集合类型

- 集合是无序组合，用大括号 {} 表示，可包含0个或多个数据项。
- 定义空集合必须使用 set() 函数
- 集合中元素不可重复，元素类型只能是固定数据类型。
- 它没有索引和位置的概念，集合中元素可以动态增加或删除。
- 集合类型有4个操作符，交集（&）、并集（|）、差集 （-）、补集（^），操作逻辑与数学定义相同。

集合类型有一些常用的操作函数或方法

函数或方法	描述
s.add(x)	如果数据项x不在集合s中，将x增加到s
s.remove(x)	如果x在集合s中，移除该元素。不在产生 KeyError异常
s.clear()	移除s中所有数据项
len(s)	返回集合s元素个数
x in s	如果x是s的元素，返回True，否则返回False
x not in s	如果x不是s的元素，返回True，否则返回False

列表类型

- 所有元素放在中括号 [ ] 内
- 列表属于序列类型，元素类型可以不同
- 可以使用 list() 或 [ ] 创建空列表
- 支持双向索引
- 列表是可变的

列表的索引

1、索引是列表的基本操作，用于获得列表的一个元素。使用中括号作为索引操作符。

```
>>> ls = [1010,"1010",[1010,"1010"],1010]
>>> ls[3]
1010
>>> ls[-2]
[1010, '1010']
>>> ls[5]
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    ls[5]
IndexError: list index out of range
```

列表的切片

1、切片是列表的基本操作，用于获得列表的一个片段，即获得一个或多个元素。切片后的结果也是列表类型。切片有两种使用方式：

<列表或列表变量>[N：M]　　或　　<列表或列表变量>[N：M：K]

说明：切片获取列表类型从N到M（不包含M）的元素组成新的列表。

当K存在时，切片获取列表类型从N到M（不包 含M）以K为步长所对应元素组成的列表。

列表支持的操作符

```
>>> [1,3,5,7] + [2,4]
[1, 3, 5, 7, 2, 4]
>>> [1,3,5] * 3
[1, 3, 5, 1, 3, 5, 1, 3, 5]
```

元组类型

- 所有元素放在圆括号（）内
- 如果元组中只有一个元素，必须在最后增加一个逗号
- 可以使用 tuple() 或（）创建空元组
- 支持双向索引
- 元组是不可变的

元组的索引

1、索引是列表的基本操作，用于获得列表的一个元素。使用中括号作为索引操作符。

```
>>> t = (1,3,5,7)
>>> print(t[1])
3
>>> print(t[-1])
7
>>> t[4]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    t[4]
IndexError: tuple index out of range
```

元组的切片

1、元组可以支持切片操作，用于获得元组的一个片段，即获得一个或多个元素。切片后的结果也是元组类型。切片有两种使用方式：

<元组或元组变量>[N：M]　　或　　<元组或元组变量>[N：M：K]

说明：切片获取元组类型从N到M（不包含M）的元素组成新的元组。

当K存在时，切片获取元组类型从N到M（不包 含M）以K为步长所对应元素组成的元组。

元组支持的操作符

```
>>> (1, 3, 5) + (2, 4, 6)
(1, 3, 5, 2, 4, 6)
>>> (1, 3, 5) * 3
(1, 3, 5, 1, 3, 5, 1, 3, 5)
```

字典类型

- 元素放在花括号 { } 内
- 每个元素是一个键值对， 元素之间用逗号隔开
- 使用方式 { key1 : value1, key2 : value2 }
- 可以使用 dict() 或 { } 创建空字典
- 键值对之间没有顺序且不能重复

查找字典中元素

字典中键值对的索引模式如下，采用中括号格式：

<值> = <字典变量>[<键>]

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> print(d["201802"])
小红
```

对字典中的元素修改

利用索引和赋值（=）配合，可以对字典中每个元素进行修改。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d['201802'] = '新小红'
>>> print(d)
{'201803': '小白', '201802': '新小红', '201801': '小明'}
```

