第3章 chapter 3

Python的基本数据类型

导读

数据即变量的值，变量是用来反映/保持状态以及状态变化的。在程序设计语言中，针对不同的状态就用不同类型的数据去标识。本章介绍Python的数字、字符串、布尔、列表、元组、字典与集合等基本类型的数据。Python最大的特色就是引入了列表、元组、字典与集合等特色的数据类型，这些类型数据的使用能改变读者的编程思维。

3.1 数值数据

数值数据类型用于存储数值。在Python中，数值数据属于不可变数据类型。不可变数据类型就是当为变量分配数值数据时，Python将创建数值对象，且把变量指向创建的数值对象，当更改指向数值对象变量的值时，Python就重新创建新值的数值对象，把变量重新指向新的数值对象，而数值对象本身却不能被修改。例如：

```
>>>var1=10      #在栈内存中创建 var1 变量,在堆内存中创建数值对象 10,且把 var1 指向
                #10 对象
>>>id(var1)     #返回 var1 指向的对象的地址
1742209776      #说明数值对象 10 的起始地址是 1742209776
>>>var1=20      #在堆内存中创建数值对象 20,且把 var1 指向 20 数值对象,原来的 10 仍然
                #还在
>>>id(var1)     #返回 var1 指向的对象的地址
1742210096      #说明 var1 指向的数值对象的地址是 1742210096,不再是数值对象 10 的
                #地址
#说明上述 var1 不是指向同一个对象
```

可以使用del命令删除对数值对象的引用。del语句的语法：

```
del var1[,var2[,var3[…,varN]]]
```

见如下示例：

```
>>>var1=20
>>>id(var1)
1742210096
>>>del var1
>>>id(var1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'var1' is not defined
```

此例说明删除 var1 后，显示 var1 对象的地址时报错“name 'var1' is not defined”。说明该对象已经不存在了。

3.1.1 Python 3 支持的数值数据类型

Python 3 支持的数值数据类型有带符号整数、浮点实数与复数 3 种。

(1) int(带符号整数)：该类型通常被称为整数，是没有小数点的正或负数。

注意：Python 3 中的整数是不限大小的。

(2) float(浮点实数)：该类型也被称为浮点数，表示实数。例如，2.5、3.14 就是一个浮点数，浮点数也可以用科学符号表示，如+2.5E3 或+2.5e3，表示 2.5 乘 10 的三次方。

(3) complex(复数)：复数是以 a+bj 的形式表示，其中 a 和 b 是浮点，J(或 j)表示 −1 的平方根(虚数)。数据的实部是 a，虚部是 b。复数在 Python 编程中用得比较少，了解一下就可以了。

在 Python 中可以用十六进制或八进制形式表示整数，如果用十六进制表示整数，就在十六进制数前加上 0x 或 0X，如果用八进制表示整数，就在八进制数前加上 0o 或 0O，见下面例子：

```
>>>number =0xA0F #Hexa-decimal
>>>number
2575
>>>number =0o37 #Octal
>>>number
31
```

3.1.2 Python 3 中与数值相关函数

1. 数值类型转换函数

在 Python 编程过程中，有时需要把数据从一个类型转化到另一个类型，以满足运算符或函数参数的要求。Python 中有如下两个常用函数用于数据类型的转换，如表 3.1 所示。

表 3.1 数值类型转换函数

函　数	描　述	示　例
int(x)	将 x 转换为纯整数	int(3.14)返回结果是整数 3。
float(x)	将 x 转换为浮点数	float(3)返回结果是浮点数 3.0。

