**functools**，用于高阶函数(那些作用于函数或者返回其它函数的函数)

通常只要是**可以被当做函数调用的对象**就是这个模块的目标。

functools.partial(func, \*args, \*\*keywords)

#**函数装饰器**，返回一个新的partial对象。

调用partial对象和调用被修饰的函数func相同，只不过调用partial对象时传入的参数个数通常要少于调用func时传入的参数个数。**当一个函数func可以接收很多参数，而某一次使用只需要更改其中的一部分参数，其他的参数都保持不变时，partial对象就可以将这些不变的对象冻结起来**，这样调用partial对象时传入未冻结的参数，partial对象调用func时连同已经被冻结的参数一同传给func函数，从而可以简化调用过程。

**def** take(n, iterable):  
 """Return first \*n\* items of the iterable as a list.  
 >>> take(3, range(10))  
 [0, 1, 2]  
 """**return** list(itertools.islice(iterable, n))

**def** chunked(iterable, n):  
 """Break \*iterable\* into lists of length \*n\*:  
 >>> list(chunked([1, 2, 3, 4, 5, 6], 3))  
 [[1, 2, 3], [4, 5, 6]]  
 >>> list(chunked([1, 2, 3, 4, 5, 6, 7, 8], 3))  
 [[1, 2, 3], [4, 5, 6], [7, 8]]  
 """**return** iter(functools.partial(take, n, iter(iterable)), []) #iter(callable, sentinel)

iter(callable, sentinel) 会持续调用参数callable直至其返回sentinel

因此会使得iter(iterable)不断进行直至抛出错误

**黑科技**