第3章　Python容器数据类型

除整型、浮点型和布尔型等基本数据类型外，Python还提供列表、元组、字典和集合等容器数据类型，容器数据可以容纳批量数据。这些容器数据类型又可分为序列类型、映射类型和集合类型。序列类型包括字符串、列表和元组等，它们的元素都是有序排列的，可以通过索引访问它们的元素。字典属于映射类型。映射类型的元素由键（key）和值（value）组成，简称“键值对”。在一个映射型的变量中，键是唯一的。集合类型变量中的元素是没有顺序的，且不允许出现重复的元素，类似数学中的集合。

3.1　列表

列表（list）是Python中最常用的序列类型。它由一系列元素（又称数据项）组成，所有元素都被包含在一对方括号“[]”中。列表中的元素可以是任何类型的数据，并且不需要所有元素具有相同的类型。列表具有如下特性：

（1）列表中的元素类型可以相同，也可以不同。

（2）每个列表元素都有索引和值两个属性，索引用于标识元素在列表中的位置，值指的是元素本身的值。

序列类型的元素索引分为正索引和负索引（见图3.1）。正索引的索引值从0开始，在列表中从左向右依次递增1，因此最后一个元素的索引为元素个数减1。负索引的索引值从-1开始，在列表中从右向左依次递减1。一般情况下列表的索引是正索引。

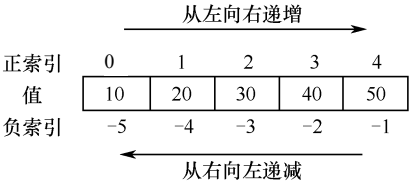

图 3.1　元素的正索引和负索引

3.1.1　创建列表和存取列表元素

可以使用方括号“[]”创建一个列表对象，也可以使用函数list()创建一个列表对象。创建列表的代码格式是

```
[元素1, 元素2,... ]
```

或

```
list(元素1, 元素2, ...)
```

【例】列表的创建。

```
In: lst = [ ]                    # 创建空列表
    lst
Out: []

In: list()                       # 创建空列表
Out: []

In: [12, 3.0, 'python']          # 列表元素的数据类型可以不相同
Out: [12, 3.0, 'python']
```

```
In: list('python')                # 字符串中的每个字符成为列表中的一个元素
Out: ['p', 'y', 't', 'h', 'o', 'n']
```

3.1.2 列表基本操作

1. 访问列表元素

可以使用索引访问列表中的元素，其格式如下：

```
列表对象[index]
```

index：元素索引。

【例】访问列表中的元素。

（1）获取lst列表中索引为1的元素

```
In: lst = [1, 2, 3]
    lst[1]
Out: 2
```

（2）修改lst列表中索引为1的元素的值

```
In: lst = [1, 2, 3]
    lst[1] = 5                    # 将索引为1的元素的值修改为5
    lst
Out: [1, 5, 3]
```

2. 列表合并

可以使用运算符“+”将两个列表合并在一起。

【例】合并两个列表。

```
In: [1, 2] + [3, 'abc']
Out: [1, 2, 3, 'abc']
```

3. 重复

可以使用运算符“*”创建具有重复元素的列表。

【例】创建元素“1, 2”重复3次的列表。

```
In: [1, 2] * 3
Out: [1, 2, 1, 2, 1, 2]
```

4. 迭代

迭代操作可用于遍历列表中的元素。

【例】迭代输出列表lst中的元素。

```
In: lst = [1, 2, 3, 4, 5]
    for x in lst:
        print(x)
1
2
3
4
5
```

5. 成员判断

使用“in”操作符判断对象是否属于列表。

【例】判断对象是否属于列表。

```
In: 2 in [1, 2, 3]
Out: True
In: 5 in [1, 2, 3]
Out: False
```