2. 数值函数

Python 中常见的一些数值函数如表 3.2 所示。

表 3.2 数值函数

编号	函　数	描　述
1	abs(x)	返回 x 的绝对值
2	ceil(x)	返回不小于 x 的最小整数
3	exp(x)	返回 e 的 x 次幂
4	floor(x)	返回不大于 x 的最大整数
5	log(x)	返回 x 的自然对数(x ＞ 0)
6	log10(x)	返回以基数为 10 的 x 的对数(x ＞ 0)
7	max(x1,x2,…)	给定参数中的最大值,最接近正无穷
8	min(x1,x2,…)	给定参数中的最小值,最接近负无穷
9	modf(x)	将 x 的分数和整数部分切成两项放入元组中,两个部分与 x 具有相同的符号。整数部分作为浮点数返回
10	pow(x, y)	返回 x 的 y 次幂
11	round(x[,n])	返回 x 从小数点舍入到 n 位数。round(0.5)结果为 1.0, round(-0.5)结果为－1.0
12	sqrt(x)	返回 x 的平方根(x ＞ 0)

3. 三角函数

Python 包括的三角函数如表 3.3 所示。

表 3.3 三角函数

编号	函　数	描　述
1	sin(x)	返回 x 弧度的正弦值
2	cos(x)	返回 x 弧度的余弦值
3	tan(x)	返回 x 弧度的正切值
4	asin(x)	返回 x 的反正弦弧度值
5	acos(x)	返回 x 的反余弦弧度值

续表

编号	函　数	描　述
6	atan(x)	返回x的反正切弧度值
7	degrees(x)	将角度x从弧度转换为度
8	radians(x)	将角度x从度转换为弧度

4. 数学常数

在Python中,可以用到pi与e两个数学常量分别表示圆周率的值与e的值。

注意:在Python中要使用上述函数必须导入math包,导入包有两种方式,一种是用import math,该种导入方式在使用函数时需要在函数前加上包名math.,如math.sqrt(x)。另一种是用"from math import *"命令,用此种方式导入包时,调用math模块中任何函数时,无须在前面加上math,直接使用函数就可以,但使用此种方式导入时,如果用户编写了与math模块相同的函数,就会被导入的math模块中的同名函数覆盖!

5. 随机数函数

随机数用于游戏、模拟、测试、安全和隐私应用。Python包括如表3.4所示的随机数函数。

表3.4　随机数函数

编号	函　数	描　述
1	choice(seq)	返回列表,元组或字符串的随机项目
2	randrange ([start,] stop [,step])	返回从范围(start, stop, step)中随机选择的元素
3	random()	返回随机浮点数r(0 <=r < 1)
4	seed([x])	用来确定生成的随机数,如果使用相同的x值,则生成的随机数相同。函数返回值为None
5	shuffle(lst)	将列表的元素随机化,函数返回值为None
6	uniform(x, y)	返回随机浮点数r (x <=r < y)

注意:在Python中要使用随机数函数必须导入random包。

为了让读者理解shuffle函数与seed函数,在此分别以一个示例进行说明。

本示例使用shuffle函数对列表(list)中的元素进行随机排列。

```
import random
list =[20, 16, 10, 5];
random.shuffle(list)
print("随机排序列表 : ",list)
random.shuffle(list)
```

```
print("随机排序列表 : ",list)
#输出结果为:
随机排序列表:[10, 16, 5, 20]
随机排序列表:[10, 20, 16, 5]
```

本示例说明用 seed 设置相同种子数 10 时,使用 random 函数产生的随机数相同。

```
import random
random.seed( 10 )
print("Random number with seed 10 : ", random.random())
#生成同一个随机数
random.seed( 10 )
print("Random number with seed 10 : ", random.random())
#生成同一个随机数
random.seed(10)
print("Random number with seed 10 : ", random.random())
#输出结果为
Random number with seed 10 :0.5714025946899135
Random number with seed 10 :0.5714025946899135
Random number with seed 10 :0.5714025946899135
```

3.1.3 Python 3 中数值函数的应用

猜数字游戏:编写程序随即生成一个 0～100 的随机整数。程序提示用户输入一个数字,不停猜测,直到猜对为止。最后,输出猜测的数字和猜测的次数。如果用户没有猜中,要提示用户猜的值是大了还是小了。程序代码如下:

```
#author chenzhen
#date 2019/1/7
from random import *
count=0
num =int(random() * 100)+1          #产生 1~100 的随机整数
while 1:
    guess_num=input("请输入猜测的整数[1~100]:")
    if not guess_num.isdigit():  #input 方法接收的是字符串,判断输入是否为数字串
        print("输入的数无效,请重新输入!!")
        continue
    guess_num=int(guess_num)
    count+=1
    if (guess_num ==num):
        print("你猜中啦!!")
        print("你总计猜了%d 次"%count)
```

```
        if count>8:
            print("你猜的次数超过了8次,还需继续努力!!")
        break
    elif guess_num<num:
        print("你猜小啦!!")
    else:
        print("你猜大啦!!")
```

