第3章

组合数据类型

CHAPTER 3

本章学习目标

- 了解 Python 语言的数据类型。
- 掌握各种组合数据类型的特点和应用场景，并能够合理使用。
- 掌握各种组合数据类型的创建、访问、操作方法。

Python 包含基本数据类型和组合数据类型，数字类型是基本数据类型，包括整数类型、浮点数类型、布尔类型和复数类型，这些类型仅能表示一个数据，这种表示单一数据的类型称为基本数据类型。然而，实际计算中却存在大量同时处理多个数据的情况，这就需要将多个数据有效组织起来并统一表示，这种能够表示多个数据的类型称为组合数据类型。

3.1 组合数据类型概述

利用计算机系统进行数据处理的一个优势是可以处理大量数据。为了更有效地完成数据处理，Python 会将相关的数据组织到一个结构中，称为组合数据。例如，一个班级的所有同学的姓名，图书馆书架上所有图书的书名，网上商店的所有商品名等。再如，一位同学的姓名、性别、出生日期、专业，一本图书的编号、书名、价格，商品的编号、商品名、价格、生产商等。

根据数据之间的关系，Python 的组合数据类型可以分为三类：序列类型、映射类型和集合类型。

Python 中典型的序列类型包括字符串(str)、列表(list)、元组(tuple)。字符串是单一字符的有序集合，所以也可以视作基本数据类型。无论哪种序列类型，都可以使用相同的索引体系，从左到右序列号递增(下标值从 0 开始)，从右到左序列号递减(下标值从 −1 开始)。序列可以进行的操作包括索引、切片、加、乘、检查成员。

集合类型与数学中集合的概念一致，Python 提供了一种同名的具体数据类型集合(set)，集合中的元素是无序的，集合中不允许有相同的元素存在；映射类型用键值对表示数据，典型的映射类型是字典(dict)。

Python 的组合数据类型如图 3-1 所示。

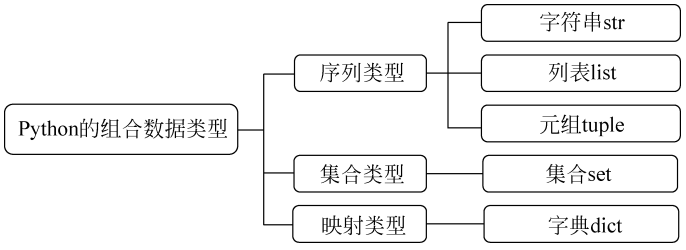

图 3-1 Python 的组合数据类型

3.2 列表

Python 支持多种复合数据类型，用于将多个值组合在一起。最通用的是列表，将一些值用逗号隔开放在方括号内就形成了列表。列表中的元素之间存在先后关系，所以列表中可以存在数值相同但位置(下标)不同的元素。一般的数据类型元素都可以保存在列表中，同一列表中元素的类型可以不同。

在列表中，元素与下标存在对应关系，可以方便地使用循环结构处理列表中的值，并用列表实现数组、栈等的功能。

3.2.1 列表的创建

列表的创建比较灵活，一般可以采用以下三种方法来创建。

(1) 赋值并创建列表。例如：

```
>>> squares = [1, 4, 9, 16, 25]
```

（2）通过 list()函数创建列表。例如：

```
>>> a = list()                    # 创建空列表
>>> a
[]
>>> c = list("abcd")              # 创建列表同时赋初值
>>> c
['a', 'b', 'c', 'd']              # 结果是多个字符元素
```

（3）可以利用 range()函数创建序列元素的列表。例如：

```
>>> c = [x for x in range(1,10,2)]
>>> c
[1, 3, 5, 7, 9]
```

3.2.2　列表的索引和切片

列表属于序列的一种，与字符串一样，列表可以进行索引和切片。列表的索引就是通过下标读取列表的元素。列表元素的下标有两种表示方法。

（1）从前到后的顺序，第一个元素的下标为 0，其后的元素下标依次加 1。

（2）从后向前的顺序，最后一个元素的下标为－1，之前的元素下标依次减 1。

例如：

```
>>> squares = [1, 4, 9, 16, 25]
 [1, 4, 9, 16, 25]
>>> squares[0]
1
>>> squares[ - 1]
25
```

