```
#encoding=utf-8
#***************  集合类型  ********************

#***********  Part 1：  集合  ***********************

#数学上， 把 set 称做由不同的元素组成的集合，集合(set)的成员通常被称做集合元素(set
#elements)。Python 把这个概念引入到它的集合类型对象里。集合对象是一组无序排列的可
哈希的值。
#是的，集合成员可以做字典中的键。数学集合转为 Python 的集合对象很有效，集合关系测
试和 union、
#intersection 等操作符在 Python 里也同样如我们所预想地那样工作。
#
#和其他容器类型一样，集合支持用 in 和 not in 操作符检查成员，由 len() 内建函数得到
集
#合的基数(大小)，用 for 循环迭代集合的成员。但是因为集合本身是无序的，你不可以为集
合创建
#索引或执行切片(slice)操作，也没有键(keys)可用来获取集合中元素的值。
#
#集合(sets)有两种不同的类型，可变集合(set) 和 不可变集合(frozenset)。如你所想，对可
#变集合(set)，你可以添加和删除元素，对 不可变集合(frozenset)则不允许这样做。请注意，
可变
#集合(set)不是可哈希的，因此既不能用做字典的键也不能做其他集合中的元素。不可变集
合
#(frozenset)则正好相反，即，他们有哈希值，能被用做字典的键或是作为集合中的一个成员。


#创建集合类型和赋值
#集合与列表( [ ] )和字典( { } ) 不同，没有特别的语法格式。列表和字典可以分别用他们自
#己的工厂方法 list() 和 dict() 创建，这也是集合被创建的唯一方法 - 用集合的工厂方法
set()和 frozenset():

#s = set('cheeseshop')
#print(s)
#t = frozenset('bookshop')
#print(t)
#print(type(s))
#print(type(t))
```

```
#print('s: %s, t:%s ' %(len(s),len(t)))
#-->
#{'c', 'e', 'h', 'o', 'p', 's'}
#frozenset({'b', 'h', 'k', 'o', 'p', 's'})
#<class 'set'>
#<class 'frozenset'>
#s: 6, t:6

#访问集合中的值
#你可以遍历查看集合成员或检查某项元素是否是一个集合中的成员:

#s = set('Dave is DBA')
#t = frozenset('Dave come from anqing')
#for i in s:
#       print(i)
#-->
#a
#
#B
#e
#D
#i
#s
#A
#v
#--注意这里是无序的

#print('l' in s)
#print('l' in t)
#-->
#False
#False


#更新集合
#用各种集合内建的方法和操作符添加和删除集合的成员:
#s = set('Dave is DBA')
#t = frozenset('Dave come from anqing')
#
#s.add('Oracle')
#print(s)
#-->{'a', ' ', 'B', 'e', 'D', 'i', 's', 'A', 'v', 'Oracle'}
#
#s.update('China')
```

```
#print(s)
#-->{'a', ' ', 'C', 'B', 'e', 'D', 'i', 'h', 'n', 's', 'A', 'v', 'Oracle'}
#--通过这个，我们可以看出 add 和 update 的区别， add 把内容作为一个元素添加进去，
#--而 updte 是把内容分别放进去
#
#s.remove('D')
#print(s)
#-->{'a', ' ', 'C', 'B', 'e', 'i', 'h', 'n', 's', 'A', 'v', 'Oracle'}
#
#s -=set('China')
#print(s)
#-->{' ', 'B', 'e', 's', 'A', 'v', 'Oracle'}
#--这个是集合的相减操作

#只有可变集合能被修改。试图修改不可变集合会引发异常。
#t.add('Dave')
#-->
#AttributeError: 'frozenset' object has no attribute 'add'



#删除集合中的成员和集合
#删除集合本身，可以像删除任何 Python 对象一样，令集合超出它的作用范围，或调用 del 将他们直接清除出当前的名字空间。
#如果它的引用计数为零，也会被标记以便被垃圾回收。
# del s



#**********  Part 2： 集合类型操作符  **********

################标准类型操作符（所有的集合类型）
#成员关系 (in, not in)
#就序列而言，Python 中的 in 和 not in 操作符决定某个元素是否是一个集合中的成员。

#s = set('Dave is DBA')
#t = frozenset('Dave come from anqing')
#
#print('D' in s)
#print('D' not in s)
#-->
#True
#False
```

#集合等价/不等价
#等价/不等价被用于在相同或不同的集合之间做比较。两个集合相等是指，对每个集合而言，当
#且仅当其中一个集合中的每个成员同时也是另一个集合中的成员。
#你也可以说每个集合必须是另一个集合的一个子集， 即，s <= t 和 s >= t 的值均为真(True),
#或(s <= t and s>= t) 的值为真(True)。集合等价/不等价与集合的类型或集合成员的顺序无关，
#只与集合的元素有关。

#s=set('Dave come from anqing')
#s1=set('Dave')

#print(s==s1)
#-->False

#子集/超集
#Sets 用 Python 的比较操作符检查某集合是否是其他集合的超集或子集。“小于”符号(<, <=)
#用来判断子集，“大于”符号(>, >=)用来判断超集。

#print(s1<s)
#-->
#True

################集合类型操作符（所有的集合类型）

#联合(|)
#联合(union)操作和集合的 OR(又称可兼析取(inclusive disjunction))其实是等价的，两个集
#合的联合是一个新集合，该集合中的每个元素都至少是其中一个集合的成员，即，属于两个集合其
#中之一的成员。联合符号有一个等价的方法，union().