3.2 字 符 串

字符串是Python中最受欢迎、最常用的一种数据类型。在Python中,加引号(单引号或双引号)的字符就是字符串类型,用于标识描述性的内容,如姓名、性别、国籍、种族等。字符串的定义方法如下:

```
str1='hello world!'
str2=str('hello world!')     #用构造方法创建字符串类型数据
```

字符串数据类型与数值数据一样是不可变数据类型,当更改字符串值时会导致重新创建与分配对象。

注意:Python并没有字符类型,字符会被视为长度为1的字符串。

3.2.1 访问字符串中的字符与更新字符串

在Python中要访问子串,需使用方括号的切片加上索引或直接使用索引来获取子字符串。例如:

```
str1="hello world"
print("str1[0:4]:",str1[0:4])          #切片加上索引,不包括索引为4的字符
print("last character is:",str1[10]) #使用索引
    #输出结果为
    str1[0:4]: hell
    last character is: d
```

更新字符串可以通过将变量分配给另一个字符串来更新现有的字符串。新值可以是与其原值相关或完全不同的字符串。例如:

```
str1="hello world"
print(id(str1))
str1=str1[:6] +'Python'
print(id(str1))              #此处的id的值与第二行代码的输出值不一样
```

```
print ("Updated String : ", str1)     #生成"hello" 与'Python'连接生成一个新字符串
#输出结果为
2056835947056
2056835947120
```

3.2.2 转义字符

在 Python 中,可以用反斜杠表示转义或不可打印字符的列表。单引号以及双引号字符串的转义字符被解析。转义字符如表 3.5 所示。

表 3.5 转义字符

反斜线符号	十六进制字符	描述/说明
\a	0x07	铃声或警报
\b	0x08	退格
\n	0x0a	新一行
\nnn		八进制符号,其中 n 在 0～7
\r	0x0d	回车返回
\s	0x20	空格
\t	0x09	制表符
\v	0x0b	垂直制表符
\x		字符 x
\xnn		十六进制符号,其中 n 在 0～9、a～f 或 A～F

看如下示例:

```
>>>print("abcd\babcd\nabcd\tabcd\r  CD")
abcabcd
    CD    abcd
>>>
```

在上面示例中,\b 是退格控制符,当输出完第一个 abcd 后,立即退一格,输出第二个 abcd 时,第二个 abcd 的字符 a 把前一个 abcd 中的 d 覆盖,\n 是新起一行,所以输时就换行输出第三个 abcd,\t 是制表符,此时输出空出 4 个空位,再输出第四个 abcd,\r 是回车符,输出回到当前行的开始位置,输出 CD 把当前行的第一个 abcd 覆盖,输出就完成了。

3.2.3 字符串特殊运算符

假设变量 a 指向字符串'hello',变量 b 指向字串值'Python',表 3.6 说明了字符串特殊运算符的作用和运算结果。

表 3.6 字符串特殊运算符

运算符	说　明	示　例
+	连接,将运算符两边的值连接	a+b 结果为 HelloPython
*	重复,创建新字符串,连接相同字符串的指定副本个数	a * 2 结果为 HelloHello
[]	字符切片,给出指定索引的字符,产生原字符串的子串。注意,字符串的索引从 0 开始	a[1]结果为 e
[:]	范围切片,产生指定范围内的子字符串	a[1:4]结果为 ell
in	成员关系,如果给定字符串中存在指定的字符串,则返回 True	'll' in a 结果为 True
not in	成员关系,如果给定字符串中不存在指定的字符串,则返回 True	'll' not in a 结果为 False
r/R	原始字符串,抑制转义字符的实际含义。原始字符串的语法与正常字符串的格式完全相同,除了原始字符串运算符在引号之前加上字母 r。r 可以是小写(r)或大写(R),并且必须紧靠在第一个引号之前	print(r'\n')将打印\n,或者 print(R'\n') 将打印\n,要注意的是如果不加 r 或 R 作为前缀,打印的结果就是一个换行

