第3章

基本数据类型与字符处理

Python 语言支持两大类数据类型：基本数据类型和组合数据类型。基本数据类型包括整数类型、浮点数类型、复数类型、布尔类型和字符串类型，组合数据类型包括列表类型、元组类型、集合类型和字典类型，共计 9 种数据类型，如图 3-1 所示。

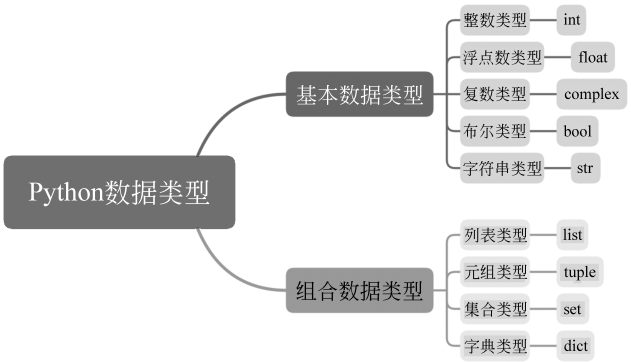

图 3-1　Python 数据类型

3.1　整 数 类 型

整数类型类似于数学中的整数，不同于其他语言的是该类型无取值范围的限制，有 4 种进制表示方法，即十进制、二进制、八进制和十六进制，如表 3-1 所示。

表 3-1　整数类型的 4 种进制表示

进 制 类 型	引 导 符 号	组 成 数 字	示　　例
十进制	无	0～9	123，－123
二进制	0b/0B	0，1	0b0111 1011，－0b0111 1011
八进制	0o/0O	0～7	0o173，－0o173
十六进制	0x/0X	0～9，a/A～f/F	0x7b，－0x7b

3.2 浮点数类型

浮点数就是数学中的小数，在 Python 中，浮点数的取值范围是 $-10^{308}\sim10^{308}$，精度是 10^{-16}。浮点数有两种表示方法，即一般表示法和科学记数法，如表 3-2 所示。

表 3-2 浮点数的两种表示方法

表示方法	符号	示例
一般表示法	无	1234.56789，−1234.56789
科学记数法	e/E	1.234568e+03，1.234568E+03 −1.234568e+03，−1.234568E+03

注意：由于浮点数在计算机中存储时的特殊方式，在进行运算时，存在不确定的尾数，如下面示例所示。

【程序源码】(LX0301.py)

```
1.  f1 = 0.1
2.  f2 = 0.2
3.  print(f1 + f2)
4.  print(f1 + f2 == 0.3)
```

【运行结果】

```
0.30000000000000004
False
```

3.3 复数类型

复数类型类似于数学中的复数，由实数部分、虚数部分和虚数单位 j/J 所组成，例如 1+2j、1+2J。复数的运算属于数学中的复变函数部分，主要用于科学计算。

对于一个复数 c，c.real 和 c.imag 可以分别获取这个复数 c 的实数部分和虚数部分。

3.4 布尔类型

布尔类型也叫逻辑类型，用于表示逻辑判断的结果，逻辑真——True，逻辑假——False。常用的布尔运算如表 3-3 所示。

表 3-3　常用的布尔运算

运算方法	运　算　符	示　　例
非	not	not True → False，not False → True
与	and	True and True → True，True and False → False False and True → False，False and False → False
或	or	True or True → True，True or False → True False or True → True，False or False → False

True 和 False 也可以参加算术运算，在运算时，True 为 1，False 为 0。

3.5　字符串类型

字符串是指用定界符一对单引号(')或双引号(")括起来的一串字符，如'The Zen of Python, by Tim Peters',"Python 之禅"。

(1) 定界符要成对使用，不能混合使用。

(2) 如果字符串中已经有某一种定界符，则必须使用另一种定界符。

(3) 空串：长度为零的字符串，或者说不包含任何字符的字符串叫作空串。

(4) 子串：如果一个字符串 A 完整包含在另一个字符串 B 中，则可以说 A 是 B 的一个子串。

(5) 多行字符串：一对单引号(')或一对双引号(")只能创建单行字符串，一对三个单引号(''')或一对三个双引号(""")可以创建多行字符串。

(6) 转义字符：以符号(\)开头，用于表示具有特定含义或者无法直接输入的特殊字符的字符，常用的转义字符及含义如表 3-4 所示。

表 3-4　常用的转义字符及含义

序号	转义字符	含　　义
1	\\	\
2	\'	'
3	\"	"
4	\a	响铃
5	\b	退格
6	\f	换页
7	\n	换行
8	\r	回车
9	\t	水平制表位
10	\v	垂直制表位

