字符串、列表、元组、字典和集合，它们有很多相同点，都是由多个元素组合成的一个可迭代对象，它们都有一些可以共同使用的方法。

算数运算符

在Python里，常见的算数运算符，有一些可以使用于可迭代对象，它们执行的结果也稍有区别。

运算符	Python 表达式	结果	描述	支持的数据类型
+	[1, 2] + [3, 4]	[1, 2, 3, 4]	合并	字符串、列表、元组
-	{1,2,3,4} - {2,3}	{1,4}	集合求差集	集合
*	['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	复制	字符串、列表、元组
in	3 in (1, 2, 3)	True	元素是否存在	字符串、列表、元组、字典
not in	4 not in (1, 2, 3)	True	元素是否不存在	字符串、列表、元组、字典

+

加法运算符可以用于字符串、列表和元组，用来拼接多个可迭代对象，不能用于字典和集合(思考：为什么字典和集合不能使用)。

```
>>> "hello " + "world"
'hello world'
>>> [1, 2] + [3, 4]
[1, 2, 3, 4]
>>> ('a', 'b') + ('c', 'd')
('a', 'b', 'c', 'd')
```

-

减法只能用于集合里，用来求两个集合的差集。

```
>>> {1, 6, 9, 10, 12, 3} - {4, 8, 2, 1, 3}
{9, 10, 12, 6}
```

*****

加法运算符可以用于字符串、列表和元组，用来将可迭代对象重复多次，同样不能用于字典和集合。

```
>>> 'ab' * 4
'abababab'
>>> [1, 2] * 4
[1, 2, 1, 2, 1, 2, 1, 2]
>>> ('a', 'b') * 4
('a', 'b', 'a', 'b', 'a', 'b', 'a', 'b')
```

in

in和not in成员运算符可以用于所有的可迭代对象。但是需要注意的是，in 和 not in 在对字典进行判断时，是查看指定的key是否存在，而不是value.

```
>>> 'llo' in 'hello world'
True
>>> 3 in [1, 2]
False
>>> 4 in (1, 2, 3, 4)
True
>>> "name" in {"name":"chris", "age":18}
True
```

遍历

通过for ... in ... 我们可以遍历字符串、列表、元组、字典、集合等可迭代对象。

字符串遍历

```
>>> a_str = "hello world"
>>> for char in a_str:
...     print(char,end=' ')
...
h e l l o   w o r l d
```

列表遍历

```
>>> a_list = [1, 2, 3, 4, 5]
>>> for num in a_list:
...     print(num,end=' ')
...
1 2 3 4 5
```

元组遍历

```
>>> a_turple = (1, 2, 3, 4, 5)
>>> for num in a_turple:
...     print(num,end=" ")
1 2 3 4 5
```

带下标的遍历

可迭代对象都可以使用 enumerate 内置类进行包装成一个 enumerate 对象。对enumerate进行遍历，可以同时得到一个可迭代对象的下标和元素。

```
nums = [12, 9, 8, 5, 4, 7, 3, 6]

# 将列表 nums 包装成 enumerate 对象
for i, num in enumerate(nums): # i表示元素下标，num表示列表里的元素
    print('第%d个元素是%d' % (i, num))
```