3.2.4 字符串格式化运算符

Python 中内置的%运算符可用于格式化字符串操作,控制字符串的输出格式。当格式化字符串时需要一个字符串作为模板,模板中有格式符,这些格式符为真实输出时预留位置。Python 可用一个 tuple 或字典将多个值传递给字符串模板,每个值对应一个格式符,也可以直接用一个变量名来传替一个参数。

使用元组传替参数的格式:

```
格式化字符串 %(参数值 1, 参数值 2, …)
```

说明:格式化字符串中%为占位符,后跟类型码,占位符的位置将用参数值替换。见如下示例:

```
print("I'm %s. I'm %d year old" %('Lili', 19))
#该语句的输出为 I'm Lili. I'm 19 year old
```

在上述 print 方法中的"I'm %s. I'm %d year old" 为字符串模板。%s 为第一个格式符,表示输出时该位置需要用一个字符串取代,%d 为第二个格式符,表示输出时要用一个整数取代。('Lili', 19)是一个元组,元组中的两个元素按顺序分别传替给第一个%s 和第二个%d 的位置。在字符串模板和元组之间有一个%号分隔,它代表进行格式化操作。

在格式化字符串操作中也可以用字典来传参数,见如下例子:

```
print("I'm %(name)s. I'm %(age)d year old" %{'name':'Lili', 'age':19})
#该语句的输出为 I'm Lili. I'm 19 year old
```

该语句中的字符串模板对两个格式符进行了命名，一个命名为 name，另一个命名为 age，且使用括号括起来。第一个命名 name 与字典中的 name 键相同，第二个命名与字典中的 age 键相同。在字典中的 name 键的值为'Lilie'，age 键的取值为 19，输出时就用 Lili 代替了%(name)s，用 19 代替了%(age)d。

当然，如果只传替一个参数，就可以直接使用变量传替参数。见如下示例：

```
name="Lili"
print("I'm %s" %name)
#该语句输出为 I'm Lili
```

表 3.7 列出了可以与%符号一起使用的符号集，即字符串格式的运算符。

表 3.7 字符串格式化运算符

编号	格式化符号	转　　换
1	%s	在格式化之前通过 str()函数转换字符串
2	%i	带符号的十进制整数
3	%d	带符号的十进制整数
4	%u	无符号的十进制整数
5	%o	八进制整数
6	%x，%X	十六进制整数(“x”或“X”)
7	%e，%E	指数符号('e'或'E')
8	%f	浮点实数
9	%g	%f 和%e

3.2.5 字符串的内置方法

Python 提供了很多的内置函数用于处理字符串。下面介绍一些常用的内置函数。

1. center 函数

格式：

```
center(width, fillchar)
```

作用：center 函数返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格。

举例：字符串为 str，填充字符为“&”。

```
>>>str ="this is string example....wow!!!";
>>>print("str.center(40, '&') : ", str.center(40, '&') )
str.center(40, '&') :&&&&this is string example....wow!!!&&&&
```

2. count 函数

格式：

```
count(str, beg =0,end =len(string))
```

作用：统计字符串里某个字符出现的次数。可选参数为在字符串搜索的开始与结束位置。

示例：

```
str ="this is string example....wow!!!";
sub ="i";
print("str.count(sub, 4, 40):", str.count(sub, 4, 40));
sub="wow"
print ("str.count(sub) : ", str.count(sub))
#输出结果为
str.count(sub, 4, 40): 2
str.count(sub) :  1
```

3. encode 函数

格式：

```
encode(encoding =,errors =)
```

作用：把字符串按指定的编码转化为字节码，过程是 str--->(encode)--->bytes。

说明：encoding 用于指定要使用的编码名，如 encoding="UTF-8"。errors 用于设置不同错误的处理方案。默认为 strict，意为编码错误引起一个 UnicodeError。其他可能的值有 ignore、replace 等。

