ）

按位取反是对一个二进制数的每一位进行取反操作。

例如，100 的二进制表示是 01100100，～100 的结果就是 10011011，十进制形式的数就是−101。

2. 按位与（&）

按位与是对两个二进制数的对应位进行逻辑与运算。在按位与操作过程中，只要对应的两位有一个是 0，结果就是 0，除非这两位都是 1，结果位才可能是 1。

例如，56&78，结果是 8。

```
56        00111000
78    &   01001110
          00001000
```

按位与有如下特殊用途。

①清零：希望某个指定位结果为 0，只需跟对应位是 0 的进行位与运算。

②取出指定位：希望取出某指定位原先的内容，可以跟对应位是 1 的进行位与运算。

3. 按位或（|）

按位或是对两个二进制数的对应位进行逻辑或运算，只要对应的两位有一个是 1，则结果就是 1，只有这两位都是 0 时，其结果位才可能是 0。

例如，56|78，结果是 126。

```
  56          00111000
  78      |   01001110
              01111110
```

按位与的特殊用途：置位，即可以利用跟对应位是 1 的数进行按位或运算，无论原位内容是否为 1，其结果都将是 1。

4. 按位异或（^）

按位异或是对两个二进制数的对应位进行逻辑异或运算，若对应的两位内容相同则结果是 0，若相异则为 1。

例如，56^78，结果是 118。

```
  56          00111000
  78      ^   01001110
              01110110
```

按位异或的特殊用途：可使指定位翻转，即一个数要翻转的位通过跟对应位是 1 的数进行按位异或运算，就可实现原先是 1 的位变成 0，原先是 0 的位变成 1。

5. 按位左移（<<）

按位左移是对左边的二进制数进行左移若干位的操作，左移的次数由右边的数决定，每左移一位，右端补一个 0，相当于左边的数乘一个 2。

例如，5<<4，表示把 00000101 左移 4 次，高位 4 个 0 移出，低位补 4 个 0，变成 01010000，对应的十进制数恰好是 5×2^4 的值 80。

6. 按位右移（>>）

按位右移是对左边的二进制数进行右移若干位的操作，右移的次数由右边的数决定，每右移一位，左端补一个 0，相当于左边的数除一个 2。

例如，5>>2，表示把 00000101 右移 2 次，高位补 2 个 0，低位移出两位，变成 00000001，对应的十进制数恰好是 $5//2^2$ 的值 1。

位运算的优先级从高到低，依次为~、<<、>>、&、|、^。

3.3.7 成员运算符

成员运算符描述与举例如表 3-6 所示。

表 3-6　成员运算符

运算符	描　述	举　例
a in b	判断在 b 的序列中是否存在 a	如果 a 在 b 的序列中，则返回 True
a not in b	判断在 b 的序列中是否不存在 a	如果 a 不在 b 的序列中，则返回 True

【例 3-11】应用举例。

```
>>>x="fa"
>>>y="ghdfjhjk"
>>>z="jhj"
>>>print(x not in y)
>>>print(z in y)
```

运行结果：

```
True
True
```

3.3.8　身份运算符

身份运算符的描述与举例如表 3-7 所示。

表 3-7　身份运算符

运算符	描　述	举　例
a is b	判断 a 和 b 是否引自同一个对象	如果 id(a)和 id(b)一致，则返回 True
a not is b	判断 a 和 b 是否引自不同对象	如果 id(a)和 id(b)不同，则返回 True

【例 3-12】应用举例。

```
>>>x=10
>>>y=10
>>>print(x is y)
>>>y=20
>>>print(x is y)
```

运行结果：

```
True
False
```

3.3.9　类型转换

一个表达式中有时会遇到多种类型数据的混合运算。由于不同类型数据的存储格式不同，因此要首先统一类型然后再进行相应的运算。数据转换的操作分为自动类型转换和强制类型转换两种方式。

1. 自动类型转换

混合了整数和实数的算术运算表达式计算时，以不降低精度的原则进行自动类型转

换，即整数一般先转换成浮点数再参与和实数的运算。

2. 强制类型转换

根据需要，在编程时可以将一种类型的数据强制转换成另一种类型的数据。强制转换内置函数及描述如表 3-8 所示。

表 3-8 强制转换内置函数

函 数	描 述	函 数	描 述
int(x)	将 x 强制转换成整数	ord(x)	将 x 转换成对应的 ASCⅡ值
float(x)	将 x 强制转换成浮点数	chr(x)	将 x 转换成对应的字符
str(x)	将 x 强制转换成字符串	eval(x)	将字符串 x 作为表达式计算