列表的切片就是通过指定下标的起始和结束数值读取列表中连续的部分元素，其结果是一个新的列表，包含指定的起始下标元素，不包含指定的结束下标元素。使用列表的切片时，下标可以采用上述两种表示方法中的一种，例如：

```
>>> squares[1:3]                  # 结果不包含 squares[3]
[4, 9]
>>> squares[ - 3:]                # 结束数值为空，表示到列表末尾
[9, 16, 25]
>>> squares[:]                    # 起始和结束数值均为空，表示整个列表
[1, 4, 9, 16, 25]
```

3.2.3　列表的修改

1. 修改列表中元素的值

和字符串不同的是，列表是可变的，可以在列表中指定下标的值对元素进行修改。

例如：

```
>>> cubes = [1, 8, 27, 65, 125]
>>> cubes[3] = 64
>>> cubes
 [1, 8, 27, 64, 125]
```

2. 修改列表切片的值

Python 可以通过为切片的元素重新赋值实现一次修改多个元素的值。为切片赋空列表可以实现删除部分元素的效果；如果为整个列表重新赋空列表，相当于删除所有元素。例如：

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> letters[:] = []
>>> letters
[]
```

3.2.4 列表的连接和嵌套

1. 列表的连接

可以使用“＋”将两个列表合成为一个列表，“＋”号后的列表元素将排列在前一列表之后，例如：

```
>>> firstList = [1, 2, 3]
>>> secondList = [ 3,4, 5, 6]
>>> combinedList = firstList + secondList            # 无论元素是否重复
 [ 1, 2, 3, 3, 4, 5, 6]
```

2. 列表的嵌套

当列表的元素为列表时，称为列表的嵌套。在此情况下，要读取内部列表的元素需要在列表名后使用两个方括号，第一个方括号内的值为目标元素所在子列表，第二个方括号内的值为子列表中的目标元素，例如：

```
>>> stu1 = ['张三','M',20]
>>> stu2 = ['李四','F',19]
>>> stu3 = ['王五','M',21]
>>> stu = [stu1,stu2,stu3]
>>> stu
[['张三', 'M', 20], ['李四', 'F', 19], ['王五', 'M', 21]]
```

```
>>> stu[1]
['李四', 'F', 19]
>>> stu[1][2]
19
```

3.2.5　列表的方法

列表还有一些特有的方法，它们的主要功能是完成列表元素的增删改查等操作。处理列表的常用方法有 12 个，如表 3-1 所示。

表 3-1　处理列表的常用方法

方　　法	说　　明
len(listName)	列表包含元素的个数
listName.append(X)	在列表末尾添加元素，X 为添加到列表末尾的元素，可以是值或表达式
listName.insert(i,X)	在指定的位置插入元素。第一个参数为插入元素的索引。如果该参数为 0，则在列表最前面插入；如果该参数为列表的长度，则在最后面插入元素，相当于 append()方法
listName.remove(X)	删除列表中第一个值为 X 的元素。如果列表中没有值为 X 的元素，则会产生 ValueError 异常
dellistName[i]	删除指定索引的元素，可以使用切片指定元素范围
listName.pop(i)	删除并返回列表中索引为 i 的元素，如果该方法未设置参数，则删除并返回最后一个元素
listName.clear()	删除列表的所有元素
listName.index(x[,start[,end]])	在指定切片范围中搜索值为 X 的第一个元素，并返回其索引值。start,end 的值可选
listName.count(X)	返回列表中出现 X 的次数
listName.sort()	对列表进行排序。设置 reverse＝True 的参数可以实现倒置
listName.reverse()	对列表元素的顺序进行倒置
listName.copy()	复制列表

注意：listName 表示列表名。例如：

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')                    # 返回符合条件的第一个下标
3
>>> fruits.index('banana', 4)                 # 从索引 4 之后查找
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
```

```
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()
'pear'
```