4. decode 函数

格式：

```
decode(encoding =,errors =)
```

作用：把字节码按指定的编码转化为字符串，过程是 bytes--->(decode)--->str。

说明：encoding 用于指定字节码的编码，转化后的编码为 Unicode。

encode 与 decode 方法的使用见如下示例：

```
S ="中国";
S_utf8 =S.encode("UTF-8");
S_gbk =S.encode("GBK");
```

```
print(S);
print("UTF-8 编码:", S_utf8);
print("GBK 编码:", S_gbk);
print("UTF-8 解码:", S_utf8.decode('UTF-8','strict'));
print("GBK 解码:", S_gbk.decode('GBK','strict'));
#输出结果为
中国
UTF-8 编码:b'\xe4\xb8\xad\xe5\x9b\xbd'
GBK 编码:b'\xd6\xd0\xb9\xfa'
UTF-8 解码:中国
GBK 解码:中国
```

5. endwith 函数

格式：

```
endswith(suffix, beg =0, end =len(string))
```

作用：确定字符串或字符串的子字符串(如果启动索引结束和结束索引结束)都以后缀结尾。如果是则返回 True,否则返回 False。注意结束位置要比索引大 1。

示例：

```
Str="Hello world!!"
suffix='!!'
print(Str.endswith(suffix))           #输出为 True
print(Str.endswith(suffix,11))        #输出为 True
print(len(Str))                       #输出为 13
print(Str.endswith(suffix,1,12))      #输出为 False
print(Str.endswith(suffix,1,13))      #输出为 True
```

6. join 方法

格式：

```
join(seq)
```

作用：将序列(seq)中的元素以指定的字符连接生成一个新的字符串。

示例：

```
str ="-";
seq =("a", "b", "c");     #字符串序列
print str.join( seq );
#输出结果
a-b-c
```

7. split 方法

格式：

```
split(str, num)
```

作用：用指定分隔符(str)对字符串进行切片，如果参数 num 有指定值，则分隔 num+1 个子字符串。

示例：

```
str ="Line1-abcdef \nLine2-abc \nLine4-abcd";
print(str.split( ));           #以空格为分隔符,包含 \n
print(str.split(' ', 1 ));     #以空格为分隔符,分隔成两个
#输出结果
['Line1-abcdef', 'Line2-abc', 'Line4-abcd']
['Line1-abcdef', '\nLine2-abc \nLine4-abcd']
```

8. ljust 方法

格式：

```
ljust(width[, fillchar])
```

作用：返回一个原字符串左对齐，并使用指定的填充字符填充至指定长度形成新的字符串。如果指定的长度小于原字符串的长度则返回原字符串。

示例：

```
str ="Hello PYTHON!!";
print (str.ljust(20, '*'));      #输出为 Hello PYTHON!!******
```

9. find 方法

格式：

```
find(str, beg =0 end =len(string))
```

作用：如果启动索引 beg 和结束索引 end 给定，则确定 str 是否在字符串或字符串的子字符串中，如果找到则返回索引，否则为-1。

示例：

```
str1 ="this is string example!!";
str2 ="example";
print(len(str1));            #输出为 24
```

```
print(str1.find(str2));          #输出为 15
print(str1.find(str2, 10));      #输出为 15
print(str1.find(str2, 30));      #输出为-1
```

10. splitlines 方法

格式：

```
splitlines([keepends])
```

作用：按照行('\r', '\r\n',\n')分隔，返回一个包含各行作为元素的列表。如果参数 keepends 为 False，不包含换行符；如果参数 keepends 为 True，则保留换行符。

示例：

```
str1 = 'ab c\n\nde fg\rkl\r\n'
print(str1.splitlines());
str2 = 'ab c\n\nde fg\rkl\r\n'
print(str2.splitlines(True))
#输出结果为
['ab c', '', 'de fg', 'kl']
['ab c\n', '\n', 'de fg\r', 'kl\r\n']
```