注意：某些转义字符在不同平台上效果略有不同，甚至没有效果，即不支持。

(7) 字节型字符串：以前缀 b 开头并且只能使用 ASCII(American Standard Code for Information Interchange,美国信息交换标准代码)字符的字符串,如 b'The Zen of Python'。

(8) 中文型字符串：以前缀 u 开头,后面字符以 Unicode 格式进行编码的字符串,一般用在中文字符串中,防止乱码,如 u"计算生态"。

(9) 普通型字符串：以前缀 r 开头的字符串,去掉了转义字符的功能,如 r'\\The Zen of Python\\'。

3.6 字符数据处理

字符数据的处理在程序设计中特别是非数值计算的程序设计中应用非常广泛,相比较于数值计算,字符数据的处理也较为复杂,需要更多的方法和技巧来实现。

3.6.1 字符串索引

在 Python 中,字符串实质上是一种有序的序列类型,对于某一个字符串,字符串中字符的顺序是固定的,并且有正向索引和反向索引两种编号方式,通过任何一种索引都可以对字符串的单个字符或部分字符进行引用。假如有一个字符串 s = "The Zen of Python",其索引如图 3-2 所示。

正向索引	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
字符串	T	h	e		Z	e	n		o	f		P	y	t	h	o	n
反向索引	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

图 3-2 字符串索引

3.6.2 字符串引用

1. 单字符引用

引用字符串中单个字符的格式为

```
< 字符串名 >[< 索引号 >]
```

索引号使用正向索引或反向索引都可以,如 s[4]和 s[-13]均为'Z'。

2. 字符串分片

字符串分片,也叫字符串切片或者取子串,就是从原字符串中按照要求取出一部分字符组成一个新的字符串,字符串分片的格式为

```
< 字符串名 >[[ start [: end [: step ]]]
```

在字符串中,从 start 到 end 的前闭后开的区间内,以 step 为步长,取出字符生成一个

新的字符串，start 默认值为 0，end 默认值为字符串长度，step 默认值为 1。

【程序源码】(LX0302.py)

```
s = "The Zen of Python"
print(s[4], s[-13])
print(s[4:7])
print(s[11:])
print(s[:3])
print(s[11:17:2])
print(s[-13:-10])
print(s[-11:-14:-1])
print(s[: : -1])
```

【运行结果】

```
Z Z
Zen
Python
The
Pto
Zen
neZ
nohtyP fo neZ ehT
```

3.6.3 字符串处理

1. 字符串运算

(1) + ：两个字符串首尾连接生成新的字符串；

(2) s * n 或 n * s ：把字符串重复 n(正整数)次生成新的字符串。

【程序源码】(LX0303.py)

```
s1 = "The Zen of Python, "
s2 = "by Tim Peters. "
print(s1 + s2)
print(s1 * 3)
print(3* s2)
```

【运行结果】

```
The Zen of Python, by Tim Peters.
The Zen of Python, The Zen of Python, The Zen of Python,
by Tim Peters. by Tim Peters. by Tim Peters.
```

2. 字符串常用函数

(1) str(object)：返回一个对象的 string 格式，即转换成一个字符串类型；

(2) len(s)：返回字符串的长度；

(3) chr(i)：返回参数 i 对应的 Unicode 字符，i 的取值范围为 0 到 1,114,111；

(4) ord(c)：返回字符 c 对应的 Unicode 值。

【程序源码】(LX0304.py)

```
a1 = 123
a2 = 1234.56789
a3 = 1 + 2j
a4 = True
s = "The Zen of Python"
print(str(a1))
print(str(a2))
print(str(a3))
print(str(a4))
print(len(s))
print(chr(65),ord('A'))
```

【运行结果】

```
123
1234.56789
(1+2j)
True
17
A 65
```

3. 字符串常用方法

字符串的常用方法如表 3-5 所示。

表 3-5　字符串常用方法

序号	方　　法	功 能 描 述	备注
1	s.capitalize()	字符串 s 的首字母大写，其余小写	
2	s.title()	字符串 s 中每个单词的首字母大写，其余小写	单词由空格分隔
3	s.swapcase()	字符串 s 中字母大小写转换	
4	s.upper()	字符串 s 中的所有字母转换成大写	
5	s.lower()	字符串 s 中的所有字母转换成小写	
6	s.index(x[, m[, n]])	从左向右查找，返回子串 x 在字符串 s 中指定范围内(前闭后开)第一次出现的位置；范围默认为整个字符串；没有找到则抛出异常	
7	s.rindex(x[, m[, n]])	从右向左查找，返回子串 x 在字符串 s 中指定范围内(前闭后开)第一次出现的位置；范围默认为整个字符串；没有找到则抛出异常	

