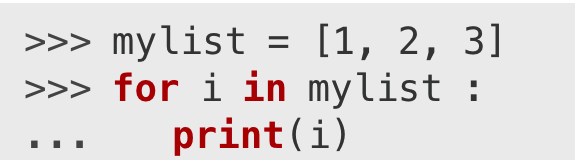
Python中的yield关键字

要理解yield，需要理解生成器，在理解生成器之前，我们需要先了解迭代器。

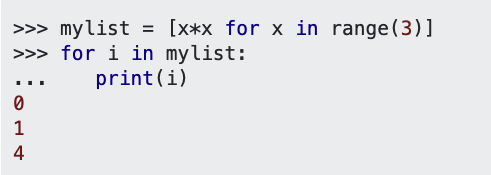
**可迭代对象**

当你建立了一个列表，你可以逐项的读取这个列表，这就叫一个可迭代对象：



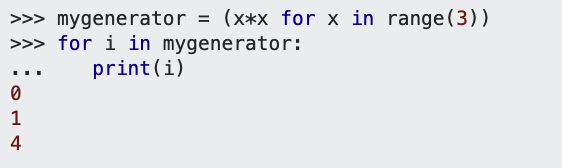
Mylist是一个可迭代的对象，当你使用一个列表生成式来建立一个列表的时候，就建立了一个可迭代的对象：

所以我们可以使用for…in…语法的叫做一个迭代器，比如列表、字符串、文件都是迭代器。我们经常使用它们是因为我们可以如愿的读取其中的元素，但是你把所有的值都存储到了内存中，如果有大量数据的话这个方式并不是我们想要的。

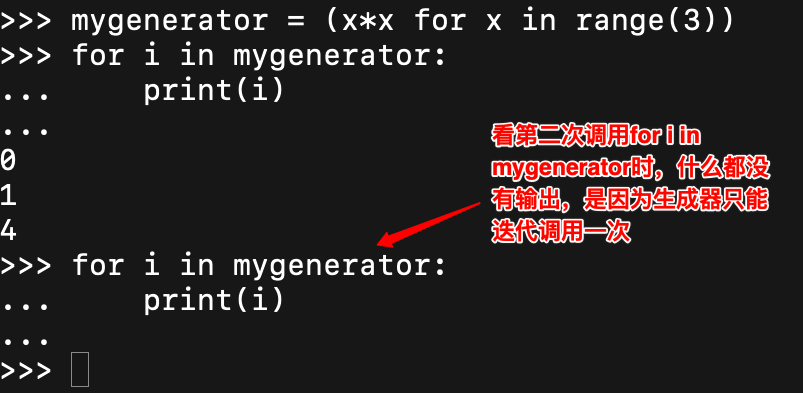


**生成器**

**生成器是可以迭代的，但是你只可以读取它一次，因为它并不把所有的值放入内存中，它是实时地生成数据：**

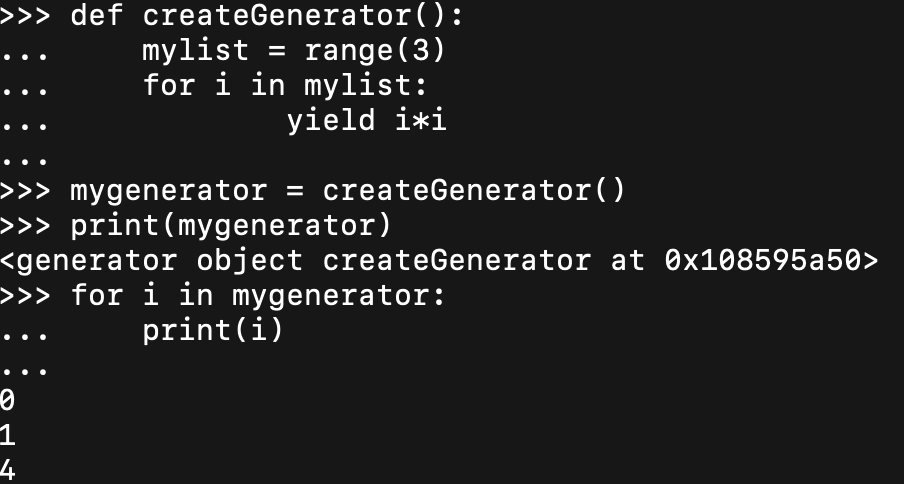


看起来除了把[]换成()外没什么不同。但是，你不可以再次使用for I in mygenerator，因为生成器只能被迭代一次：先计算出0，在计算出1，然后在计算出4，一个跟一个的。。。



**yield关键字**

**yield是一个类似于return的关键字，只是这个函数返回的是个生成器。**

****

**这个例子没什么用途，但是他让我们知道，这个函数会返回一大批我们只需要读一次的值。**

**为了精通yield，我们必须理解：当我们调用这个函数的时候，函数内部的代码并不立马执行，这个函数只是返回一个生成器对象，那么函数内的代码什么时候执行呢？当你使用for进行迭代的时候。**

**现在到关键点了**

**第一次迭代中你的函数会执行，从开始到达yield关键字，然后返回yield后的值作为第一次迭代的返回值。然后，每次执行这个函数都会继续执行你在函数内部定义的那个循环的下一次，再返回那个值，直到没有可以返回的。**

**如果生成器内部没有定义yield关键字，那么这个生成器被认为是空的。这种情况可能是因为循环进行没了。**