Python3 集合set

Python3 集合set
- 集合概念
- 集合的基本操作
  - 1、添加元素
  - 2、移除元素
  - 3、计算集合元素个数
  - 4、清空集合
  - 4、判断元素是否在集合中存在
  - 集合内置方法完整列表
- 集合应用实例

集合概念

集合（set）是一个无序的不重复元素序列。

可以使用大括号 { } 或者 set() 函数创建集合，注意：**创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。**

创建格式：

```
parame = {value01,value02,...}
或者
set(value)
```

实例

```
>>>basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> print(basket)                      # 这里演示的是去重功能
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket                 # 快速判断元素是否在集合内
True
>>> 'crabgrass' in basket
False

>>> # 下面展示两个集合间的运算.
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b                              # 集合a中包含元素
{'r', 'd', 'b'}
>>> a | b                              # 集合a或b中包含的所有元素
```

```
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b                              # 集合a和b中都包含了的元素
{'a', 'c'}
>>> a ^ b                              # 不同时包含于a和b的元素
{'r', 'd', 'b', 'm', 'z', 'l'}
```

类似列表推导式，同样集合支持**集合推导式(Set comprehension)**:

实例

```
>>>a = {x for x in 'abracadabra' if x not in 'abc'}
>>> a
{'r', 'd'}
```

集合的基本操作

1、添加元素

语法格式如下:

```
s.add( x )
```

将元素 x 添加到集合 s 中，如果元素已存在，则不进行任何操作。

实例

```
thisset = set(("Google", "Yahoo", "Taobao"))
thisset.add("Facebook")
print(thisset)
#输出    {'Yahoo', 'Facebook', 'Taobao', 'Google'}
```

还有一个方法，也可以添加元素，且参数可以是列表，元组，字典等，语法格式如下:

```
s.update( x )
```

x 可以有多个，用逗号分开。

实例

```
thisset = set(("Google", "Yahoo", "Taobao"))
thisset.update({1,3})
print(thisset)
#输出  {'Google', 1, 'Taobao', 3, 'Yahoo'}

thisset.update([1,4],[5,6])
print(thisset)
#输出  {'Taobao', 1, 3, 4, 5, 6, 'Google', 'Yahoo'}
```

2、移除元素

语法格式如下：

```
s.remove( x )
```

将元素 x 从集合 s 中移除，如果元素不存在，则会发生错误。

实例

```
thisset = set(("Google", "Runoob", "Taobao"))
thisset.remove("Taobao")
print(thisset)
{'Yahoo', 'Google'}
thisset.remove("Facebook")   # 不存在会发生错误

Exception has occurred: KeyError
'Facebook'
  File "D:\python_study\1.py", line 6, in <module>
    thisset.remove("Facebook")
```

此外还有一个方法也是移除集合中的元素，且如果元素不存在，不会发生错误。格式如下所示：

```
s.discard( x )
```

实例

```
thisset = set(("Google", "Yahoo", "Taobao"))
thisset.discard("Facebook")  # 不存在不会发生错误
print(thisset)
#输出: {'Google', 'Taobao', 'Yahoo'}
```

我们也可以设置随机删除集合中的一个元素，语法格式如下：

```
s.pop()
```

脚本模式实例

```
thisset = set(("Google", "Yahoo", "Taobao", "Facebook"))
x = thisset.pop()

print(x)
```

输出结果：

```
$ python3 test.py
Yahoo
```

多次执行测试结果都不一样。

然而在交互模式，pop 是删除集合的第一个元素（排序后的集合的第一个元素）。

实例

```
>>>thisset = set(("Google", "Yahoo", "Taobao", "Facebook"))
>>>thisset.pop()
'Google'
>>>print(thisset)
{'Taobao', 'Yahoo', 'Facebook'}
>>>
```