续表

序号	方　法	功 能 描 述	备注
8	s.find(x[, m[, n]])	从左向右查找，返回子串 x 在字符串 s 中指定范围内(前闭后开)第一次出现的位置；范围默认为整个字符串； 没有找到则返回－1	
9	s.rfind(x[, m[, n]])	从右向左查找，返回子串 x 在字符串 s 中指定范围内(前闭后开)第一次出现的位置；范围默认为整个字符串； 没有找到则返回－1	
10	s.count(x[, m[, n]])	返回子串 x 在字符串 s 中指定范围内(前闭后开)出现的次数； 范围默认为整个字符串	
11	s.startswith(x[, m[, n]])	判断一个字符串 s 中指定范围(前闭后开)是否以子串 x 开头； 范围默认为整个字符串	
12	s.endswith(x[, m[, n]])	判断一个字符串 s 中指定范围(前闭后开)是否以子串 x 结尾； 范围默认为整个字符串	
13	s.replaces(u, v[, n])	在字符串 s 中用子串 v 替换子串 u 共计 n 次； 默认为全部替换	
14	s.expandtabs([n])	在字符串 s 中，用 n 个空格替换制表位字符； 默认为 8 个空格	
15	s.strip([x])	去掉字符串 s 中前后的子串 x； 默认为空格	
16	s.lstrip([x])	去掉字符串 s 中前面的子串 x； 默认为空格	
17	s.rstrip([x])	去掉字符串 s 中后面的子串 x； 默认为空格	
18	s.join(l)	l 是一个字符串列表，s 是一个字符串，用 s 把 l 中所有字符串连接起来，生成一个新的字符串	
19	s.split(c[, n])	把字符串 s 以字符串 c 拆分 n 次，返回一个由拆分项所组成的字符串列表； 无参数 n 则全部拆分	
20	s.splitlines(y)	如果 s 是一个多行字符串，把 s 的每一行作为一个元素，返回一个字符串列表；如果 s 是单选字符串，则返回的列表中只有一个元素； y 为 True，则保留换行符，为 False 则去掉换行符	
21	s.isspace()	判断一个字符串 s 中的字符是否全为空格，返回 True 或 False	
22	s.isdigit()	判断一个字符串 s 中的字符是否全为数字，返回 True 或 False	
23	s.isalpha()	判断一个字符串 s 中的字符是否全为字母，返回 True 或 False	
24	s.isalnum()	判断一个字符串 s 中的字符是否全为字母或数字，返回 True 或 False	

续表

序号	方　法	功能描述	备注
25	s.istitle()	判断一个字符串 s 中的每个单词是否首字母大写，其余为小写，返回 True 或 False	单词由空格分隔
26	s.isupper()	判断一个字符串 s 中的字符是否全为大写，返回 True 或 False	
27	s.islower()	判断一个字符串 s 中的字符是否全为小写，返回 True 或 False	
28	s.center(w[, c])	以 w 为总宽度，字符串 s 居中，两边填充字符 c，返回一个新的字符串； 无参数 c，则以空格进行填充	
29	s.ljust(w[, c])	以 w 为总宽度，字符串 s 左对齐，右边填充字符 c，返回一个新的字符串； 无参数 c，则以空格进行填充	
30	s.rjust(w[, c])	以 w 为总宽度，字符串 s 右对齐，左边填充字符 c，返回一个新的字符串； 无参数 c，则以空格进行填充	
31	s.zfill(w)	以 w 为总宽度，字符串 s 右对齐，左边填充字符 0，返回一个新的字符串	

3.7　常量与变量

3.7.1　常量

常量是指在程序运行过程中其值不会发生改变的数据，例如 123、1234.56789、1＋2j、True、'The Zen of Python'分别是一个整数常量、浮点数常量、复数常量、布尔常量和字符串常量。

3.7.2　变量

变量是一个标识符(变量名)，用来指向一个常量(即变量的值)，在程序运行过程中，其数据类型和值可以根据需要发生改变。Python 中的变量在程序中不必提前定义，在使用的同时进行定义即可。