3.1.3　列表常用函数

Python提供了一系列处理列表对象的函数，可以对列表进行元素添加、删除、排序和反转等操作。表3.1列出了列表对象lst的常用函数。

表3.1　列表对象lst的常用函数

函　数　名	说　　明
lst.append(x)	在尾部添加新元素x
lst.index(x)	返回x在列表中的索引位置，如不含x将返回运行时错误
lst.count(x)	统计指定元素值x出现的次数，如不含x将返回0
del lst[index]	删除列表对象中索引为index的元素
lst.extend(lst2)	将lst2的元素添加到lst的尾部
lst.insert(index, x)	将x插入到列表中的index位置
len(lst)	求列表长度（注意，不是lst.len()语法）
lst.pop()	默认删除尾部元素，并返回删除对象
lst.remove(x)	删除元素x
lst.reverse()	将列表元素前后颠倒
lst.sort()	默认从小到大排序

【例】列表常用函数的应用。

（1）在lst列表末尾添加元素

```
In: lst = [1, 2, 3]
    lst.append(5)       # 末尾插入
    lst
Out: [1, 2, 3, 5]
In: lst.index(1)        # 数据1在列表的第0个位置
Out:0
In: lst.index(4)        # 报错，列表中不包含数据4
ValueError: 4 is not in list
```

（2）统计lst列表中元素值2出现的次数

```
In: lst = [1, 2, 2, 2, 3]
    lst.count(2)        # 2出现了3次
Out: 3
In: lst.count(4)        # 列表中不含数据4，返回0次
Out: 0
```

（3）删除lst列表索引为1的元素

```
In: lst = [1, 2, 3]
    del lst[1]
    lst
Out: [1, 3]
```

（4）将列表x中的元素添加到lst列表中

```
In: lst = [1, 2, 3]
    x = [4, 5]
    lst.extend(x)
    lst
Out: [1, 2, 3, 4, 5]
# 注：lst.extend()和lst.append(x)的效果不同，对比如下
In: lst = [1, 2, 3]
    lst.append(x)
    lst
Out: [1,2,3,[4,5]]  # x作为一个整体append添加到lst的末尾
```

（5）在lst列表索引为1的位置插入值为5的元素

```
In: lst = [1, 2, 3]
    lst.insert(1, 5)
    lst
Out: [1, 5, 2, 3]
```

（6）获取lst列表的长度（列表元素的个数）

```
In: lst = [1, 2, 3]
    len(lst)
Out: 3
```

（7）删除lst列表指定索引位置的元素，并返回该删除元素，若没有指定索引，则删除列表末尾的元素

```
In: lst = [1, 2, 3]
    lst.pop(1)
Out: 2
In: lst
Out: [1, 3]
```

（8）删除lst列表中值为2的元素，如果有多个，那么只删除第一个

```
In: lst = [1, 2, 2, 3]
    lst.remove(2)
    lst
Out: [1, 2, 3]
```

（9）将lst列表中的元素反转

```
In: lst = [1, 2, 5, 4]
    lst.reverse()
    lst
Out: [4, 5, 2, 1]
```

（10）对lst列表中的元素排序，若是数值型数据则按从小到大的顺序排序，若是字符串则按字典顺序排序。sort()函数的参数reverse=True时为逆序排序，默认为reverse=False升序排序

```
In: lst = [1, 4, 3, 5, 2]
    lst.sort()                          # 默认升序
    lst
Out: [1, 2, 3, 4, 5]
In: lst = [1, 4, 3, 5, 2]
    lst.sort(reverse=True)              # 参数reverse=True表示逆序排序
    lst
Out: [5, 4, 3, 2, 1]
```

注意不能写为lst=lst.sort()，因为lst.sort()执行后未返回新列表（返回值为None），若

将None值赋给lst，则lst原来的数据就会丢失。

lst.sort()方法直接改变原列表的顺序，如果排序时希望原列表不变，返回一个新的有序列表，那么可以借助sorted()函数。

```
In: lst2 = sorted(lst)                # 执行后lst不变，得到新的有序列表lst2
```

