 通常的for...in...循环中，in后面是一个数组，这个数组就是一个可迭代对象，类似的还有链表，字符串，文件。它可以是mylist = [1, 2, 3]，也可以是mylist = [x\*x for x in range(3)]。  
它的缺陷是所有数据都在内存中，如果有海量数据的话将会非常耗内存。

 生成器是可以迭代的，但只可以读取它一次。因为用的时候才生成。比如 mygenerator = (x\*x for x in range(3))，注意这里用到了()，它就不是数组，而上面的例子是[]。

 我理解的生成器(generator)能够迭代的关键是它有一个next()方法，工作原理就是通过重复调用next()方法，直到捕获一个异常。可以用上面的mygenerator测试。

 带有 yield 的函数不再是一个普通函数，而是一个生成器generator，可用于迭代，工作原理同上。

 yield 是一个类似 return 的关键字，迭代一次遇到yield时就返回yield后面(右边)的值。重点是：下一次迭代时，从上一次迭代遇到的yield后面的代码(下一行)开始执行。

 简要理解：yield就是 return 返回一个值，并且记住这个返回的位置，下次迭代就从这个位置后(下一行)开始。

 带有yield的函数不仅仅只用于for循环中，而且可用于某个函数的参数，只要这个函数的参数允许迭代参数。比如array.extend函数，它的原型是array.extend(iterable)。

 send(msg)与next()的区别在于send可以传递参数给yield表达式，这时传递的参数会作为yield表达式的值，而yield的参数是返回给调用者的值。——换句话说，就是send可以强行修改上一个yield表达式值。比如函数中有一个yield赋值，a = yield 5，第一次迭代到这里会返回5，a还没有赋值。第二次迭代时，使用.send(10)，那么，就是强行修改yield 5表达式的值为10，本来是5的，那么a=10

 send(msg)与next()都有返回值，它们的返回值是当前迭代遇到yield时，yield后面表达式的值，其实就是当前迭代中yield后面的参数。

 第一次调用时必须先next()或send(None)，否则会报错，send后之所以为None是因为这时候没有上一个yield(根据第8条)。可以认为，next()等同于send(None)