【例 3-13】应用举例。

```
>>> type(3.0+2)
<class 'float'>
>>>eval("3.14+3*5")
18.14
>>>ord("a")
97
```

3.3.10 运算符的优先级

如果一个表达式中出现多种运算符，则不同运算求值的顺序是有区别的，括号内的运算有最高优先权，其他的运算按照运算符的默认顺序进行。各类运算符的优先级顺序参见表 3-9。

表 3-9 各类运算符的优先级顺序

运算符	描述	优先级
**	幂运算	↓
~、+、-	按位取反、正、负号	
*、/、%、//	乘、除、取余、整除	
+、-	加法、减法	
>>、<<	右移、左移	
&	位与	
^、|	异或、位或	
==、!=、<=、<、>=、>	关系运算	
is、not is	身份运算	
in、not in	成员运算	
not	否定运算	
and	与运算	
or	或运算	

3.4 数值处理常用标准函数

Python 语言除提供可以直接使用的内置函数外，还有大量标准函数放在对应标准库和第三方库中，需要时只需到该库调取即可，大大方便了程序的编写。因此，有必要了解一些常用标准函数，以备编程时使用。

3.4.1 math 库的使用

平时，我们需要编程解决大量的数学类问题，一些常用的算术运算和常数 Python 已经定义成标准函数，并存放在 math 库中，需要时可通过 import 语句引入 math 库，这样就可以直接调用该函数了。调用格式是：

```
math.函数名(参数)
```

常用算术运算函数如表 3-10 所示。

表 3-10　常用算术运算函数

函数名或常数	描　述	举　例
math.e	自然常数 e	math.e
math.pi	圆周率 π	math.pi
math.log10(x)	返回以 10 为底的对数	math.log10(2)
math.pow(x,y)	返回 x 的 y 次方	math.pow(4,3)
math.sqrt(x)	返回 x 的平方根	math.sqrt(5)
math.ceil(x)	返回不小于 x 的最小整数	math.ceil(4.3)
math.floor(x)	返回不大于 x 的最大整数	math.floor(5.8)
math.trunc(x)	返回 x 的整数部分	math.trunc(7.6)
math.fabs(x)	返回 x 的绝对值	math.fabs(-4.3)
math.sin(x)	返回 x 的三角正弦值	math.sin(3)
math.asinx)	返回 x 的反三角正弦值	math.asin(0.5)
math.cos(x)	返回 x 的三角余弦值	math.cos (1.7)
math.acos(x)	返回 x 的反三角余弦值	math.acos(0.3)
math.tan(x)	返回 x 的三角正切值	math.tan(5)
math.atan(x)	返回 x 的反三角正切值	math.atan(1.6)

【例 3-14】执行以下代码：

```
import math
print(math.pi)
print(math.sqrt(2))
print(math.acos(0.6))
print(math.pow(3,4))
print(math.floor(7.4))
```

运行结果：

```
3.141592653589793
1.4142135623730951
0.9272952180016123
81.0
7
```

3.4.2 random 库的使用

随机数在计算机应用中十分普遍。random 库包含了各种伪随机数的生成函数。因为由计算机生成的随机数其实是按照一定算法产生的，所以其结果是必然、可预见的，不是真正意义上的随机数。

若程序中需要调用某个随机函数，则只有事前使用 import 语句引入 random 库，才能成功调取函数。

如果要调用若干个不同的随机函数，就可以引入整个 random 库，格式如下：

```
import  random
```

在程序中调取函数时要添加库名作为前缀（如 random.randint()），才能成功。

如果只是调用固定的某一个随机函数，就使用下面的格式，从库里直接调取那个函数，引用函数时无须添加库名作为前缀：

```
from  random  import  randint
```

引入的库名　引入的函数名

seed()是影响随机数发生器生成随机数的种子，它的使用格式是：

```
random.seed(a=None)
```

其中， a 为种子，一般是个整数，未设定时默认为系统时间。当 seed()函数的 a 取不同值时，前后两次生成的随机数序列是相异的，若 seed()函数的 a 取值相同，则前后两次生成的随机数序列是一样的，这种区分便于测试和同步数据。调用不同的随机函数，可以生成不同区间的各类随机数，具体如表 3-11 所示。

