PCS102　For 循环

概述

For 语句是主要的流程控制语句之一，其基本的用法和其他语言类似，特殊之处在于 Python 的 for 语句具有两种不同的风格形式，下面将会详细介绍。

应用

首先举个最简单的例子：

```
for item in [1, 2, 3]:
    print item
```

上面的代码其实就是遍历列表[1, 2, 3]，将其中每一个元素都打印出来。所有的循环语句中都可以使用这么两条语句：break 和 continue。break 表示要退出循环，continue 是说直接进入到下一轮的循环中去：

```
>>> for item in range(10):
...     if item<5:        # 遇到比 5 小的
...         continue      # 进入下一轮循环
...     else:            # 否则遇到的是大于等于 5 的
...         print item     # 输出
...         break         # 并直接退出循环
...
5
```

上面这个例子中的循环体也可以这么写：

```
    if item>=5:
        print item
        break
```

for 语句后面可以跟 else 子句，在这里 else 子句的含义是：如果 for 一直循环到了末尾，最后正常退出循环，那么随后就会执行它的 else 子句，否则由于 break 语句或异常等原因退出循环的，则不会执行 else 子句。其实在程序中常常会遇到这样的场景：对某个列表中每一个元素执行某个操作，如果成功执行则马上 break 跳出循环，如果遍历整个列表，发现没有一个元素满足要求，也就是意味着遍历失败，那么处理失败情况的语句就可以放在 else 子句中了，比如：

```
>>> for item in range(10):
...     if item<10:
...         continue
...     else:
...         print item
...         break
... else:
...     print '没有大于等于10的数字'
...
没有大于等于10的数字
```

如果熟悉 C 语言的话，便会看出 Python 的 for 和 C 的 for 之不同，除了表面上的语法差异，他们的含义更是大相径庭。要类比 Python 这种形式的 for 语句，可以想象某些语言的 for in 或者 foreach 之类的语法。

```
/* c 语言 */
for(int i=0; i<count; i++){
    ...
}

# python
for item in a_iterable:
    ...
```

简单地说，C 语言形式的 for 语句的工作原理是这样的：取第 0 个、第 1 个、第 2 个，一直取到最后一个。而迭代器呢，就是对迭代器取下一个、取下一个、取下一个，一直取到迭代器自己喊停为止。实际上 Python 的 for 语句是同时支持这两种风格的。先来剖析一下 Python 的 for 语句：

```
for item in obj:
    ...
```

如果 obj 对象实现了__iter__方法，就是说它是个迭代器，那这就是迭代器风格的 for 语句，

而上面这段代码也就等价于：

```
iterator = iter(obj) # 获取迭代器。
# iter(obj) 等价于 obj.__iter__()
try:
    item = iterator.next() # 取下一个
    ...
    item = iterator.next() # 取下一个
    ...
except StopIteration:     # 迭代器喊停
    pass
```

如果上面的 obj 对象实现了__getitem__方法，也就是说它支持索引操作，这就成了 C 语言风格的那种迭代器，这段代码便等价于：

```
try:
  item = obj[0]     # 取第 0 个
  # obj[0] 等价于 obj.__getitem__(0)
  ...
  item = obj[1]     # 取第 1 个
  ...
except IndexError:     # 取到最后一个
    pass
```

最后再测试一下：

```
>>> class Indexable(object):
...     def __getitem__(self, i):   # 定义__getitem__, 如果 i 大于 10，就停止迭代
...         if i>10:
...             raise StopIteration()
...         print 'get object %d'%i
...
>>> class Iterable(object):
...     def __init__(self):
...         self.counter = 0
...     def __iter__(self):
...         return self
...     def next(self):
...         if self.counter>10: # 如果计数器大于 10，就停止迭代
...             raise StopIteration()
...         print 'get next, current is %d'%self.counter
...         self.counter += 1 # 计数器增 1
...
>>> container = Indexable()
```

```
>>> for i in container:pass
...
get object 0
get object 1
get object 2
get object 3
get object 4
get object 5
get object 6
get object 7
get object 8
get object 9
get object 10
>>> container = Iterable()
>>> for i in container:pass
...
get next, current is 0
get next, current is 1
get next, current is 2
get next, current is 3
get next, current is 4
get next, current is 5
get next, current is 6
get next, current is 7
get next, current is 8
get next, current is 9
get next, current is 10
```

小结

For 循环是比较重要的控制流语句，本文介绍 for 语句的基本使用。

- For 循环及相关控制流：
  http://docs.python.org/tut/node6.html#SECTION006200000000000000000
  精巧地址：*http://bit.ly/3it1MI*
- The for statement：*http://docs.python.org/ref/for.html*
  精巧地址：*http://bit.ly/4sKUbJ*