1. 变量赋值

变量赋值就是通过赋值语句让变量名指向一个常量，从而获得一个数据类型和值。Python 中赋值符号为等号“＝”，有 3 种赋值格式。

(1) 单个变量赋值：＜变量＞ ＝ ＜表达式＞。

例如：“a ＝ 123”，就是让变量名为 a 的变量指向一个值为 123 的整数类型常量，如图 3-3 所示。此时变量 a 的类型为整数类型，值

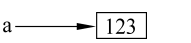

图 3-3　单个变量赋值

为 123。

(2) 多个变量赋同一个值：<变量 1 = … = 变量 n> = <表达式>。

例如："a1 = a2 = a3 = 123"，就是让变量 a1、a2、a3 均指向一个值为 123 的整数类型常量，如图 3-4 所示。此时，这 3 个变量的类型均为整数类型，值都为 123。

(3) 多个变量依次赋不同的值：<变量 1，…，变量 n> = <表达式 1，…，表达式 n>。

例如："a1，a2，a3 = 123，456，789"，就是让变量 a1 指向一个值为 123 的整数类型常量，变量 a2 指向一个值为 456 的整数类型常量，变量 a3 指向一个值为 789 的整数类型常量，如图 3-5 所示。

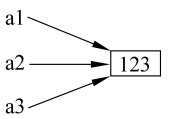

图 3-4　多个变量赋同一个值

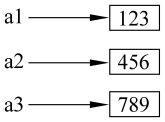

图 3-5　多个变量依次赋不同的值

2. 函数 id()和 type()

在 Python 中，系统会为每一个常量分配一个内存单元，内存单元的地址是这个内存单元在物理内存中的具体位置，内存单元的内容是存放在这个内存单元中的常量。

变量赋值就是让变量指向一个常量的内存地址。

(1) id(对象)：获取变量所指向常量的内存单元地址，其返回值是一个整数类型的数据。为了方便理解，也可以称为变量的地址。

(2) type(对象)：获取变量所指向常量的数据类型，其返回值是一个对象类型的数据。为了方便理解，也可以称为变量的数据类型。

【程序源码】(LX0305.py)

```
1.  a = 123
2.  b1 = b2 = b3 = 1234.56789
3.  c1, c2, c3 = 1 + 2j, True, 'The Zen of Python'
4.  print(a, b1, b2, b3, c1, c2, c3)
5.  print(id(a), id(b1), id(b2), id(b3), id(c1), id(c2), id(c3))
```

【运行结果】

```
123 1234.56789 1234.56789 1234.56789 (1+2j) True The Zen of Python
1842248256 2841503996160 2841503996160 2841503996160 2841526797008
1841754272 2841526943224
```

注意：

(1) Python 在程序运行时会实时为每个常量分配一个内存单元，不再使用的常量系统会自动回收这个内存单元。同一个源程序，每次运行时 id()的返回值不尽相同；在不同的计算机上运行，id()的返回值也不尽相同；

(2) Python 为了提高程序的运行效率，将整数类型的常量－5～256 和布尔类型的常量 True、False 常驻内存，不必每次再进行内存单元的分配和回收；

(3) Python 支持变量的内存自动管理机制，也可以通过语句“del ＜变量 1，变量 2，…，变量 n＞”人工删除变量。

3. 动态数据类型

变量的数据类型和值由赋值时表达式的数据类型和值所决定，Python 语言中的变量，不仅其值可以发生改变，其数据类型也可以随之发生改变，称之为动态数据类型的变量，这也是 Python 语言的一个特色。

【程序源码】(LX0306.py)

```
1.  a = 123
2.  print(a, type(a), id(a))
3.  a = 456
4.  print(a, type(a), id(a))
5.  a = 1234.56789
6.  print(a, type(a), id(a))
7.  a = 1 + 2j
8.  print(a, type(a), id(a))
9.  a = True
10. print(a, type(a), id(a))
11. a = 'The Zen of Python'
12. print(a, type(a), id(a))
```

【运行结果】

```
123 <class 'int'> 1842248256
456 <class 'int'> 2153907313808
1234.56789 <class 'float'> 2153903893168
(1+2j) <class 'complex'> 2153908471440
True <class 'bool'> 1841754272
The Zen of Python <class 'str'> 2153908488424
```

3.8 运算符与表达式

3.8.1 运算符及优先级

Python 语言的运算符及优先级如表 3-6 所示。

表 3-6 运算符及优先级

优先级	运 算 符	功 能	说 明
1	()	括号	提高优先级
2	**	幂/乘方	
3	～	按位取反	二进制且补码运算规则

续表

优先级	运 算 符	功 能	说 明
4	+、-	正号、负号	正号可以省略
5	*、/、%、//	乘、除、取模、整除	除运算结果为浮点数
6	+、-	加、减	
7	>>、<<	右移、左移	二进制且补码运算规则；左移时低位补0，右移时正数高位补0，负数高位补1
8	&	按位与	二进制且补码运算规则
9	^、|	按位异或，按位或	二进制且补码运算规则
10	<=、<、>、>=	比较运算：小于或等于、小于、大于、大于或等于	
11	==、!=	等于、不等于	只比较值
12	=、% =、/=、//=、-=、+=、*=、**=	赋值运算	
13	is、is not	身份运算：是、不是	既比较值，又比较内存地址
14	in、not in	成员运算：在、不在	
15	and or not	逻辑运算：与、或、非	