#s=set('Dave come from anqing')
#s1=set('Dave')
#print(s|s1)
#-->
#{'a', ' ', 'c', 'e', 'D', 'g', 'f', 'i', 'm', 'o', 'n', 'q', 'r', 'v'}

#交集(&)
#你可以把交集操作比做集合的 AND(或合取)操作。两个集合的交集是一个新集合，该集合中的每
#个元素同时是两个集合中的成员，即，属于两个集合的成员。交集符号有一个等价的方法，intersection().
#print(s&s1)

```
#-->{'a', 'e', 'D', 'v'}

#差补/相对补集( - )
#两个集合(s 和 t)的差补或相对补集是指一个集合 C，该集合中的元素，只属于集合 s，而不属
#于集合 t。差符号有一个等价的方法，difference().
#print(s-s1)
#-->{' ', 'c', 'g', 'f', 'i', 'm', 'o', 'n', 'q', 'r'}

#对称差分( ^ )
#和其他的布尔集合操作相似，对称差分是集合的 XOR(又称”异或“ (exclusive disjunction)).
#两个集合(s 和 t)的对称差分是指另外一个集合 C,该集合中的元素，只能是属于集合 s 或者集合 t
#的成员，不能同时属于两个集合。对称差分有一个等价的方法，symmetric_difference().
#print(s^s1)
#-->{' ', 'c', 'g', 'f', 'i', 'm', 'o', 'n', 'q', 'r'}

################集合类型操作符（仅适用于可变集合）

#(Union) Update ( |= )
#这个更新方法从已存在的集合中添加(可能多个)成员，此方法和 update()等价.

#s=set('David')
#s1=set('Dai')
#s|=s1
#print(s)
#-->{'a', 'D', 'i', 'v', 'd'}

#保留/交集更新( &= )
#保留(或交集更新)操作保留与其他集合的共有成员。此方法和 intersection_update()等价.
#s&=s1
#print(s)
#-->{'a', 'i', 'D'}

#差更新 ( -= )
#对集合 s 和 t 进行差更新操作 s-=t，差更新操作会返回一个集合，该集合中的成员是集合 s 去
#除掉集合 t 中元素后剩余的元素。此方法和 difference_update()等价.
#s-=s1
#print(s)
#-->{'d', 'v'}

#对称差分更新( ^= )
#对集合 s 和 t 进行对称差分更新操作(s^=t),对称差分更新操作会返回一个集合，该集合中
```

的成
#员仅是原集合 s 或仅是另一集合 t 中的成员。此方法和 symmetric_difference_update()等价.
#s^=s1
#print(s)
#-->{'d', 'v'}

#********** Part 3： 内建函数 **********
##########标准类型函数

#len()
#把集合作为参数传递给内建函数 len()，返回集合的基数(或元素的个数)。
#s=set('dave come from anqing')
#print(len(s))
#-->14

########## 集合类型工厂函数
#set() and frozenset()
#set()和 frozenset()工厂函数分别用来生成可变和不可变的集合。如果不提供任何参数，默认
#会生成空集合。如果提供一个参数，则该参数必须是可迭代的，
#即，一个序列，或迭代器，或支持迭代的一个对象，例如：一个文件或一个字典。

#********** Part 4： 集合类型内建方法 **********

##########方法（所有的集合方法）

#内建方法 copy() 没有等价的操作符。

#方法名称 操作
#s.issubset(t) 如果 s 是 t 的子集，则返回 True,否则返回 False
#s.issuperset(t) 如果 t 是 s 的超集，则返回 True,否则返回 False
#s.union(t) 返回一个新集合，该集合是 s 和 t 的并集
#s.intersection(t) 返回一个新集合，该集合是 s 和 t 的交集
#s.difference(t) 返回一个新集合，该集合是 s 的成员，但不是 t 的成员
#s.symmetric_difference(t) 返回一个新集合，该集合是 s 或 t 的成员，但不是 s 和 t 共有的
成员
#s.copy() 返回一个新集合，它是集合 s 的浅复制

########## 操作符和内建方法比较

#很多内建的方法几乎和操作符等价。我们说"几乎等价"，意思是它们间是有一个
#重要区别： 当用操作符时，操作符两边的操作数必须是集合。 在使用内建方法时，对象

也可以是
#迭代类型的。为什么要用这种方式来实现呢？ Python 的文档里写明： 采用易懂的
#set('abc').intersection('cbs') 可以避免用 set('abc') [and] 'cbs' 这样容易出错的构建方
#法。

##可变集合类型的方法
#方法名 操作
#s.update(t) 用 t 中的元素修改 s，即，s 现在包含 s 或 t 的成员
#s.intersection_update(t) s 中的成员是共同属于 s 和 t 的元素。
#s.difference_update(t) s 中的成员是属于 s 但不包含在 t 中的元素
#s.symmetric_difference_update(t) s 中的成员更新为那些包含在 s 或 t 中，但不 是 s 和 t 共有的元素
#s.add(obj) 在集合 s 中添加对象 obj
#s.remove(obj) 从集合 s 中删除对象 obj；如果 obj 不是集合 s 中的元素(obj not in s)，将引发 KeyError 错误
#s.discard(obj) 如果 obj 是集合 s 中的元素，从集合 s 中删除对象 obj；
#s.pop() 删除集合 s 中的任意一个对象，并返回它
#s.clear() 删除集合 s 中的所有元素