**builtin function**

**filter(function, sequence)**：对sequence中的item依次执行function(item)，将执行结果为True的item组成一个List/String/Tuple（取决于sequence的类型）返回：  
>>> def f(x): return x % 2 != 0 and x % 3 != 0   
>>> filter(f, range(2, 25))   
[5, 7, 11, 13, 17, 19, 23]  
>>> def f(x): return x != 'a'   
>>> filter(f, "abcdef")   
'bcdef'

**map(function, sequence)** ：对sequence中的item依次执行function(item)，见执行结果组成一个List返回：  
>>> def cube(x): return x\*x\*x   
>>> map(cube, range(1, 11))   
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]  
>>> def cube(x) : return x + x   
...   
>>> map(cube , "abcde")   
['aa', 'bb', 'cc', 'dd', 'ee']  
另外map也支持多个sequence，这就要求function也支持相应数量的参数输入：  
>>> def add(x, y): return x+y   
>>> map(add, range(8), range(8))   
[0, 2, 4, 6, 8, 10, 12, 14]

**reduce(function, sequence, starting\_value)**：对sequence中的item顺序迭代调用function，如果有starting\_value，还可以作为初始值调用，例如可以用来对List求和：  
>>> def add(x,y): return x + y   
>>> reduce(add, range(1, 11))   
55 （注：1+2+3+4+5+6+7+8+9+10）  
>>> reduce(add, range(1, 11), 20)   
75 （注：1+2+3+4+5+6+7+8+9+10+20）

**lambda**：这是Python支持一种有趣的语法，它允许你快速定义单行的最小函数，类似与[C语言](http://www.07net01.com/tags-C%E8%AF%AD%E8%A8%80-0.html)中的宏，这些叫做lambda的函数，是从LISP借用来的，可以用在任何需要函数的地方：  
>>> g = lambda x: x \* 2   
>>> g(3)   
6   
>>> (lambda x: x \* 2)(3)   
6  
  
  
我们也可以**把filter map reduce 和lambda结合起来**用，函数就可以简单的写成一行。  
例如  
kmpathes = filter(lambda kmpath: kmpath,   
map(lambda kmpath: string.strip(kmpath),  
string.split(l, ':')))   
看起来麻烦，其实就像用语言来描述问题一样，非常优雅。  
对 l 中的所有[元素](http://www.07net01.com/tags-%E5%85%83%E7%B4%A0-0.html)以':'做分割，得出一个列表。对这个列表的每一个元素做[字符串](http://www.07net01.com/tags-%E5%AD%97%E7%AC%A6%E4%B8%B2-0.html)strip，形成一个列表。对这个列表的每一个元素做直接返回操作(这个地方可以加上过滤条件限制)，最终获得一个字符串被':'分割的列表，列表中的每一个字符串都做了strip，并可以对特殊字符串过滤。

[转] http://hi.baidu.com/black/item/307001d18715fc322a35c747

---------------------------------------------------------------

**lambda**[**表达式**](http://www.07net01.com/tags-%E8%A1%A8%E8%BE%BE%E5%BC%8F-0.html)返回一个函数对象例子：func = lambda x,y:x+yfunc相当于下面这个函数def func(x,y):return x+y注意def是语句而lambda是表达式下面这种情况下就只能用lambda而不能用def[(lambda x:x\*x)(x) [for](http://www.07net01.com/program/) x in range(1,11)]map，reduce，filter中的function都可以用lambda表达式来生成！**map(function,sequence)**把sequence中的值当参数逐个传给function，返回一个包含函数执行结果的list。如果function有两个参数，即map(function,sequence1,sequence2)。例子：求1\*1,2\*2,3\*3,4\*4map(lambda x:x\*x,range(1,5))返回值是[1,4,9,16]**reduce(function,sequence)**function接收的参数个数只能为2先把sequence中第一个值和第二个值当参数传给function，再把function的返回值和第三个值当参数传给function，然后只返回一个结果。例子：求1到10的累加reduce(lambda x,y:x+y,range(1,11))返回值是55。**filter(function,sequence)**function的返回值只能是True或False把sequence中的值逐个当参数传给function，如果function(x)的返回值是True，就把x加到filter的返回值里面。一般来说filter的返回值是list，特殊情况如sequence是string或tuple，则返回值按照sequence的类型。例子：找出1到10之间的奇数filter(lambda x:x%2!=0,range(1,11))返回值[1,3,5,7,9]如果sequence是一个stringfilter(lambda x:len(x)!=0,'hello')返回'hello'filter(lambda x:len(x)==0,'hello')返回''