3.8.2 表达式

表达式就是用运算符将常量、变量、函数等按照 Python 语言的语法规则连接起来的一个有意义的式子。根据运算符和运算对象的不同，表达式可分为算术表达式、字符串表达式、关系表达式和逻辑表达式。

3.9 单元拓展：内置函数

3.9.1 函数简介

函数是一个具有独立功能的程序段，可以反复调用，提高程序设计的效率和简洁性。函数的三个基本要素是功能、参数和返回值，当然有个别函数没有参数，也有个别函数没有返回值。一个函数有没有参数，其圆括号都不可省略。

Python 的函数分为三大类：系统函数、内置函数和用户函数。

1. 系统函数

系统函数是 Python 自带的函数，它的调用者不是用户，而是 Python 系统本身，如构造函数等。

2. 内置函数

内置函数也是 Python 自带的函数，不需要用户去定义，可以直接调用。

3. 自定义函数

自定义函数是用户根据实际需要通过 def 关键字自己创建的实现某种特定功能的函数。

3.9.2 内置函数

Python 的内置函数丰富且强大，如表 3-7 所示，熟练掌握内置函数并且灵活应用于程序设计中，可以简化算法和程序的设计，提高程序设计的效率。

表 3-7 内置函数一览表

类别	编号	函数名	功能描述	示例
运算函数	1	abs(x)	返回一个数的绝对值。 参数可以是一个整数或浮点数，如果参数是一个复数，则返回它的模	abs(－123) → 123 abs(3＋4j) → 5.0
	2	divmod(a, b)	接收两个数字类型(非复数)的参数，返回一个包含商和余数的元组(a // b, a % b)	divmod(9,2) → (4, 1)
	3	max(x, key＝None)	返回可迭代对象中最大的元素，或者返回两个及以上实参中的最大值； 如果有多个最大值，则返回第一个值； 可以通过 key 返回最大值	max([1,2,3,4]) → 4 max(1,2,3,4) → 4 max([－1,－2,0,1],key＝abs) → －2
	4	min(x, key＝None)	返回可迭代对象中最小的元素，或者返回两个及以上实参中的最小值； 如果有多个最小值，则返回第一个值； 可以通过 key 返回最小值	min([1,2,3,4]) → 1 min(1,2,3,4) → 1 max([－1,－2,0,1],key＝abs) → 0
	5	pow(x, y[, mod])	返回 x 的 y 次方，如果 mod 存在，则再对结果进行取模	pow(3,2) → 9 pow(3,2,2) → 1
	6	round(x[,n])	返回浮点数 x 的四舍五入值，n 为小数的位数，无 n 则取整	round(123.456789, 3) → 123.457 round(123.456789) → 123
	7	sum(iterable[, start])	对 iterable 中的所有项进行求和，返回再与 start 相加的值； start 默认为 0	sum([1,2,3,4],10) → 20 sum([1,2,3,4])→ 10
类型转换函数	1	bool()	将一个整数 0、浮点数 0.0，复数 0＋0j 和空字符串转换为 False，其他的值转换为 True	bool(123) →True bool(0.0) →False
	2	complex([real[, imag])	返回值为 real ＋ imag * 1j 的复数，或将字符串或数字转换为复数； 如果第一个参数是字符串，则它被解释为一个复数，并且不能有第二个参数； 无参数则返回 0j	complex(1,2) → (1＋2j) complex('123') → (123＋0j) complex() → 0j