11. translate 方法

格式：

```
translate(table [,deletechars])
```

说明：table 是翻译表，翻译表是通过 maketrans 方法转换而来，deletechars 是设置字符串中要过滤的字符列表。

作用：该方法根据参数 table 给出的表(包含 256 个字符)转换字符串的字符，要过滤掉的字符放到 deletechars 参数中，返回翻译后的字符串。若给出了 delete 参数，则将原来的 bytes 中的属于 delete 的字符删除，剩下的字符要按照 table 中给出的映射进行映射。

示例①：

```
intab = "aeiou"
outtab = "12345"
trantab = str.maketrans(intab, outtab)       #制作翻译表
str = "this is string example....wow!!!"
print (str.translate(trantab))
#输出结果为
th3s 3s str3ng 2x1mpl2....w4w!!!
```

示例②：

```
#制作翻译表
bytes _ tabtrans  = bytes. maketrans (b ' abcdefghijklmnopqrstuvwxyz ',  b '
ABCDEFGHIJKLMNOPQRSTUVWXYZ')
#转换为大写,并删除字母 o
print(b'runoob'.translate(bytes_tabtrans, b'o'))
#输出结果为
b'RUNB'
```

12. isdecimal 方法

格式：

```
isdecimal()
```

作用：检查字符串是否只包含十进制字符。这种方法只存在于 unicode 对象。
示例：

```
>>>str1 =u"this2009";
>>>str2 =u"23443434";
>>>print(str1.isdecimal(),str1.isdecimal());
False False
```

3.3 列　　表

列表(list)是有序(有先后顺序之分)的可变(可修改)的元素集合，是 Python 中最基本的数据结构，也是最常用的 Python 数据类型。在列表中的数据项不需要具有相同的类型。列表中的每个元素都有索引值，第一个索引是 0，第二个索引是 1，依此类推，每个元素可以根据索引值来获得。

3.3.1 列表的创建与列表值的访问

列表的定义方法是把逗号分隔的不同的数据项使用方括号定界起来即可。例如：

```
list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
list2=["大学语文",87, "高等数学",84, "英语", 90]          #列表中的元素数据类型不同
list3=[["大学语文",87],["高等数学",84],["英语", 90]]     #列表中的元素为列表
```

访问列表中的值可以使用下标索引，也可以使用方括号的形式截取列表元素：

```
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
>>>list1[1]
'高等数学'
>>>list1[2:4]
['英语', '政治经济学']
>>>list3=[["大学语文",87],["高等数学",84],["英语", 90]]
>>>list3[2]
['英语', 90]
```

如果要把列表(或元组)中的元素赋给变量,如 list4=[1,2,3,4],可以通过如下命令把列表的值赋给 a,b,c,d 变量。

```
>>>list4=[1,2,3,4]
>>>a,b,c,d=list4
>>>print(a,b,c,d)
#输出结果为
1 2 3 4
```

注意:该方法要求接收的变量个数要与列表元素一致,否则系统会报错。

3.3.2 修改或删除列表元素

由于列表是有序的、可变的元素集合,因此,列表中的元素是可以修改与删除的。修改列表中元素的值可以用赋值号(=)给列表元素重新赋值:

```
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
>>>list1[3]="宏观经济学"
>>>list1
['大学语文', '高等数学', '英语', '宏观经济学', '中国文化', '艺术欣赏']
```

在 Python 中,可以使用 del 语句来删除列表或列表元素:

```
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
>>>del list1[2]
>>>list1
['大学语文', '高等数学', '政治经济学', '中国文化', '艺术欣赏']#索引为 2 的元素被删除
>>>del list1                                                    #删除了 list1 对象
>>>list1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'list1' is not defined
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
>>>del list1[0:3]
```

```
>>>list1
['政治经济学', '中国文化', '艺术欣赏']                    #删除了索引为0~2的元素,不包括3。
```

也可以用remove()方法来删除指定的元素。见如下示例：

```
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
>>>list1.remove("政治经济学")
>>>list1
['大学语文', '高等数学', '英语', '中国文化', '艺术欣赏']
>>>list3=[["大学语文",87],["高等数学",84],["英语", 90]]
>>>list3.remove(["英语",90])
>>>list3
[['大学语文', 87], ['高等数学', 84]]
```

