# 6 集合

之前看到的列表和字符串都是一种有序序列，而集合 `set` 是一种无序的序列。

因为集合是无序的，所以当集合中存在两个同样的元素的时候，Python只会保存其中的一个（唯一性）；同时为了确保其中不包含同样的元素，集合中放入的元素只能是不可变的对象（确定性）。

基本功能包括关系测试和消除重复元素 。
集合对象还支持 union( 联合),intersection(交),difference(差)和 sysmmetric difference(对称差集)等数学运算。

## 6.1 集合生成

可以用 `set()` 函数来显示的生成空集合：

```
a = set()  # 创建空集合,你必须使用set() 而不是 {} 。{}用于创建空字典;
type(a)
```

```
set
```

也可以使用一个列表来初始化一个集合：

```
a = set([1, 2, 3, 1])
a
```

```
{1, 2, 3}
```

集合会自动去除重复元素 `1` 。

可以看到，集合中的元素是用大括号 `{}` 包含起来的，这意味着可以用 `{}` 的形式来创建集合：

```
a = {1, 2, 3, 1}
a
```

```
{1, 2, 3}
```

但是创建空集合的时候只能用`set`来创建，因为在Python中`{}`创建的是一个空的字典：

```
s = {}
type(s)
```

```
dict
```

# 6.2 集合操作

假设有这样两个集合：

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
```

## 6.2.1并集

两个集合的并，返回包含两个集合所有元素的集合（去除重复）。

可以用方法`a.union(b)`或者操作`a | b`实现。

```
a.union(b)
```

```
{1, 2, 3, 4, 5, 6}
```

```
b.union(a)
```

```
{1, 2, 3, 4, 5, 6}
```

```
a | b
```

```
{1, 2, 3, 4, 5, 6}
```

## 6.2.2 交集

两个集合的交，返回包含两个集合共有元素的集合。

可以用方法 `a.intersection(b)` 或者操作 `a & b` 实现。

```
a.intersection(b)
```

```
{3, 4}
```

```
b.intersection(a)
```

```
{3, 4}
```

```
a & b
```

```
{3, 4}
```

```
print(a & b)
```

```
set([3, 4])
```

注意：一般使用print打印set的结果与表示方法并不一致。

## 6.2.3 差

`a` 和 `b` 的差集，返回只在 `a` 不在 `b` 的元素组成的集合。

可以用方法 `a.difference(b)` 或者操作 `a - b` 实现。

```
a.difference(b)
```

```
{1, 2}
```

```
a - b
```

```
{1, 2}
```

注意，`a - b` 与 `b - a` 并不一样，`b - a` 返回的是返回 b 不在 a 的元素组成的集合：

```
b.difference(a)
```

```
{5, 6}
```

```
b - a
```

```
{5, 6}
```

## 6.2.4 对称差

`a` 和 `b` 的对称差集，返回在 `a` 或在 `b` 中，但是不同时在 `a` 和 `b` 中的元素组成的集合。

可以用方法 `a.symmetric_difference(b)` 或者操作 `a ^ b` 实现（异或操作符）。

```
a.symmetric_difference(b)
```

```
{1, 2, 5, 6}
```

```
b.symmetric_difference(a)
```

```
{1, 2, 5, 6}
```

```
a ^ b
```

```
{1, 2, 5, 6}
```

## 6.2.5 包含关系

假设现在有这样两个集合：

```
a = {1, 2, 3}
b = {1, 2}
```

要判断 `b` 是不是 `a` 的子集，可以用 `b.issubset(a)` 方法，或者更简单的用操作 `b <= a` ：

```
b.issubset(a)
```

```
True
```

```
b <= a
```

```
True
```

与之对应，也可以用 `a.issuperset(b)` 或者 `a >= b` 来判断：

```
a.issuperset(b)
```

```
True
```

```
a >= b
```

```
True
```

方法只能用来测试子集，但是操作符可以用来判断真子集：

```
a <= a
```

```
True
```

自己不是自己的真子集：

```
a < a
```

```
False
```

# 6.3 集合方法

## 6.3.1 `add` 方法向集合添加单个元素

跟列表的 `append` 方法类似，用来向集合添加单个元素。

```
s.add(a)
```

将元素 `a` 加入集合 `s` 中。

```
t = {1, 2, 3}
t.add(5)
t
```

```
{1, 2, 3, 5}
```

如果添加的是已有元素，集合不改变：

```
t.add(3)
t
```

```
{1, 2, 3, 5}
```

## 6.3.2 `update` 方法向集合添加多个元素

跟列表的`extend`方法类似，用来向集合添加多个元素。

```
s.update(seq)
```

将`seq`中的元素添加到`s`中。

```
t.update([5, 6, 7])
# t.update({5, 6, 7})
t
```

```
{1, 2, 3, 5, 6, 7}
```

## 6.3.3 `remove` 方法移除单个元素

```
s.remove(ob)
```

从集合`s`中移除元素`ob`，如果不存在会报错。

```
t.remove(1)
t
```

```
{2, 3, 5, 6, 7}
```

```
t.remove(10)
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-6-3bc25c5e1ff4> in <module>()
----> 1 t.remove(10)


KeyError: 10
```

## 6.3.4 pop方法弹出元素

由于集合没有顺序，不能像列表一样按照位置弹出元素，所以`pop`方法删除并返回集合中任意一个元素，如果集合中没有元素会报错。

```
t.pop()
```

```
2
```

```
print(t)
```

```
{3, 5, 6, 7}
```

```
s = set()
# 报错
s.pop()
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-9-9f9e06c962e6> in <module>()
      1 s = set()
      2 # 报错
----> 3 s.pop()


KeyError: 'pop from an empty set'
```

## 6.3.5 discard 方法

作用与`remove`一样，但是当元素在集合中不存在的时候不会报错。

```
t.discard(3)
```

```
t
```

```
{5, 6, 7}
```

不存在的元素不会报错：

```
t.discard(20)
```

```
t
```

```
{5, 6, 7}
```

### 6.3.6 difference_update方法

```
a.difference_update(b)
```

从a中去除所有属于b的元素：

## 6.4 不可变集合

对应于元组（`tuple`）与列表（`list`）的关系，对于集合（`set`），**Python**提供了一种叫做不可变集合（`frozen set`）的数据结构。

使用`frozenset`来进行创建：

```
s = frozenset([1, 2, 3, 'a', 1])
s
```

```
frozenset({1, 2, 3, 'a'})
```

与集合不同的是，不可变集合一旦创建就不可以改变。

不可变集合的一个主要应用是用来作为字典的键，例如用一个字典来记录两个城市之间的距离：

```
flight_distance = {}
city_pair = frozenset(['Los Angeles', 'New York'])
flight_distance[city_pair] = 2498
flight_distance[frozenset(['Austin', 'Los Angeles'])] = 1233
flight_distance[frozenset(['Austin', 'New York'])] = 1515
flight_distance
```

```
{frozenset({'Los Angeles', 'New York'}): 2498,
 frozenset({'Austin', 'Los Angeles'}): 1233,
 frozenset({'Austin', 'New York'}): 1515}
```

由于集合不分顺序，所以不同顺序不会影响查阅结果：

```
flight_distance[frozenset(['New York','Austin'])]
```

```
1515
```

```
flight_distance[frozenset(['Austin','New York'])]
```

```
1515
```