续表

类别	编号	函数名	功能描述	示例
类型转换函数	3	float([x])	将整数和字符串转换为浮点数； 无参数则返回 0.0	float(123) → 123.0 float('123.456') 123.456 float() → 0.0
	4	int([x])	将一个字符串或数字转换为整型； 没有参数则返回 0	int('123') → 123 int(123.456) → 123 int() → 0
	5	str(object)	返回一个对象的 string 格式	str(123) → '123'
	6	bytearray([source[, encoding[, errors]]])	如果 source 为整数，则返回一个长度为 source 的初始化数组； 如果 source 为字符串，则必须提供 encoding 参数，并按照指定的 encoding 将字符串转换为字节序列； 如果 source 为可迭代类型，则元素必须为[0,255]中的整数； 如果没有任何参数，则创建大小为 0 的数组	
	7	bytes([source[, encoding[, errors]]])	返回一个新的 bytes 对象，它是 bytearray() 的不可变版本	
	8	memoryview(object)	返回对象的内存查看对象，简单理解为对内存地址的直接访问	
进制转换函数	1	bin()	将一个整数转变为一个前缀为“0b”的二进制字符串	bin(123) → '0b1111011'
	2	hex(x)	将整数转换为以“0x”为前缀的小写十六进制字符串	hex(123) → '0x7b'
	3	oct(x)	将整数转换成八进制的字符串	oct(123) → '0o173'
	4	ord(c)	返回 c 对应的 Unicode 值	ord('A') → 65
	5	chr(i)	返回参数 i 对应的 Unicode 字符。 i 的取值范围为 0 到 1,114,111	chr(65) → 'A'
序列操作函数	1	tuple(seq)	将可迭代系列转换为元组； 无参数则返回一个空元组	tuple([1,2,3,4])→ (1, 2, 3, 4) tuple()→ ()
	2	list()	将元组或字符串转换为列表； 无参数则返回一个空列表	list((1,2,3))→ [1, 2, 3] list('China') → ['C', 'h', 'i', 'n', 'a'] list()→ []
	3	set([iterable])	创建一个集合； 无参数则创建一个空集合	set([1,2,3,4]) → {1, 2, 3, 4} set()→ set()
	4	frozenset([iterable])	返回一个冻结的集合，冻结后集合不能再添加或删除任何元素，也叫不可变集合	frozenset ([1, 2, 3]) → frozenset({1, 2, 3})

续表

类别	编号	函数名	功能描述	示例
序列操作函数	5	dict()	创建一个字典； 无参数则返回一个空字典	dict(sid="1001",sname="胡凡林",sage=20)→{'sid': '1001', 'sname': '胡凡林', 'sage': 20} dict()→ dict()
	6	range([start,] stop[, step])	返回一个从 start 到 stop 之间，步长为 step，前闭后开的可迭代对象； 无 start 则从 0 开始，无 step 则步长为 1	list(range(5))→ [0, 1, 2, 3, 4]
	7	enumerate(sequence, [start=0])	将一个可遍历的数据对象（如列表、元组或字符串）组合为一个索引序列，同时列出下标和数据	list(enumerate(['red','green','blue'],start = 1)) → [(1, 'red'), (2, 'green'), (3, 'blue')]
	8	iter(object[, sentinel])	返回一个 iterator 对象	
	9	slice(start, stop[, step])	返回一个切片对象，主要用在切片操作函数里的参数传递	
	10	object()	返回一个没有特征的新对象	
	11	super(type[, object-or-type])	调用超类，用来解决多重继承问题	
排序函数	1	sorted(iterable, key=None, reverse=False)	对所有可迭代的对象进行排序，默认为升序，返回一个列表	sorted([2,1,4,3]) → [1, 2, 3, 4]
	2	reversed(seq)	返回给定序列的反向迭代器	list(reversed([1,2,3,4])) → [4, 3, 2, 1]
迭代对象函数	1	all(iterable)	如果一个可迭代对象 iterable 的所有元素均为 True 或者 iterable 为空，则返回 True，否则返回 False	all([ ]) →True all([1,2,3,4,5]) →True all([0,1,2,3,4]) →False
	2	any(iterable)	如果一个可迭代对象 iterable 的任一元素为 True，则返回 True，否则返回 False，如果 iterable 为空也返回 False	any([ ]) →False any([0,1,2,3,4]) →True any([0,0,0,0,0]) →False
	3	map(function, iterable, …)	返回一个将 function 应用于 iterable 中每个元素并输出其结果的迭代器	list(map(lambda x: x * x, [1,2,3,4])) → [1, 4, 9, 16]
	4	filter(function, iterable)	用于过滤序列，过滤掉不符合条件的元素，返回一个迭代器对象	list(filter(lambda x: x%2==0,[1,2,3,4])) → [2, 4]
	5	next(iter[,default])	返回迭代器的下一个元素。 如果迭代器耗尽，则返回给定的 default，否则触发异常	
	6	zip([iterable,…])	将对象中对应的元素打包成一个个元组，返回由这些元组构成的一个可迭代对象	list(zip([1,2,3],['a','b','c'],[1.1, 2.2, 3.3, 4.4])) → [(1, 'a', 1.1), (2, 'b', 2.2), (3, 'c', 3.3)]