表 3-11　随机函数

函数名	描述
seed(a)	初始化随机数种子
random()	生成一个[0.0,1.0]之间的随机小数
randint(a,b)	生成一个[a,b]之间的整数
getrandbits(k)	生成一个 k 比特长度的随机整数
randrange(start,stop[,step])	生成一个[start,stop]之间以 step 为步长的随机整数
uniform(a,b)	生成一个[a,b]之间的随机小数
choice(seq)	从序列 seq 中随机返回一个元素
shuffle(seq)	将序列 seq 中的元素随机打乱，返回打乱后的序列

【例 3-15】执行以下代码：

```
import random
random.seed(15)
//循环产生 10 个 100 以内的随机整数，for 语句的使用将在第 4 章做详细介绍
for i in range(1,11):
        print(random.randint(1,100),end=", ")
print()
for i in range(1,11):
        print(random.randint(1,100),end=", ")
print()
random.seed(15)
for i in range(1,11):
        print(random.randint(1,100),end=", ")
print()
print(random.random())
for i in range(1,11):
        print(random.randrange(1,100,2),end=", ")
```

运行结果：

```
27, 2, 67, 95, 5, 21, 31, 3, 8, 88,
19, 89, 48, 31, 15, 44, 60, 91, 46, 36,
27, 2, 67, 95, 5, 21, 31, 3, 8, 88,
0.14724835647152645
89, 47, 31, 15, 43, 59, 91, 45, 35, 51,
```


3.5 字符串处理函数及方法

3.5.1 字符串的基本操作

字符串是 Python 常用的处理对象，字符串的基本操作如表 3-12 所示。

表 3-12 字符串的基本操作

操作符	描述
x+y	连接字符串 x 和 y
x*n 或 n*x	将字符串 x 复制 n 次
x in s	若字符串 x 是字符串 s 的子串，则返回 True，否则返回 False
str[i]	索引，返回字符串 str 中第 i 个字符
str[m:n]	切片，返回字符串 str 中第 m 个到第 n-1 个字符的子串

Python 不需要调用连接函数，而使用+符号就可以实现几个字符串的连接操作。例如：

```
>>>s="今天"+"是"+"星期天"
>>>print(s)
今天是星期天
```

在算术运算中，*代表乘法或幂次运算，而在字符串的处理中，*是进行字符串复制的运算符。例如：

```
>>>s="555"*3
>>>print(s)
555555555
```

Python 有特殊的运算符处理字符串的子串判断，不需要特别编写代码以实现此项功能。例如：

```
>>>print('tu'in "study")
True
```

一个字符串涉及多个不同字符，如何区分彼此呢？考虑到每个字符所在的位置是固定且唯一的，因此可以采用类似门牌编号的方式，将字符串中的每个字符的位置按排列顺序编号，从 0 开始，然后 1、2、…、n−1 一直到最后第 n 个字符。若要查找或表示其中的某个字符，只要知道它的位置就能锁定。有一点要特别注意，就是字符串的索引编号是从 0 开始的，不是从 1 开始的，所以索引号是 i 的序号，其实对应的是第 i+1 那个字符。例如：

```
>>>s="Python"
>>>print(s[0],s[5])
P n
```

除可用索引法单独提取字符串中的字符外，还可以用切片法提取字符串中的一段子串，子串的长度、起始位置由两个参数 m 和 n 决定，但要注意子串是不包括第 n 个字符的，只取字符串中第 m 个字符到第 n−1 个字符为止的子串。例如：

```
>>>s="Python"
>>>print(s[1:4])
yth
```

3.5.2 字符串的常用内置处理函数

在 Python 解释器内部，有一些字符串处理的内置函数。在面向对象中的叫法就是“方法”，使用效果没有区别。常用的字符串处理内置函数如表 3-13 所示。

表 3-13 字符串处理的内置函数

功能	函数名/方法	描述
字符串类型判断	str.isalpha()	判断字符串 str 是否全为字母，是返回 True，否返回 False
	str.isdigit()	判断字符串 str 是否全为数字，是返回 True，否返回 False
	str.islower()	判断字符串 str 是否全为小写字母，是返回 True，否返回 False
	str.isupper()	判断字符串 str 是否全为大写字母，是返回 True，否返回 False
大小写字母转换	str.lower()	字符串 str 变小写字母
	str.upper()	字符串 str 变大写字母
	str.title()	字符串 str 首字母变大写字母
	str.swapcase()	字符串 str 中的大小写字母互换