向字典中添加元素

使用大括号可以创建字典。通过索引和赋值配合，可以向字典中增加元素。

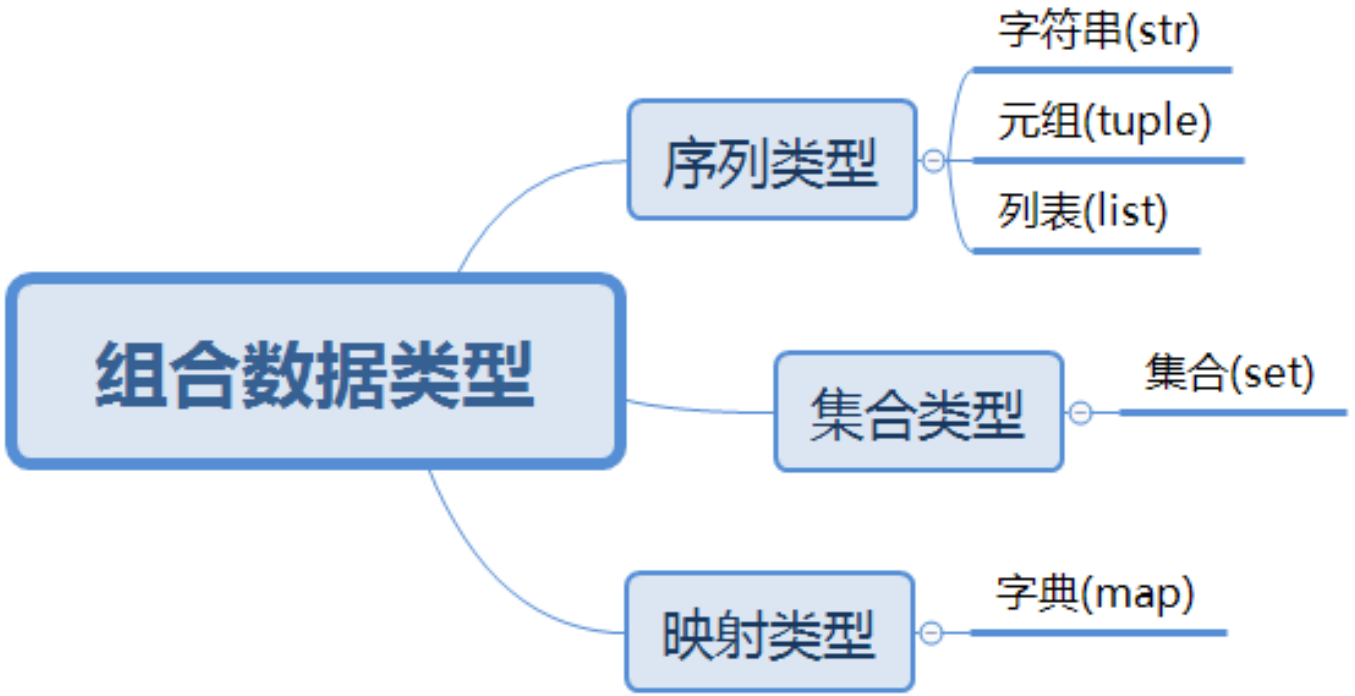
```
>>> t = {}
>>> t['201804'] = '小新'
>>> print(t)
{'201804': '小新'}
```

删除字典的元素

使用del关键字可以删除字典的对应元素或字典本身。

```
>>> d = {"小红":20,"小黑":35,"小绿":23}
>>> del d["小红"]
>>> print(d)
{'小黑': 35, '小绿': 23}
>>> d.clear()
>>> print(d)
{}
>>> del d
>>> print(d)
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print(d)
NameError: name 'd' is not defined
```

组合数据类型



列表的操作函数

列表类型继承序列类型特点，有一些通用的操作函数

操作函数	描述
len(ls)	列表ls的元素个数（长度)
min(ls)	列表ls中的最小元素(内部数据可比较)
max(ls)	列表ls中的最大元素 (内部数据可比较)
list(x)	将x转变成列表类型