3.3 元组

列表和字符串有很多共同的属性，都可以进行索引和切片操作。因为它们都属于序列，另外一种序列就是元组。

元组的特性和列表相似，两者最大的不同是元组中的元素是不可修改的，所以元组一般用于数据准备。例如，一副扑克牌的花色（只有 4 种而且不变）可以保存在元组当中。

3.3.1 元组的创建和访问

元组中的元素使用逗号分隔，所有元素可以放在圆括号内，圆括号也可以省略。与列表不同，元组中的元素值是不可以修改的。

1. 元组的创建

（1）通过赋值形式创建。例如：

```
>>> x = (1,3,4,5)
>>> x
(1, 3, 4, 5)
>>> t = 12345, 54321, 'hello!'          # 创建元组时，外层的圆括号可以省略
>>> t
(12345, 54321, 'hello!')
```

（2）创建只包含单个元素的元组。当创建的元组只有一个元素时，该元素后面的“,”不能省略，否则，Python 将认为是在创建一个其他类型的变量。例如：

```
>>> t1 = 3,                             # 只有一个元素时，末尾的逗号不能省略
>>> t1
(3,)
>>> t2 = 3                              # t2 为整型变量
>>> t2
3
>>> type(t1)                            # 检测 t1 的类型，为元组类型
<class 'tuple'>
type(t2)                                # 检测 t2 的类型，为整型
<class 'int'>
```

（3）使用 tuple()函数和 range()函数创建序列元素的元组。例如：

```
>>> t = tuple(range(1,10,2))
>>> t
(1, 3, 5, 7, 9)
```

2. 访问元组中的元素

访问元组中的元素方法与列表相同，可以通过下标或切片访问单个元素或多个元素。例如：

```
>>> x = (1,3,5,7,9)
>>> x[0]                              # 使用索引访问元组中的元素
1
>>> x[ - 1]
9
>>> x[0:3]                            # 使用切片访问元组中的元素
(1, 3, 5)
```

3.3.2　元组的更新和删除

1. 元组的更新

元组是不可变序列，元组中的元素不能被修改，试图修改元组中的元素会出现错误。例如：

```
>>> x = (1,3,5,7,9)
>>> x[0] = - 5
TypeError: 'tuple' object does not support item assignment
```

（1）可以通过创建一个新的元组去替代旧的元组。例如：

```
>>> x = (1,3,5,7,9)
>>> x
(1, 3, 5, 7, 9)
>>> x = ('a','b','c')                 # 对元组 x 进行重新赋值
>>> x
('a', 'b', 'c')
```

（2）使用运算符“＋”连接多个元组，生成一个新的元组，实现向元组中添加新元素。例如：

```
>>> x = (1,3,5,7,9)
>>> y = ('a','b','c')
>>> z = x + y                         # 通过" + "连接多个元组,生成一个新的元组
>>> z
(1, 3, 5, 7, 9, 'a', 'b', 'c')
```

（3）间接更新。元组的元素既可以是基本数据类型，也可以是列表或元组等组合数据类型。元组中元素的值不可以被修改，但如果元组的某个元素是可以修改的数据类型，则可通过修改该元素中的值实现元组的间接更新。例如：

```
>>> t = ([1,2,3],'x','y')             # 元组 t 中的第 1 个元素是列表类型
>>> t[0]
[1, 2, 3]
```

```
>>> t[0][1] = 4                    # 通过下标修改列表类型中的元素
>>> t
([1, 4, 3], 'x', 'y')
```

2. 删除的元组

当创建的元组不再使用时，可以使用 del 关键字将其删除。Python 自带垃圾回收功能，会自动销毁不用的元组，所以一般不需要使用 del 来手动删除。例如：

```
>>> x = (1,3,5,7,9)
>>> del x
>>> x
Traceback (most recent call last):
  File "< pyshell # 23 >", line 1, in < module >
    x
NameError: name 'x' is not defined
```

此时，如果再调用 x，编译器将会抛出异常信息，提示该变量未定义。

3.3.3 元组同时为多个变量赋值

将多个数据赋值给元组可以看成是将数据组合在一起的过程，元组可以同时将所有元素一次赋值给多个变量，这时就要求赋值符号左边的变量数与元组的元素个数相同，否则会产生异常。例如：

```
>>> t = (1,2,3)
>>> x,y,z = t
>>> x
1
>>> y
2
>>> z
3
```

