第八章：条件和循环

本章主题

z if 语句

z else 语句

z elif 语句

z 条件表达式

z while 语句

z for 语句

z break 语句

z continue 语句

z pass 语句

z else 语句 (再看)

z Iterators 迭代器

z 列表解析（List Comprehensions）

z 生成器表达式（Generator Expressions ）

我们发现，除了迭代器、列表解析、生成器表达式之外的都很简单，所以我重点看这三部分，总结这三部分。

目录

[8.1简单部分（条件和循环） 1](#_Toc350505926)

[8.2迭代器 3](#_Toc350505927)

[1、为什么要迭代器? 3](#_Toc350505928)

[2、如何迭代? 3](#_Toc350505929)

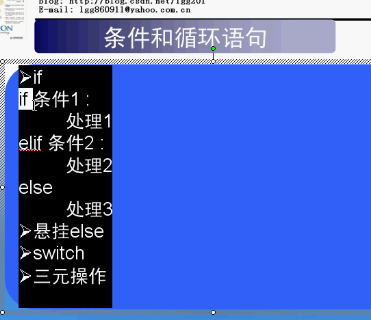
[3、使用迭代器 3](#_Toc350505930)

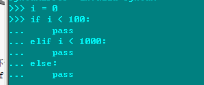
[4、其他 4](#_Toc350505931)

[8.3列表解析: 4](#_Toc350505932)

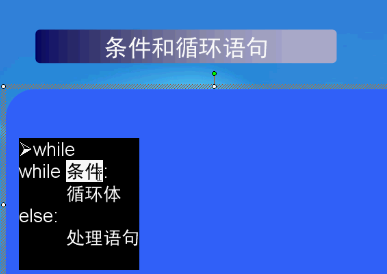
[8.4生成器 4](#_Toc350505933)

## 8.1简单部分（条件和循环）

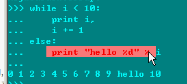




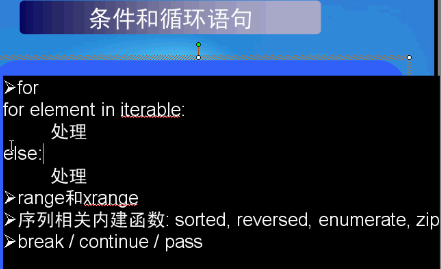
中间的是100到1000

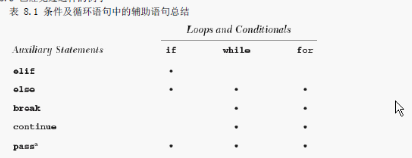


While和for都有一个else，这是跟其他语言不同的地方。



Else是在循环完成之后去执行。





## 8.2迭代器

迭代器用起来很灵巧, 你可以迭代不是序列但表现出序列行为的对象, 例如字典的 key , 一个文件的行, 等等. 当你使用循环迭代一个对象条目时, 你几乎不可能分辨出它是迭代器还是序列. 你不必去关注这些, 因为 Python 让它象一个序列那样操作.

一次性的，用一次之后就不能再使用了。

### 1、为什么要迭代器?

z 提供了可扩展的迭代器接口.

z 对列表迭代带来了性能上的增强.

z 在字典迭代中性能提升.

z 创建真正的迭代接口, 而不是原来的随机对象访问.

z 与所有已经存在的用户定义的类以及扩展的模拟序列和映射的对象向后兼容

z 迭代非序列集合(例如映射和文件)时, 可以创建更简洁可读的代码.

### 2、如何迭代?

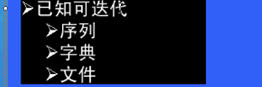
根本上说, 迭代器就是有一个 next() 方法的对象, 而不是通过索引来计数. 当你或是一个循环机制(例如 for 语句)需要下一个项时, 调用迭代器的 next() 方法就可以获得它. 条目全部取出后, 会引发一个 StopIteration 异常, 这并不表示错误发生, 只是告诉外部调用者, 迭代完成.

不过, 迭代器也有一些限制. 例如你不能向后移动, 不能回到开始, 也不能复制一个迭代器.

如果你要再次(或者是同时)迭代同个对象, 你只能去创建另一个迭代器对象. 不过, 这并不糟糕，因为还有其他的工具来帮助你使用迭代器.

reversed() 内建函数将返回一个反序访问的迭代器. enumerate() 内建函数同样也返回迭代器. 另外两个新的内建函数, any() 和 all() , 在 Python 2.5 中新增, 如果迭代器中某个/所有条目的值都为布尔真时，则它们返回值为真. 本章先前部分我们展示了如何在 for 循环中通过索引或是可迭代对象来遍历条目. 同时 Python 还提供了一整个 itertools 模块, 它包含各种有用的迭代器.

### 3、使用迭代器



序列

字典和文件是另外两个可迭代的 Python 数据类型. 字典的迭代器会遍历它的键(keys).

语句 for eachKey in myDict.keys() 可以缩写为 for eachKey in myDict , 例如:

文件对象生成的迭代器会自动调用 readline() 方法. 这样, 循环就可以访问文本文件的所有

行 .程 序 员 可 以 使 用 更 简 单 的 for eachLine in myFile 替 换

for eachLine in myFile.readlines() :

### 4、其他

python提供的三元操作: X if C else Y, 如果c输出x, 否则输出y

reversed()和enumerate()内建函数返回的是一个迭代器

iter(iterable) 用来创建一个迭代器

在迭代的时候, 不要尝试改变可变对象

## 8.3列表解析:

lambda, 动态创建函数

map(function, sequence), 对sequence的每一个元素应用function, 返回一个新的列表

filter(function, sequence), 对sequence的每一个元素应用function, 某个元素应用该function后如果返回false, 该元素将被从最后返回的列表中过滤掉

列表解析中可以使用多个循环, 来达到生成矩阵或多维序列的目的..例如【(x, y) for x in range(5) for y in range(3)】 越靠后的循环就相当于普通循环中的内层循环

## 8.4生成器

生成器表达式和列表解析表达式是一样的， 但是生成器表达式没有中括号， 单独使用的时候， 使用圆括号， 作为参数的时候， 不需要圆括号

生成器表达式可以被用作迭代

生成器表达式是lazy的

示例: 获取文件的最长的行的长度: logest = max(len(line.strip()) for line in file)