续表

功能	函数名/方法	描述
删除空格	str.strip([chars])	删除字符串 str 左右的空格
	str.lstrip([chars])	删除字符串 str 左边的空格
	str.rstrip([chars])	删除字符串 str 右边的空格
字符串操作	str.find(sub[,start[,end]])	在字符串 str 的[start,end]区间中查找 sub 子串，返回 sub 首次出现的位置
	str.replace(old,new[,count])	在字符串 str 中用 new 子串替换 old 子串
	str.split()	将字符串 str 拆分成列表
	str.count(sub[,start[,end]])	统计 sub 子串在字符串 str 的[start,end]区间出现的次数

【例 3-16】应用举例。

```
s="    s2 345d gfd645   "
print(s.isdigit())
print(s.title())
print(s.upper())
print(s.strip())
print(s.find('45',1,9))
print(s.count('45'))
print(s.split())
```

运行结果：

```
False
    S2 345D Gfd645
    S2 345D GFD645
s2 345d gfd645
7
2
['s2', '345d', 'gfd645']
```


3.6 字符串格式化方法

3.6.1 字符串的 format()格式化方法

早前人们较多地使用%控制字符串的输出格式，目前一般采用 format()方法对字符串实行格式化控制。

1. format()方法的基本格式

```
<模板字符串>.format（参数）
```

在“模板字符串”中用{}给后面的参数提前占位，一对{}表示一个参数值的输出，至于输出哪个参数值，一种方法是按照{}在“模板字符串”中的排列顺序，依次由后面的参

数值逐个填入；另一种方法是在{}内填上后面参数的序列编号（从 0 开始），按编号填充相应的参数值。例如：

```
name="zhangsan"
age=10
print('My name is {}.My age is {}.'.format(name,age))
```

显示结果：

```
My name is zhangsan.My age is 10.
```

修改成：

```
print('My age is {1}.My name is {0}.'.format(name,age))
```

显示结果：

```
My age is 10.My name is zhangsan.
```

2. format()方法的格式控制

如果对输出的数值有格式要求，那么就要在{}内添加相应的格式控制标记符，可以实现对输出项的对齐方式、宽度、精度等方面的控制。格式如下：

```
{<参数序号>:<格式控制符>}
```

format()格式说明如表 3-14 所示。

表 3-14 format()格式说明

:	<填充>	<对齐>	<宽度>	<，>	<.精度>	<类型>
引导符	单个字符	<左对齐 >右对齐 ^居中对齐	设定输出宽度	数字的千位分隔符 用于整数和浮点数	浮点数小数部分的精度或字符串的输出长度	整数类型： b、c、d、o、x、X 浮点类型： e、E、f、%

<填充>部分的字符是指当输出值未达到设定的宽度时填充空位的字符，若没有指定，则默认由空格填充空位。

<对齐>输出值可以设定左对齐、右对齐和居中对齐三种不同对齐方式。

<宽度>对输出值可以指定显示的宽度，实际数值未达到指定宽度时将出现空位，根据设定的对齐方式，空位出现的位置将会不同：左对齐方式，空位出现在数值的右侧；右对齐方式，空位出现在数值的左侧；居中对齐方式，空位出现在数值的两侧。

<.精度>浮点数输出时可以进行小数点后保留几位的控制，系统会自动截取小数部分。

<类型>当输出整数或浮点数时，根据需要可以有不同进制的选择，由不同的格式控制符决定。

b：以二进制形式显示整数。

c：以整数对应的 Unicode 字符显示。

d：以十进制形式显示整数。

o：以八进制形式显示整数。

x 或 X：以十六进制形式显示整数。

e 或 E：以指数形式显示浮点数。

f：以小数形式显示浮点数。

%：以百分号形式显示浮点数。

例如，应用举例。

```
name="shen"
a=12345.2356
print('{:-<10},{:>10,.2f}'.format(name,a))
```