续表

类别	编号	函数名	功能描述	示例
对象元素操作函数	1	help([object])	查看对象的帮助信息	help(print)
	2	id([object])	返回对象的内存地址	id(123) → 1842248256
	3	type(object)	返回对象的类型； type(name, bases, dict)返回一个新的类型对象	type(123) → <class 'int'>
	4	dir([object])	如果没有参数，则返回当前范围中的名称；带参数时，返回对象的属性、方法列表	dir(print)
	5	len(object)	返回对象的长度	len('China')→ 5
	6	hash(object)	返回对象的哈希值	
	7	ascii(object)	返回对象的纯 ASCII 表示形式	ascii('China') →"'China'" ascii('China-中国') → "'China-\\u4e2d\\u56fd'"
	8	repr(object)	返回包含一个对象的可打印表示形式的字符串	repr(123)→ '123'
	9	vars([object])	返回对象的属性和属性值的字典对象	
	10	format(value[, format_spec])	格式化字符串	
属性操作函数	1	isinstance(object, classinfo)	判断一个对象是否是一个已知的类型	isinstance(123,int)→ True isinstance(123,str)→ False
	2	issubclass(class, classinfo)	判断参数 class 是否是类型参数 classinfo 的子类	
	3	getattr(object, name[, default])	返回 object 对象的 name 属性的值； 如果指定的属性不存在，但是提供了 default 值，则返回它，否则触发异常	
	4	hasattr(object, name)	如果 name 是对象 object 的属性名之一，则返回 True，否则返回 False	
	5	setattr(object, name, value)	给对象的属性设置属性值	
	6	delattr(object,name)	删除对象 object 的名为 name 的属性	
	7	callable(object)	检查一个对象是否可调用，可调用则返回 True，否则返回 False	
	8	__import__(name[, globals [, locals [, fromlist[, level]]]])	动态加载类和函数	
变量函数	1	globals()	返回包含当前作用域的全局变量的字典	
	2	locals()	以字典类型返回当前位置的全部局部变量	

续表

类别	编号	函数名	功能描述	示例
人机交互函数	1	print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)	将 objects 打印到 file 指定的文本流，默认为 sys.stdout	print("Hello World") → Hello World
	2	input([prompt information])	接受一个标准输入数据，返回为 string 类型	
	3	open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)	打开一个文件，并返回文件对象	
编译函数	1	compile(source,filename, mode, flags=0, dont_inherit=False, optimize=-1)	将 source 编译成代码或 AST 对象。代码对象可以被 exec() 或 eval() 执行	
	2	eval(expression [, globals[, locals]])	执行一个字符串表达式，并返回表达式的值	eval("1+2*3") → 7
	3	exec(object[, globals[, locals]])	执行存储在字符串或文件中的 Python 语句	exec("print('Hello World')") → Hello World
装饰器函数	1	breakpoint(*args, **kws)	调用 sys.breakpointhook()，直接传递 args 和 kws，进入 pdb 调试器	
	2	classmethod()	将一个方法封装成类方法，该方法不需要实例化，也不需要 self 参数，第一个参数是表示自身类的 cls 参数，可以用来调用类的属性和方法	
	3	property([fget[, fset[, fdel[, doc]]]])	在新式类中返回属性值	
	4	staticmethod()	将方法转换为静态方法	

3.10 项目训练

3.10.1 变量交换

(1) 项目编号：XMXL0301。

(2) 项目要求：交换两个变量的值，打印输出原值、类型、地址和交换后的值、类型和地址。

(3) 程序源码。

```
1.  #-*- coding:UTF-8 -*-
```

```
"""
项目编号:XMXL0301
交换两个变量的值,打印输出原值、类型、地址和交换后的值、类型和地址
"""

a = 123
b = "China"
print(a, b)
print(type(a), type(b))
print(id(a), id(b))
a, b = b, a
print(a, b)
print(type(a), type(b))
print(id(a), id(b))
```

(4) 运行结果。

```
123 China
<class 'int'> <class 'str'>
1437039168 1325127422728
China 123
<class 'str'> <class 'int'>
1325127422728 1437039168
```

3.10.2 计算 BMI

(1) 项目编号：XMXL0302。

(2) 项目要求：BMI(Body Mass Index,身体质量指数、身体体质指数、身体体重指数)，用身体体重千克数除以身高米数的平方所得出的一个数字，是国际上常用的衡量人体胖瘦程度以及是否健康的一个标准，如图 3-6 所示。

