【Python】集合set

集合类型

> ## 集合类型 --- set, frozenset
>
> *set* 对象是由具有唯一性的 hashable 对象所组成的无序多项集。 常见的用途包括成员检测、从序列中去除重复项以及数学中的集合类计算，例如交集、并集、差集与对称差集等等。 (关于其他容器对象请参看 dict, list 与 tuple 等内置类，以及 collections 模块。)
>
> 与其他多项集一样，集合也支持 x in set, len(set) 和 for x in set。 作为一种无序的多项集，集合并不记录元素位置或插入顺序。 相应地，集合不支持索引、切片或其他序列类的操作。
>
> 目前有两种内置集合类型，set 和 frozenset。 set 类型是可变的 --- 其内容可以使用 add() 和 remove() 这样的方法来改变。 由于是可变类型，它没有哈希值，且不能被用作字典的键或其他集合的元素。 frozenset 类型是不可变并且为 hashable --- 其内容在被创建后不能再改变；因此它可以被用作字典的键或其他集合的元素。
>
> 除了可以使用 set 构造器，非空的 set (不是 frozenset) 还可以通过将以逗号分隔的元素列表包含于花括号之内来创建，例如: {'jack', 'sjoerd'}。

可哈希

class set([*iterable*])

class frozenset([*iterable*])

返回一个新的 set 或 frozenset 对象，其元素来自于 *iterable*。 集合的元素必须为 hashable。 要表示由集合对象构成的集合，所有的内层集合必须为 frozenset 对象。 如果未指定 *iterable*，则将返回一个新的空集合。

集合可用多种方式来创建:

- 使用花括号内以逗号分隔元素的方式: {'jack', 'sjoerd'}
- 使用集合推导式: {c for c in 'abracadabra' if c not in 'abc'}
- 使用类型构造器: set(), set('foobar'), set(['a', 'b', 'foo'])

hashable -- 可哈希

一个对象的哈希值如果在其生命周期内绝不改变，就被称为 *可哈希* (它需要具有 __hash__() 方法)，并可以同其他对象进行比较 (它需要具有 __eq__() 方法)。可哈希对象必须具有相同的哈希值比较结果才会相同。

可哈希性使得对象能够作为字典键或集合成员使用，因为这些数据结构要在内部使用哈希值。

大多数 Python 中的不可变内置对象都是可哈希的；可变容器 (例如列表或字典) 都不可哈希；不可变容器 (例如元组和 frozenset) 仅当它们的元素均为可哈希时才是可哈希的。 用户定义类的实例对象默认是可哈希的。 它们在比较时一定不相同 (除非是与自己比较)，它们的哈希值的生成是基于它们的 id()

函数

`set` 和 `frozenset` 的实例提供以下操作：

`len(s)`

返回集合 *s* 中的元素数量（即 *s* 的基数）。

`x in s`

检测 *x* 是否为 *s* 中的成员。

`x not in s`

检测 *x* 是否非 *s* 中的成员。

`isdisjoint(other)`

如果集合中没有与 *other* 共有的元素则返回 `True`。 当且仅当两个集合的交集为空集合时，两者为不相交集合。

`issubset(other)`

`set <= other`

检测是否集合中的每个元素都在 *other* 之中。

`set < other`

检测集合是否为 *other* 的真子集，即 `set <= other and set != other`。

`issuperset(other)`¶

`set >= other`

检测是否 *other* 中的每个元素都在集合之中。

`set > other`

检测集合是否为 *other* 的真超集，即 `set >= other and set != other`。

`union(*others)`

`set | other | ...`

返回一个新集合，其中包含来自原集合以及 others 指定的所有集合中的元素。

`intersection(*others)`

`set & other & ...`

返回一个新集合，其中包含原集合以及 others 指定的所有集合中共有的元素。

`difference(*others)`

`set - other - ...`

返回一个新集合，其中包含原集合中在 others 指定的其他集合中不存在的元素。

`symmetric_difference(other)`

`set ^ other`

返回一个新集合，其中的元素或属于原集合或属于 *other* 指定的其他集合，但不能同时属于两者。

`copy()`

返回原集合的浅拷贝。


```
s = {1, 2, 3, 4}
x = {1, 1, 1, 1}
len(s)

print(x in s)
print(x not in s)
```

```
print(x.isdisjoint(s))
print(x.issubset(s))
print(x.issuperset(s))

print(x <= s)
print(x < s)

print(x | s)
print(x & s)
print(x - s)
print(x ^ s)
```

下表列出了可用于 set 而不能用于不可变的 frozenset 实例的操作：

update(*others)
set |= other | ...
更新集合，添加来自 others 中的所有元素。

intersection_update(*others)
set &= other & ...
更新集合，只保留其中在所有 others 中也存在的元素。

difference_update(*others)
set -= other | ...
更新集合，移除其中也存在于 others 中的元素。

symmetric_difference_update(other)
set ^= other
更新集合，只保留存在于集合的一方而非共同存在的元素。

add(elem)
将元素 *elem* 添加到集合中。

remove(elem)
从集合中移除元素 *elem*。 如果 *elem* 不存在于集合中则会引发 KeyError。

discard(elem)
如果元素 *elem* 存在于集合中则将其移除。

pop()
从集合中移除并返回任意一个元素。 如果集合为空则会引发 KeyError。

clear()
从集合中移除所有元素。


```
# 创建
t = [1,2,3]
s = set(t)

set()  # 空集合只能用函数,{}表示空字典

"""
添加:set.add(),set.update()
"""
s.add(0)
s.update({40, 60})

```

```
"""14 删除:set.pop(),set.remove(),set.discard()
"""
s.remove()   # 不安全
s.discard()  # 安全
s.pop()      # 弹出一个

min(s)
max(s)
len(s)
sum(s)
```

记忆图