3、计算集合元素个数

语法格式如下：

```
len(s)
```

计算集合 s 元素个数。

实例

```
>>>thisset = set(("Google", "Yahoo", "Taobao"))
>>>len(thisset)
3
```

4、清空集合

语法格式如下：

```
s.clear()
```

清空集合 s。

```
thisset = set(("Google", "Yahoo", "Taobao"))
thisset.clear()
print(thisset)
print(len(thisset))
#输出: set()
#      0
```

4、判断元素是否在集合中存在

语法格式如下：

```
x in s
```

判断元素 s 是否在集合 x 中，存在返回 True，不存在返回 False。

实例

```
thisset = set(("Google", "Yahoo", "Taobao"))
print("Yahoo" in thisset)
print("Facebook" in thisset)
#输出:  True
#       False
```

集合内置方法完整列表

方法	描述
add()	为集合添加元素
clear()	移除集合中的所有元素
copy()	拷贝一个集合
difference()	返回多个集合的差集
difference_update()	移除集合中的元素，该元素在指定的集合也存在。
discard()	删除集合中指定的元素
intersection()	返回集合的交集
intersection_update()	删除集合中的元素，该元素在指定的集合中不存在。
isdisjoint()	判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False。
issubset()	判断指定集合是否为该方法参数集合的子集。
issuperset()	判断该方法的参数集合是否为指定集合的子集
pop()	随机移除元素
remove()	移除指定元素
symmetric_difference()	返回两个集合中不重复的元素集合。
symmetric_difference_update()	移除当前集合中在另外一个指定集合相同的元素，并将另外一个指定集合中不同的元素插入到当前集合中。
union()	返回两个集合的并集
update()	给集合添加元素

	等价操作符	说明
所有集合类型		
len(s)		集合基数：集合s中元素个数
set([obj])		可变集合工厂函数：ojb必须是支持迭代的，由obj中的元素创建集合，否则创建一个空集合
frozenset([obj])		不可变集合工厂函数：执行方式好set()方法相同，但它返回的是不可变集合
	obj in s	成员测试
	obj not in s	非成员测试
	s == t	等价测试
	s != t	不等价测试
	s < t	（严格意义上）子集测试
s.issubset(t)	s <= t	子集测试
	s > t	（严格意义上）超集测试
s.issuperset(t)	s >= t	超集测试
s.union(t)	s | t	合并操作
s.intersection(t)	s & t	交集操作
s.difference(t)	s – t	差分操作
s.symmetric_fifference(t)	s ^ t	对称差分操作
s.copy()		赋值操作：返回s的（浅复制）副本
仅适用于可变集合		
s.update(t)	s |= t	（Union）修改操作：将t中的成员添加s
s.intersection_update(t)	s &= t	交集修改操作：s中仅包括s和t中共有的成员
s.difference_update(t)	s -= t	差修改操作：s中仅包括属于s但不属于t的成员
s.symmetric_difference_update(t)	s ^= t	对称差分修改操作：s中包括仅属于s或仅属于t的成员
s.add(obj)		加操作：将obj添加到s
s.remove(obj)		删除操作

	等价操作符	说明
s.discard(obj)		丢弃操作：remove()的友好版本，如果s中存在ojb，从s中删除它
s.pop()		Pop操作：移除并返回s中的任意一个值
s.clear()		清除操作：移除s中的所有元素

集合应用实例

```
#!/usr/bin/python3

basket1 = {'apple', 'orange', 'pear', 'banana'}
basket2 = {'apple', 'orange', 'grape'}

print(basket2.issubset(basket1))
print(basket2 <= basket1)

print(basket1 | basket2)
print(basket1.union(basket2))

print(basket1 & basket2)
print(basket1.intersection(basket2))

print(basket1 - basket2)
print(basket1.difference(basket2))

print(basket1 ^ basket2)
print(basket1.symmetric_difference(basket2))
```

答案：

```
False
False
{'grape', 'banana', 'apple', 'pear', 'orange'}
{'grape', 'banana', 'apple', 'pear', 'orange'}
{'orange', 'apple'}
{'orange', 'apple'}
{'pear', 'banana'}
{'pear', 'banana'}
{'grape', 'banana', 'pear'}
{'grape', 'banana', 'pear'}
```