3.4 集合

Python 语言中的集合是一个包含 0 个或多个数据项且无序的、不重复的数据组合，其中，元素类型只能是固定数据类型，如整数、浮点数、字符串、元组等。相反，列表、字典和集合本身都是可变数据类型，因此它们不能作为集合元素来使用。

3.4.1 集合的创建

Python 提供了两种创建集合的方法，分别是使用{}创建和使用 set()函数将列表、元组等类型数据转换为集合。

(1) 使用{}创建集合，通过定义集合变量同时赋初值的方式创建集合。例如：

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> basket
{'orange', 'banana', 'pear', 'apple'}
```

(2) 使用 set()函数创建集合。集合元素是独一无二的,使用集合类型可以过滤掉重复元素。set(x)函数用于生成集合,输入任意组合数据类型的参数,返回一个无序不重复的集合。例如:

```
>>> a = set('abracadabra')
>>> a
{'a', 'r', 'b', 'c', 'd'}
>>> square = [1,4,9,16,25]
>>> squareSet = set(square)
>>> squareSet
{1, 4, 9, 16, 25}
```

3.4.2　集合的运算

Python 中的集合与数学中集合的概念是一致的,因此,两个集合可以做数学意义上的交集、并集、差集计算等。

例如有两个集合,分别为 a={10,20,30}和 b={20,30,40},它们既有相同的元素,也有不同的元素。以这两个集合为例,分别做不同的集合运算的结果如表 3-2 所示。

表 3-2　集合的运算

运　算　符	名　　称	含　　义	实　　例
&	交集	取两个集合的公共元素	>>> a&b {20, 30}
|	并集	取两个集合的全部元素	>>> a|b {20, 40, 10, 30}
−	差集	取一个集合中另一个集合没有的元素	>>> a−b {10}
^	对称差集	合并后去掉共同包含的元素	>>> a^b {40, 10}

3.4.3　集合的操作

集合是无序组合,没有索引和位置的概念,不能切片,集合中的元素可以动态增加或删除。集合的常用方法如表 3-3 所示。

表 3-3　集合的常用方法

方　　法	等 价 符 号	说　　明
a.issubset(b)	a<=b	子集测试(允许不严格意义上的子集)
	a<b	子集测试(严格意义上的子集)
a.issuperset(b)	a>=b	超集测试(允许不严格意义上的超集)
	a>b	超集测试(严格意义上的超集)

续表

方　法	等价符号	说　明
a.union(b)	a|b	合并操作：a 或 b 中的元素
a.intersection(b)	a&b	交集操作：a 与 b 中的元素
a.copy()		返回 a 的复制
a.add(obj)		加操作：将 obj 添加到 a
a.remove(obj)		删除操作：将 obj 从 a 中删除，如果 a 中不存在 obj，将引发异常
a.pop()		弹出操作：移除并返回 a 中的任意一个元素
a.clear()		清除操作：清除 a 中的所有元素

集合的成员运算如表 3-4 所示。

表 3-4　集合的成员运算

成员运算	说　明
obj in a	判断集合 a 中是否包含 obj 元素，结果为 True 或 False
obj not in a	判断集合 a 中是否包含 obj 元素，结果为 True 或 False

例如：