列表的操作方法

列表类型存在一些操作方法，使用语法形式是：

<列表变量>.<方法名称>(<方法参数>)

方法	描述
ls.append(x)	在列表ls最后增加一个元素x
ls.insert(index, x)	在列表ls第index位置增加元素x
ls.clear()	删除ls中所有元素
ls.pop(index)	将列表ls中第index项元素取出并删除该元素
ls.remove(x)	将列表中出现的第一个元素x删除
ls.reverse()	列表ls中元素反转
ls.copy()	生成一个新列表，复制ls中所有元素

1、ls.append(x)在列表ls最后增加一个元素x：

```
>>> lt = ["1010","10.10","Python"]
>>> lt.append(1010)
>>> print(lt)
['1010', '10.10', 'Python', 1010]
>>> lt.append([1010,1111])
>>> print(lt)
['1010', '10.10', 'Python', 1010, [1010, 1111]]
```

2、ls.append(x)仅用于在列表中增加一个元素，如 果希望

增加多个元素，可以使用加号，将两个列表合并。

```
>>> lt = ["1010","10.10","Python"]
>>> ls = [1010,[1010,1111]]
>>> ls +=lt
>>> print(ls)
[1010, [1010, 1111], '1010', '10.10', 'Python']
```

3、ls.insert(i, x)在列表ls中序号i位置上增加元素x，序号i之后的元素序号依次增加。

```
>>> lt = ["1010","10.10","Python"]
>>> lt.insert(1,1010)
>>> print(lt)
['1010', 1010, '10.10', 'Python']
```

4、ls.clear()将列表ls的所有元素删除，清空列表。

```
>>> lt = ["1010","10.10","Python"]
>>> lt.clear()
>>> print(lt)
[]
```

5、ls.pop(i)将返回列表ls中第i位元素，并将该元素从列表中删除。

```
>>> lt = ["1010","10.10","Python"]
>>> print(lt.pop(1))
10.10
>>> print(lt)
['1010', 'Python']
```

6、ls.remove(x)将删除列表ls中第一个出现的x元素。

```
>>> lt = ["1010","10.10","Python"]
>>> lt.remove("10.10")
>>> print(lt)
['1010', 'Python']
```

7、可以使用Python保留字del对列表元素或片段进行删除，

del <列表变量>[<索引序号>] 或 del <列表变量>[<索引起始>: <索引结束>]

```
>>> lt = ["1010","10.10","Python"]
>>> del lt[1]
>>> print(lt)
['1010', 'Python']
>>> lt = ["1010","10.10","Python"]
>>> del lt[1:]
>>> print(lt)
['1010']
```

8、ls.reverse()将列表ls中元素进行逆序反转。

```
>>> lt = ["1010","10.10","Python"]
>>> lt.reverse()
>>> print(lt)
['Python', '10.10', '1010']
```

9、ls.copy() 复制ls中所有元素生成一个新列表。
注意：.copy()方法复制后赋值给 变量ls，将lt元素清空不影响新生成的变量ls。

```
>>> lt = ["1010","10.10","Python"]
>>> ls = lt.copy()
>>> lt.clear()
>>> print(ls)
['1010', '10.10', 'Python']
```

列表的引用

对于基本的数据类型，如整数或字符串，可以通过等号实现元素赋值。但对于列表类型，使用等号无法实现真正的赋值。其中，ls = lt语句并不是拷贝lt中元素给变量ls，而是新关联了一个引用，即ls和lt所指向的是同一套内容。

```
>>> lt = ["1010","10.10","Python"]
>>> ls = lt
>>> lt.clear()
>>> print(ls)
[]
```

ls.copy() 复制ls中所有元素生成一个新列表。
注意：.copy()方法复制后赋值给变量ls，将lt元素清空不影响新生成的变量ls。

```
>>> lt = ["1010","10.10","Python"]
>>> ls = lt.copy()
>>> lt.clear()
>>> print(ls)
['1010', '10.10', 'Python']
```