3.3.3 列表脚本操作符与列表截取

列表脚本操作符有＋和＊两个。这两个操作符对列表的操作与对字符串操作相似。＋用于组合列表，＊用于重复列表，如表3.8所示。

表3.8 列表操作符

表 达 式	结 果	描 述
[1, 2, 3]+[4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合成新列表
['Hi! '] * 4	['Hi! ', 'Hi! ', 'Hi! ', 'Hi! ']	重复相应次数构成列表

列表截取与字符串截取相同，如下示例能帮助读者认识并掌握列表的截取方法。

```
>>>list1=["大学语文", "高等数学", "英语", "政治经济学", "中国文化", "艺术欣赏"]
```

通过表3.9的列表截取方法会得到相应的结果。

表3.9 列表截取实例

表 达 式	结 果	描 述
List1[2]	"英语"	读取列表中第3个元素
List[-2]	"中国文件"	读取列表中倒数第2个元素
List1[3：]	['政治经济学', '中国文化', '艺术欣赏']	从第4个元素开始截取列表
List1[2：4]	['英语', '政治经济学']	截取列从第3个元素到第4个元素

假如有students_info列表存放多个学生的姓名、年龄、爱好信息。由于学生爱好有多项，因此，可以用如下列表来存储学生信息。

```
students_info=[['egon',18,['play',]],['alex',18,['play','sleep']]]
```

如果要截取姓名为 egon 学生的第一个爱好，可以使用 students_info[0][2][0]；如果要截取第二学生的所有爱好，可使用 students_info[1][2]；如果要截取第二学生的第二个爱好，可使用 students_info[1][2][1]。见如下示例：

```
>>>students_info=[['egon',18,['play',]],['alex',18,['play','sleep']]]
>>>students_info[0][2][0]
'play'
>>>students_info[1][2][0]
'play'
>>>students_info[1][2]
['play', 'sleep']
>>>students_info[1][2][1]
'sleep'
>>>
```

3.3.4 列表函数与方法

Python 3 提供了如表 3.10 所示的列表函数和如表 3.11 所示的列表方法，用于对列表操作。

表 3.10 列表函数

序号	函 数 名	作 用
1	len(list)	列表元素个数
2	max(list)	返回列表元素最大值
3	min(list)	返回列表元素最小值
4	list(seq)	将元组转换为列表。元组内容见 3.4 节
5	enumerate(sequencd,[start=0])	函数用于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，同时列出数据和数据下标两部分，start 用于定义开始索引。该函数一般用在 for 循环当中

在上述函数中，enumerate 函数比较难理解，在此以示例来说明：

```
str1="hello!"
for v in enumerate(str1,start=1):
    print(v)
#以元组方式输出索引与数据,输出结果为
(1, 'h')
(2, 'e')
(3, 'l')
(4, 'l')
(5, 'o')
(6, '!')
```

```
str1="hello!"
for i,v in enumerate(str1,start=1):
    print(i,v)
#用 i 接收索引,用 v 接收数据,输出结果为
1 h
2 e
3 l
4 l
5 o
6 !
```

表 3.11 列表方法

序号	方 法	作 用
1	list.append(obj)	用于在列表末尾添加新的对象
2	list.count(obj)	统计某个元素在列表中出现的次数
3	list.extend(seq)	在列表末尾一次性追加另一个序列中的多个值(用新列表扩展原来的列表)
4	list.index(obj)	从列表中找出某个值第一个匹配项的索引位置
5	list.insert(index, obj)	将指定对象插入列表的指定位置
6	list.pop([index=-1])	指定元素的索引值来移除列表中的某个元素(默认是最后一个元素),并且返回该元素的值,如果列表为空或者索引值超出范围会报一个异常
7	list.reverse()	反转列表中的元素
8	list.sort(【reverse=False】)	对原列表进行排序。reverse 默认值是 False,可以给它赋值成 True,那就是反向排序