```
>>> a = {10,20,30}
>>> a.add('OK')                    # 加操作:将'OK'添加到 a 中
>>> print("a = ",a)
a = {10, 20, 30, 'OK'}
>>> b = {20,30,40}
>>> b.remove(30)                   # 删除操作:将 30 从集合 b 中删除
>>> print("b = ",b)
b = {40, 20}
>>> 20 in b                        # 判断集合 b 中是否包含 20
True
>>> b.clear()                      # 清除操作:清除集合 b 中的所有元素
>>> print("b = ",b)
b = set()
>>> 20 in b
False
```

3.5　字典

列表可以看作有序的容器，对放进去的每一个数据在列表中有一个编号，通过这个编号可以访问数据，而这个编号是一个整数。生活中有一些类似但不相同的场景，如通讯录。通讯录中的数据以名字来存取。通过一个名字，可以访问其中的电话号码等数据。

Python 提供了一种数据类型，可以用名字做索引来访问其中的数据，这种类型就是字典。换言之，字典是一个用“键”做索引来存储数据的集合，就像列表是用整数做索引一样，“键”在这里起到的作用就是索引。列表的一个索引对应着列表中的一个数据，字典中的一个“键”对应着字典中的一个数据，即可以通过“键”获得“值”。一个键和它所对应的数据形成字典中的一个条目。数值或者元素不可变的组合数据类型都可以作为“键”，并且在一个字典中，“键”是不可以重复的。

3.5.1　字典的创建

字典用于存储成对的元素，在一个字典对象中，键值不能重复，用于唯一标识一个键值对，而对于值的存储则没有任何限制。字典可以看成是由键值对构成的列表，在搜索字典时，首先查找键，当查找到键后就可以直接获取该键对应的值，这是一种高效实用的查找办法。

字典可以用标记“{}”创建，字典中每个元素包含键和值两部分，键和值用冒号分开，元素之间用逗号分隔。dict()函数用于创建字典，函数参数为“键＝值”或“(键,值)”，参数中间用逗号隔开。

【例 3-1】 创建由“名字：电话号”构成的键值对字典。

```
tel1 = {}              # 创建一个空的字典,该字典不包含任何元素
tel2 = {'Joe': 4139, 'Peter': 4127,'Jack': 4098}
tel3 = dict(Joe =  4139,Peter =  4127,Jack = 4098)
tel4 = dict([('Joe', 4139), ('Peter', 4127), ('Jack', 4098)])
print('tel1 = ',tel1)
print('tel2 = ',tel2)
print('tel3 = ',tel3)
print('tel4 = ',tel4)
```

运行结果：

```
 ====================================
tel1 =  {}
tel2 =  {'Joe': 4139, 'Peter': 4127, 'Jack': 4098}
tel3 =  {'Joe': 4139, 'Peter': 4127, 'Jack': 4098}
tel4 =  {'Joe': 4139, 'Peter': 4127, 'Jack': 4098}
```

在例 3-1 中，第 1 行用于创建一个空的字典，该字典不包含任何元素，可以向字典中添加元素。第 2 行是典型的创建字典的方法，是用“{}”括起来的键值对。第 3 行使用 dict()函数，通过关键字参数创建字典。第 4 行使用 dict()函数，通过键值对序列创建字典。

3.5.2　字典的操作

字典包含多个键值对，而键是字典的关键数据，因此对字典的操作都是基于键的。基本操作如下。

(1) 字典的操作主要是向字典中增加新值和通过“键”读取“值”。例如：

```
>>> tel = dict([('Joe', 4139), ('Peter', 4127), ('Jack', 4098)])
>>> print(tel['Jack'])              # 读取字典元素
4098
>>> tel['Jack'] = 4088              # 修改字典元素"键"所对应的值
>>> print(tel['Jack'])
4088
>>> tel['Mike'] = 3298              # 如果访问字典的"键"不存在,则自动添加到该字典中
>>> tel
{'Joe': 4139, 'Peter': 4127, 'Jack': 4088, 'Mike': 3298}
```

(2) 可以使用 del 关键字删除字典中的元素。例如：

```
>>> del tel['Jack']
>>> tel
{'Joe': 4139, 'Peter': 4127, 'Mike': 3298}
```

(3) 使用 list()函数可以将字典的“键”生成一个列表。例如：

```
>>> list(tel)
['Joe', 'Peter', 'Mike']
```

(4) 使用 sorted()函数可以将字典的“键”生成一个列表，同时进行排序。例如：

```
>>> sorted(tel)
['Joe', 'Mike', 'Peter']
```

(5) 用 in 和 not in 运算符可以检测一个键是否存在于字典中，返回 True 或 False。例如：