使用索引配合等号（=）可以对列表元素进行修改。

```
>>> lt = ["1010","10.10","Python"]
>>> lt[1] = 1010
>>> print(lt)
['1010', 1010, 'Python']
```

字典的操作函数

字典类型有一些通用的操作函数

操作函数	描述
len(d)	字典d的元素个数（长度）
min(d)	字典d中键的最小值
max(d)	字典d中键的最大值
dict()	生成一个空字典

字典的操作方法

字典类型存在一些操作方法，使用语法形式是：

<字典变量>.<方法名称>(<方法参数>)

操作方法	描述
d.keys()	返回所有的键信息
d.values()	返回所有的值信息
d.items()	返回所有的键值对
d.get(key, default)	键存在则返回相应值，否则返回默认值
d.pop(key, default)	键存在则返回相应值，同时删除键值对，否则返回默认值
d.popitem()	随机从字典中取出一个键值对，以元组(key, value)形式返回
d.clear()	删除所有的键值对

1、d.keys()返回字典中的所有键信息，返回结果是Python 的一种内部数据类型dict_keys，专用于表示字典的键。 如果希望更好的使用返回结果，可以将其转换为列表类型。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.keys()
dict_keys(['201803', '201802', '201801'])
>>> type(d.keys())
<class 'dict_keys'>
>>> list(d.keys())
['201803', '201802', '201801']
```

2、d.values()返回字典中的所有值信息，返回结果是Python 的一种内部数据类型dict_values。如果希望更好的使用 返回结果，可以将其转换为列表类型。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.values()
dict_values(['小白', '小红', '小明'])
>>> type(d.values())
<class 'dict_values'>
>>> list(d.values())
['小白', '小红', '小明']
```

3、d.items()返回字典中的所有键值对信息，返回结果是 Python的一种内部数据类型dict_items。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.items()
dict_items([('201803', '小白'), ('201802', '小红'), ('201801', '小明')])
>>> type(d.items())
<class 'dict_items'>
>>> list(d.items())
[('201803', '小白'), ('201802', '小红'), ('201801', '小明')]
```

4、d.get(key, default)根据键信息查找并返回值信息，如果 key存在则返回相应值，否则返回默认值，第二个元素 default可以省略，如果省略则默认值为空。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.get('201802')
'小红'
>>> d.get('201804')
>>> d.get('201804','不存在')
'不存在'
```

5、d.pop(key, default)根据键信息查找并取出值信息，如果 key存在则返回相应值，否则返回默认值，第二个元素 default可以省略，如果省略则默认值为空。相比d.get() 方法，d.pop()在取出相应值后，将从字典中删除对应的键值对。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.pop('201802')
'小红'
>>> print(d)
{'201803': '小白', '201801': '小明'}
>>> d.pop('201804','不存在')
'不存在'
```

6、d.popitem()随机从字典中取出一个键值对，以元组(key, value)形式返回。

取出后从字典中删除这个键值对。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> print(d.popitem())
('201803', '小白')
>>> d
{'201802': '小红', '201801': '小明'}
```

7、d.clear()删除字典中所有键值对。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> d.clear()
>>> print(d)
{}
```

8、此外，如果希望删除字典中某一个元素，可以使用 Python保留字del。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> del d['201801']
>>> print(d)
{'201803': '小白', '201802': '小红'}
```

9、字典类型也支持保留字in，用来判断一个键是否在字典 中。

如果在则返回True，否则返回False。

```
>>> d = {"201801":"小明","201802":"小红","201803":"小白"}
>>> '201801' in d
True
>>> '201804' in d
False
```

10、for循环返回的变量名是字典的键。如果需要获得键对应的值，

可以在语句块中通过get()方法或者<字典名>[<变量>]获得。

for <变量名> in <字典名>

<语句块>