3.1.4　切片

列表的切片操作可以获得列表的多个元素。切片操作的基本格式如下：

```
列表对象[start : end : step]
```

start：索引开始位置，可以省略，默认为0。

end：索引结束位置（不包括end），可以省略，默认为列表长度。

step：步长，表示截取数据的步长，正数表示从左向右截取，负数表示从右向左截取，步长可以省略但不能为0，默认为1。

【例】列表切片应用。

```
In: lst = [1, 2, 3, 4, 5]
    lst[1:4]            # 截取列表lst中从索引1到索引3的元素
Out: [2, 3, 4]
In: lst[1:]             # 截取列表lst中从索引1到末尾的元素
Out: [2, 3, 4, 5]
In: lst[:4]             # 截取列表lst中从0开始到索引3的元素
Out: [1, 2, 3, 4]
In: lst[1:4:2]          # 截取列表lst中从索引1到索引3、步长为2的元素
Out: [2, 4]
In: lst[-4:-1]          # 截取列表lst中从索引-4到索引-2的元素
Out: [2, 3, 4]
In: lst[3:0:-1]         # 步长为负，从右向左截取列表lst中从索引3到索引1的元素
Out: [4, 3, 2]
In: lst[-2:-5:-1]       # 步长为负，从右向左截取列表lst中从索引-2到索引-4的元素
Out: [4, 3, 2]

In: lst = [1, 2, 3, 4, 5]
    lst[1:4] = ['a','b','c']          # 通过切片改变元素的值
    lst
Out: [1, 'a', 'b', 'c', 5]

In: lst = [1, 2, 3, 4, 5]
    lst[1:4] = []                     # 通过切片删除元素
    lst
Out: [1, 5]

In: lst = [1, 2, 3, 4, 5]
    lst[:] = []                       # 通过切片删除所有元素
    lst
Out: []
```

3.1.5　列表生成方式

1. 产生一个数值递增列表

使用range()函数可以产生一个数值递增且可迭代操作的对象，基本格式如下：

```
range(start, end, step)
```

start：起始元素值，可以省略，默认为0。

end：结束元素值（不包括end），不能省略。

step：步长，必须为整数，正数表示数值递增，负数表示数值递减，可以省略，默认为1。

注意range(100)并未在内存中立即产生100个数，只是产生了一个可迭代对象，具体数据将在后续迭代过程中逐一产生，这样有利于节省内存。将range()函数产生的迭代对象作为列表创建函数list()的参数，可以产生一个数值递增的列表。

【例】产生一个数值递增列表。

```
In: it = range(1, 5)          # 产生一个数值从1到4的迭代对象it
    lst = list(it)            # 以迭代对象为参数创建列表
    lst
Out: [1, 2, 3, 4]
In: list(range(2, 10, 2))     # 步长为2
Out: [2, 4, 6, 8]
```

2．产生多维列表

多维列表可视为列表中嵌套的列表。例如，二维列表可视为每个元素都是一维列表的列表，三维列表可视为每个元素都是二维列表的列表。

【例】创建一个二维列表。

```
In: lst = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

此时，二维列表lst的内容如下所示，每行是一个一维列表：

1	2	3	4
5	6	7	8
9	10	11	12

多维列表的维数是从0开始的，例如，二维列表有第0维和第1维，分别对应于行和列。访问多维列表中的元素时，使用的命令格式如下：

```
列表对象[row][col]…
```

其中第一个“[]”为列表的第0维，第二个“[]”为第1维，以此类推。

【例】多维列表的访问。

```
In: lst = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
    lst[1][2]                 # 获取第0维索引为1、第1维索引为2的数据
Out: 7
In: lst[1]                    # 获取第0维索引为1的数据，是一个一维列表
Out: [5, 6, 7, 8]
```

3．列表生成式

Python的列表还可以用其特有的列表生成式语法产生。列表生成式的一般格式如下：

```
新列表对象=[ 表达式  for  变量  in  可迭代对象 <if 条件> ]
```

它表示从可迭代对象中取得变量值，将变量值代入表达式计算得到新值，由这些新值构成新的列表。<if 条件>是可选的，如果设有筛选条件，那么满足条件的变量值才能加入新列表。