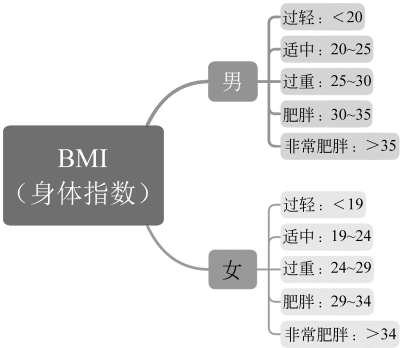

图 3-6　BMI 标准

(3) 程序源码。

```
#-*- coding:UTF-8 -*-
"""
项目编号:XMXL0302
项目要求:已知一个人的身高 h(m)和体重 g(kg),计算其 BMI 并且输出
"""

h = 1.78
g = 73.5
BMI = g / (h * h)
print(BMI)
```

(4) 运行结果。

```
23.197828557000378
```

3.10.3 查看关键字

(1) 项目编号：XMXL0303。

(2) 项目要求：调用 Python 标准库 keyword,显示当前 Python 版本的所有关键字,并且统计个数。

(3) 程序源码。

```
#-*- coding:UTF-8 -*-
"""
项目编号:XMXL0303
项目要求:调用 Python 标准库 keyword,显示当前 Python 版本的所有关键字,并且统计个数
"""

import keyword
python_kw = keyword.kwlist
print(python_kw)
print(len(python_kw))
```

(4) 运行结果。

```
['False', 'None', 'True', 'and', 'as', 'assert', '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']
33
```

注意：keyword.kwlist 返回的是一个字符串列表类型的数据。

3.11 习　　题

1. 判断题

(1) Python 变量使用前必须先声明，并且一旦声明就不能在当前作用域内改变其类型。（　　）

(2) Python 不允许使用关键字作为变量名，允许使用内置函数名作为变量名，但是这会改变原函数名的含义。（　　）

(3) Python 变量名必须以字母或下画线开头，并且字母区分大小写。（　　）

(4) 只有 Python 第三方库需要导入以后才能使用其中的对象和方法，标准库不需要导入就可以使用所有的对象和方法。（　　）

(5) 使用 import 语句可以一次导入任意多个标准库或第三方库。（　　）

(6) ceil(x)返回大于或等于 x 的最小整数(向上取整)。（　　）

(7) floor(x)返回小于或等于 x 的最大整数(向下取整)。（　　）

(8) 字符串函数 s.capitalize()返回的单词仅首字母大写，其余字母小写；s.title()返回的单词首字母大写，其余字母小写。（　　）

(9) 字符串函数 s.find(t[m[n]])返回 t 在 s 中首次出现的位置，否则返回－1。（　　）

(10) 字符串函数 s.index(t[m[n]])返回 t 在 s 中首次出现的位置，否则返回－1。（　　）

2. 单选题

函数 bin()、oct()和 hex()把十进制整数转换为二进制、八进制和十六进制，转换之后的数据是(　　)。

A. 整数类型

B. 浮点型类型

C. 字符串类型

D. 对应的二进制型、八进制型和十六进制型

第4章

控制结构与异常处理

Python 语言既支持结构化程序设计方法，也支持面向对象程序设计方法。在结构化程序设计方法中，程序由顺序结构、分支结构和循环结构三种基本结构有机构成。

4.1 三种基本结构

4.1.1 顺序结构

在结构化程序设计中，顺序结构是最基本，也是使用最多的结构，程序从上到下依次执行，其执行流程如图 4-1 所示。

4.1.2 分支结构

分支结构是首先判断一个表达式的值，然后根据表达式值的真假选择性执行某一分支。

1. 单分支结构

单分支结构是如果表达式的值为真则执行相应的分支语句或语句块，如果表达式的值为假则直接退出分支结构，其流程图如图 4-2 所示。

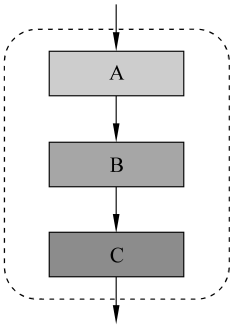

图 4-1 顺序结构

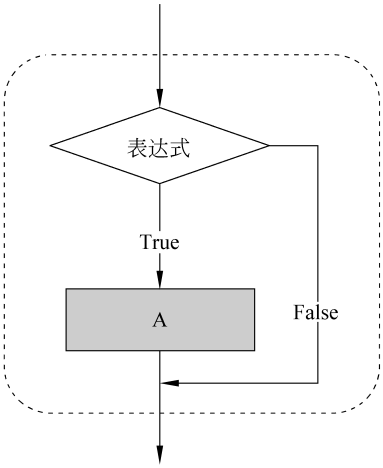

图 4-2 单分支结构