运行结果：

```
shen------, 12,345.24
```

3.6.2 字符串的 f-string 格式化方法

f-string 亦称为格式化字符串常量（formatted string literals），是 Python 3.6 新引入的一种字符串格式化方法，主要目的是使格式化字符串的操作更加简便。f-string 在形式上是以 f 或 F 修饰符引领的字符串（f'xxx'或 F'xxx'），以大括号 {}标明被替换的字段，这个字段可以是变量、表达式或函数等。通常采用如下格式设置字符串：

```
{content:format}
```

其中，content 是替换并填入字符串的内容，format 是格式描述符。格式控制的方法与 3.6.1 节介绍的 format()方法一致。采用默认格式时不必指定 {:format}，只写 {content}即可。例如：

```
s="zhangsan"
print(f'My name is {s}.')
x=78.5656
y=123
print(f'{x:10.2f}')
print(f'{y:*>10d}')
```

运行结果：

```
My name is zhangsan.
     78.57
*******123
```

小　　结

本章介绍了 Python 语言常用的数值类、布尔类和字符串类的基本数据类型，还介绍了表达式的概念及算术、关系、逻辑、赋值等常用运算符，要掌握根据运算的优先级判断表达式的值，以及一些常规标准函数的使用方法，为后续的编程打好基础。

本章详细介绍了目前主流的字符串输出格式控制函数和方法 f-string()和 format()，有助于大家阅读理解相关知识。

习　　题

一、判断题

1．Python 利用 input()输入的数据都是字符串类型。（　　）

2．Python 中 1/4 的结果是 0。（　　）

3．表达式“bool(0)+bool(1)”与“bool(0) and bool(1)”的结果一致。（　　）

4．已知 a=65，那么执行语句 a = 12.78 后 a 的值是 12。（　　）

5．可以用 math.pi 表示圆周率。（　　）

6．s="gdfdawm"，s[2:5]表示"fdaw"这段子串。（　　）

7．Python 中整数只能以十进制形式显示，浮点数只能以小数形式显示。（　　）

8．2^3 表示 2^3。（　　）

二、填空题

1．print(math.floor(-3.56))的结果是_____。

2．print(~2&(3^7))的值为_____。

3．'***'*3 的结果是_____。

4．数学表达式 $\sin 15° + \frac{e^x - 5x}{\sqrt{x^2+1}} \times 4$ 的 Python 的写法是_____。

5．print(3>6 or (not 7) and 1<=3) 的执行结果是_____。

三、选择题

1．print('dsghj'+2)的执行结果是（　　）。

A．语法错误　　B．'dsghj2'　　C．dsghj dsghj　　D．2

2．print(3.65+2)的执行结果是（　　）。

A．5　　B．6　　C．5.65　　D．语法错误

3．print(hex(16),ord('a')) 的执行结果是（　　）。

A．16 a　　B．0x10 97　　C．10 a　　D．16 97

4．print(type(4//3)) 的执行结果是（　　）。

A．<class 'int'>　　B．<class 'float'>

C．<class 'double'>　　D．<class 'complex'>

5．print(chr(50)) 的执行结果是（　　）。

A．50　　B．0　　C．2　　D．5

6. print(math.sqrt(4)**math.sqrt(9)) 的执行结果是（　　）。

A．36.0　　B．6.0　　C．8.0　　D．222

四、程序阅读题

1．写出下面程序的运行结果。

```
a=58
b=7
print('a/b={},a//b={},a%b={}'.format(a/b,a//b,a%b))
```

2．写出下面程序的运行结果。

```
a=10
b=1234.678
print('a={:*<10},b={:1,.2f}'.format(a,b))
```

3．写出下面程序的运行结果。

```
a=158
b=7
print('a/b={:e},a//b={:x},a%b={:b}'.format(a/b,a//b,a%b))
```

五、编程题

1．利用随机函数产生一个三位数，判断它个、十、百位上的数字。

2．输入两个整数，求它们的位与、位或及位异或的值。

3．某市居民用水按阶梯收费，第一阶梯的用水量为 216 立方米（含）以下，销售价格为每立方米 2.90 元；第二阶梯用水量在 216～300 立方米（含）时，销售价格为每立方米 3.85 元；300 立方米以上的销售价格为每立方米 6.7 元。请编写程序，要求输入用水量，计算需缴纳的水费。用水量和水费的输出格式要求保留两位小数。

4．编程实现输入两个字符串 a、b，进行两个字符串的连接、比较、查找的操作。