```
>>> print('Jack' in tel)
True
>>> print('Joe' not in tel)
False
```

本章习题

一、填空题

1. 列表、元组、字符串是 Python 的_____(有序/无序)序列。

2. 表达式[1,2,3] * 3 的执行结果为_____。表达式(1,2,3) * 3 的执行结果为_____。表达式'123' * 3 的执行结果为_____。

3. 表达式 [2]in[1,2,3,4]的值为_____。

4. 任意长度的 Python 列表、元组和字符串中最后一个元素的索引为_____。

5. 语句 list(range(1,10,3))的执行结果为_____。

二、判断题

1. Python 字符串提供切片访问方式，采用[N:M]格式，表示字符串中从 N 到 M 的索引子字符串(包含 N 和 M)。(　　)

2. del 命令既可以删除列表中的一个元素，也可以删除列表的所有元素。(　　)

3. 在 Python 语言中，字符类型是字符串类型的子类型。(　　)

4. s[0:-1]与 s[:]表示的含义相同。(　　)

5. 列表中的所有元素数据类型必须相同。(　　)

三、选择题

1. 访问字符串的部分字符的操作为(　　)。

A. 分片　　B. 合并　　C. 索引　　D. 赋值

2. Python 语句 print(type([1,2,3,4]))的输出结果是(　　)。

A. <class'tuple'>　　B. <class'dict'>　　C. <class'set'>　　D. <class'list'>

3. 若 alist=[3,2],则执行 alistinsert(-1,9)后,alist 的值是(　　)。

A. [3,2,9]　　B. [3,9,2]　　C. [9,3,2]　　D. [9,2,3]

4. 与代码[1, 2, 3,'1','2','3'][-2]执行结果一致的是(　　)。

A. [1, 2,3][-2]　　B. ['1', 2,'3'][-2]

C. (0, 1, 2, 3,'1', '2', '3', '4')[4]　　D. (3,'1','2')[-1]

5. 以下程序的输出结果是(　　)。

```
L2 = [1,2,3,4]
L3 = L2.reverse()
print( L3)
```

A. [4, 3, 2, 1]　　B. [3, 2, 1]　　C. [1,2,3,]　　D. None

6. 以下不是 tuple 类型的是(　　)。

A. (1)　　B. (1,)

C. ([], [1])　　D. ([{'a': 1}], ['b', 1])

7. 以下代码的执行结果是(　　)。

```
a = {'name': 'Jack', 'age': 28, 'job': 'teacher'}
print(sorted(a))
```

A. ['name', 'job', 'age']

B. {'name': 'Jack', 'job': 'teacher', 'age': 28}

C. ['age', 'job', 'name']

D. {'age': 28, 'job': 'teacher','name': 'Jack'}

8. 以下说法错误的是(　　)。

A. 元组的长度可变　　B. 列表的长度可变

C. 可以通过索引访问元组　　D. 可以通过索引访问列表

9. 以下对字典的说法错误的是(　　)。

A. 字典可以为空　　B. 字典的键不能相同

C. 字典的键不可变　　D. 字典的键的值不可变

10. 在 Python 中,不同的数据,需要定义不同的数据类型,可用方括号“[]”来定义的是(　　)。

A. 列表　　B. 元组　　C. 集合　　D. 字典

四、程序设计题

1. 为保护环境,很多城市开始对垃圾实行分类,为了让大家了解垃圾的分类情况,建立

了以下 4 类列表，list1(可回收垃圾)、list2(有害垃圾)、list3(易腐垃圾)，剩下的为其他垃圾。目前，列表中已经存储了以下数据：

list1=["玻璃瓶","旧书","金属","纸板箱","旧衣服","易拉罐"]

list2=["胶片","消毒水","纽扣电池","水银温度计","过期药水","泡沫塑料"]

list3=["动物内脏","菜叶菜梗","过期食品","香蕉皮","果壳"]

根据现有列表，实现以下功能：

(1) 从列表 list3 中取出“过期食品”。

(2) 从 list1 中截取["旧书","金属","纸板箱"]。

(3) 现又发现一个新的列表如下：list4=["过期化妆品","过期药品","杀虫剂"]，经判断，里面存放的为有害垃圾，将该列表中的元素添加到 list2 中。

(4) 小李在路上捡到了一个塑料瓶，判断为可回收垃圾，请将塑料瓶添加到列表 list1 中。

2. 有两个集合 set1={'A','e','3','f','7','','S'}，set2={'d','3','W','x','f','','9'}，求两个集合的交集、并集。