第八章 条件和循环

1.用 Python 字典给出更加优雅的case语句解决方案：

msgs = {'create': 'create item',

'delete': 'delete item',

'update': 'update item'}

default = 'invalid choice... try again!'

action = msgs.get(user.cmd, default)

2.三元操作符：

>>> x, y = 4, 3

>>> smaller = x if x < y else y

>>> smaller

3

3.for语句

**用于序列类型**

3.1通过序列项迭代

>>> nameList = ['Walter', "Nicole", 'Steven', 'Henry']

>>> for eachName in nameList:

... print eachName, "Lim"

3.2通过序列索引迭代=

另个方法就是通过序列的索引来迭代:

>>> nameList = ['Cathy', "Terry", 'Joe', 'Heather', 'Lucy']

>>> for nameIndex in range(len(nameList)):

... print "Liu,", nameList[nameIndex]

...

Liu, Cathy

Liu, Terry

Liu, Joe

Liu, Heather

Liu, Lucy

3.3===使用项和索引迭代===

两全其美的办法是使用内建的 enumerate() 函数, 它是 Python 2.3 的新增内容. 代码如下:

>>> nameList = ['Donn', 'Shirley', 'Ben', 'Janice', 'David', 'Yen', 'Wendy']

>>> for i, eachLee in enumerate(nameList):

... print "%d %s Lee" % (i+1, eachLee)

...

1 Donn Lee

2 Shirley Lee

3 Ben Lee

。。。

**用于迭代器类型**

迭代器对象有一个 next() 方法, 调用后返回下一个条目. 所有条目迭代完后, 迭代器引发一个 StopIteration 异常告诉程序循环结束. for 语句在内部调用 next() 并捕获异常.

使用迭代器做 for 循环的代码与使用序列条目几乎完全相同. 事实上在大多情况下, 你无法分辨出你迭代的是一个序列还是迭代器, 因此，这就是为什么我们在说要遍历一个迭代器时，实际上可能我们指的是要遍历一个序列，迭代器，或是一个支持迭代的对象（它有 next()方法）。

**内建函数range**

>>> for eachVal in range(2, 19, 3):

... print "value is:", eachVal

三种调用方法

range(start, end, step =1)

range(end)

range(start, end)

**与序列相关的内建函数**

sorted(), reversed(), enumerate(), zip()

下边是使用循环相关和序列相关函数的例子. 为什么它们叫"序列相关"呢? 是因为其中两个函数( sorted() 和 zip() )返回一个序列(列表), 而另外两个函数( reversed() 和 enumerate() ) 返回迭代器(类似序列)

4.迭代器和序列