3.3.5 列表生成式

列表生成式(list comprehensions)是 Python 内置的非常简单、实用的用来创建列表的方法。假如要生成列表[0,1, 2, 3, 4, 5, 6, 7, 8, 9],可以用列表生成式来实现,方法如下:

```
>>>[x for x in range(0,10)]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

假如要生成列表[0,1, 4, 9, 16,25,36, 49,64,81],也可以用列表生成式来生成,方法如下:

```
>>>[x*x for x in range(0,10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

在 for 循环后面还可以加 if 判断语句来对列表进行筛选。下面的列表生成式可生成元素为偶数的列表：

```
>>>[x for x in range(0,10) if x%2==0]
[0, 2, 4, 6, 8]
```

可以使用双层循环生成组合列表，见如下示例：

```
>>>[m+n for m in "ABC" for n in "123" ]
['A1', 'A2', 'A3', 'B1', 'B2', 'B3', 'C1', 'C2', 'C3']
```

下面列表生成式能把列表中所有的字符串变成小写生成一个列表：

```
>>>L =['Hello', 'World', 'IBM', 'Apple']
>>>[s.lower() for s in L]
['hello', 'world', 'ibm', 'apple']
```

从上可知，运用列表生成式可以快速生成列表，也可以通过一个列表推导出另一个列表，而代码却十分简洁。

3.4 元　　组

Python 中元组(tuple)数据类型与列表类似，不同之处在于元组中的元素是不能修改的。

3.4.1 元组的创建与基本操作

元组创建很简单，只需要在()圆括号中添加元素，并使用逗号分隔即可。如果元组只有一个元素，通常在元素后加逗号。下列是创建元组的方法：

```
tup1 =('physics', 'chemistry', 1997, 2000)
tup2 =(1, 2, 3, 4, 5 )
tup3 ="a", "b", "c", "d"
tup1 =();                    #创建空元组
tup1 =(50,)                  #规范写法
```

在 Python 中，访问元组中的值可使用方括号进行指定索引切片或索引，以获取该索引处的值。具体使用方法与列表相同，在此不再重复。

1. 元组基本操作

元组响应＋和＊运算符很像列表，它们执行连接和重复操作，结果是一个新的元组。元组的基本操作如表 3.12 所示。

表 3.12 元组基本操作

Python 表达式	结 果
len((1, 2, 3))	3
(1, 2, 3)+(4, 5, 6)	(1, 2, 3, 4, 5, 6)
('Hi! ',) * 4	('Hi! ', 'Hi! ', 'Hi! ', 'Hi! ')
3 in (1, 2, 3)	True
for x in (1,2,3) : print (x, end='')	1 2 3

2. 内置元组函数功能

元组的内置函数与列表相同,仅有 tuple(seq)函数是将列表转换为元组。

3.4.2 元组与列表的应用

1. 用列表与元组实现生成一副扑克牌程序(不包括大小王)

实现思路:

为了存储 52 张牌(不包括大小王),可以先定义一个临时空列表,然后把 2～10 添加到该列表中,再用列表的 extend 方法把 J、Q、K、A 也添加到此列表中。利用循环获取扑克牌类型["黑桃", "红桃", "方块", "草花"]中的元素,并和临时列表中的元素进行结合,把结合产生的结果添加到一个新的空列表 card 中。card 中的每一张牌用一个元组表示。例如:[('红心',2),('草花',2), …,('黑桃 A')]。程序代码如下:

```
#author chenzhen
#date 2019/1/7
temp_list = []
card = []
for i in range(2, 11):
    temp_list.append(i)
temp_list.extend(["J", "Q", "K", "A"])
for i in temp_list:
    for card_type in ["黑桃", "红桃", "方块", "草花"]:
        a = (card_type, i)
        card.append(a)
print(card)
```

2. 用列表与元组实现购物车

实现思路:首先要向客户展示一个商品列表,列表包括商品编号、商品名称与单价。客户从商品列表中选择商品放到购物车,客户在选择商品时,系统必须知道客户账户的资金额,是否有购买选择商品的足额资金。为了实现这项功能,客户在选择商品前系统