```
In: lst = list(range(5))
    lst
Out: [0, 1, 2, 3, 4]
# 列表生成式，将lst列表中的每个元素平方，得到列表lst2
In: lst2 = [x**2 for x in lst]
```

```
In: lst2
Out: [0, 1, 4, 9, 16]
# 抽取在lst2中但不在lst中的元素得到列表lst3
In: lst3 = [ x for x in lst2 if x not in lst ]
In: lst3
Out: [9, 16]
In: [ x for x in range(20) if x%3==0 ]  # [0, 20]内3的倍数
Out:[0, 3, 6, 9, 12, 15, 18]
```

3.2　元组

元组（tuple）是用一对圆括号“()”定义的序列。元组中的元素值自元组创建后就不能修改。元组中的元素数据类型可以相同，也可以不同。

3.2.1　创建元组和存取元组元素

1．创建元组

可以使用圆括号“()”定义创建一个元组对象，也可以使用函数 tuple()创建一个元组对象。元组创建的格式如下：

```
(元素1, 元素2, ...)          # ()可省略
```

或

```
tuple(可迭代对象)            # 可迭代对象：字符串、列表、元组、字典、集合等
```

【例】创建三个元素的元组。

```
In: tpl = (1, 2, 3)              # 使用“()”创建元组，省略“()”亦可
    tpl
Out: (1, 2, 3)
In: tpl = (1,)                   #  创建只含一个元素的元组，逗号不能漏了
In: tpl2 = tuple([1, 2, 3])      # 使用tuple()函数创建元组，参数为列表对象
    tpl2
Out: (1, 2, 3)

In: it = range(1, 5)             # 产生可迭代对象
    tpl = tuple(it)              # 使用tuple()函数创建元组，参数为可迭代对象
    tpl
Out: (1, 2, 3, 4)
```

2．索引和切片

元组的索引和切片操作与列表的索引和切片操作非常相似。

【例】元组的索引操作和切片操作。

（1）访问元组tpl中索引为1的元素

```
In: tpl = (1, 2, 3)
    tpl[1]
Out: 2
```

（2）截取元组tpl中从索引位置1到索引位置3的元素

```
In: tpl = (1, 2, 3, 4, 5)
```

```
    tpl[1:4]
Out: (2, 3, 4)
```

3．其他操作

在Python中，元组的其他操作，如求元组的长度、合并、重复、迭代、成员判断等，也与列表的对应操作类似。

【例】元组的操作。

（1）元组求长度

```
In: tpl = (1, 2, 3)
    len(tpl)
Out: 3
```

（2）合并两个元组

```
In: (1, 2, 3) + (4, 5)
Out: (1, 2, 3, 4, 5)
```

需要注意的是，由于元组中的元素是不能增删的，因此合并元组不是在已有的元组中添加元素，而是产生一个新元组，新元组与已有元组在内存中是独立的。

（3）重复，创建元素“1，2”重复3次的元组

```
In: (1, 2) * 3
Out: (1, 2, 1, 2, 1, 2)
```

（4）迭代元组中的所有元素

```
In: tpl = (1, 2, 3)
    for x in tpl:
        print(x)
1
2
3
```

（5）判断对象是否为元组中的元素

```
In: 2 in (1, 2, 3)
Out: True
In: 5 in (1, 2, 3)
Out: False
```

（6）统计tpl元组中元素值2出现的次数

```
In: tpl = (1, 2, 2, 2, 3)
    tpl.count(2)
Out: 3
```

3.2.2 元组和列表的差异

元组一旦创建，其中的元素就不可修改。元组也不能增加或删除元素。因为元组结构简单，所以元组占用的内存比列表占用的内存少，如果待处理的数据多用于查询，无须修改，那么可以考虑用元组存储。另外，由于元组是不可修改的，在函数调用或返回函数值时可考虑传递元组，以避免参数被误修改。

```
In: tpl = (1, 2, 3)
    tpl[0] = 100            # 报错，元组中的元素不可修改
```