

迭代

[2.7旧版教程](#)

阅读: 213559

如果给定一个list或tuple，我们可以通过 `for` 循环来遍历这个list或tuple，这种遍历我们称为迭代（Iteration）。

在Python中，迭代是通过 `for ... in` 来完成的，而很多语言比如C或者Java，迭代list是通过下标完成的，比如Java代码：

```
for (i=0; i<list.length; i++) {  
    n = list[i];  
}
```

可以看出，Python的 `for` 循环抽象程度要高于Java的 `for` 循环，因为Python的 `for` 循环不仅可以用在list或tuple上，还可以作用在其他可迭代对象上。

list这种数据类型虽然有以下标，但很多其他数据类型是没有下标的，但是，只要是可迭代对象，无论有无下标，都可以迭代，比如dict就可以迭代：

```
>>> d = {'a': 1, 'b': 2, 'c': 3}  
>>> for key in d:  
...     print(key)  
...  
a  
c  
b
```

因为dict的存储不是按照list的方式顺序排列，所以，迭代出的结果顺序很可能不一样。

默认情况下，dict迭代的是key。如果要迭代value，可以用 `for value in d.values()`，如果要同时迭代key和value，可以用 `for k, v in d.items()`。

由于字符串也是可迭代对象，因此，也可以作用于 `for` 循环：

```
>>> for ch in 'ABC':  
...     print(ch)  
...  
A  
B  
C
```

所以，当我们使用 `for` 循环时，只要作用于一个可迭代对象，`for` 循环就可以正常运行，而我们不太关心该对象究竟是list还是其他数据类型。

那么，如何判断一个对象是可迭代对象呢？方法是通过collections模块的Iterable类型判断：

```
>>> from collections import Iterable  
>>> isinstance('abc', Iterable) # str是否可迭代  
True  
>>> isinstance([1, 2, 3], Iterable) # list是否可迭代
```

```
True
>>> isinstance(123, Iterable) # 整数是否可迭代
False
```

最后一个小问题，如果要对list实现类似Java那样的下标循环怎么办？Python内置的`enumerate`函数可以把一个list变成索引-元素对，这样就可以在`for`循环中同时迭代索引和元素本身：

```
>>> for i, value in enumerate(['A', 'B', 'C']):
...     print(i, value)
...
0 A
1 B
2 C
```

上面的`for`循环里，同时引用了两个变量，在Python里是很常见的，比如下面的代码：

```
>>> for x, y in [(1, 1), (2, 4), (3, 9)]:
...     print(x, y)
...
1 1
2 4
3 9
```

小结

任何可迭代对象都可以作用于`for`循环，包括我们自定义的数据类型，只要符合迭代条件，就可以使用`for`循环。

参考源码

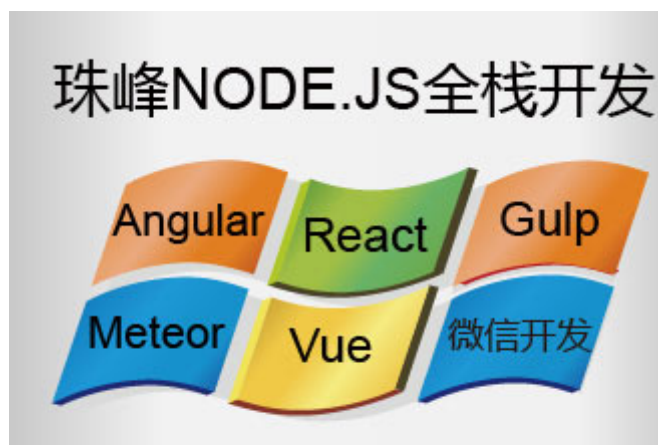
[do_iter.py](#)

感觉本站内容不错，读后有收获？

¥ 我要小额赞助，鼓励作者写出更好的教程

还可以分享给朋友

分享 你听云杉... , tianhaik... 等4人分享过





◀ [切片](#)

[列表生成式](#) ▶

评论

发表评论

Sign In to Make a Comment

[廖雪峰的官方网站](#)©2015
Powered by [iTranswarp.js](#)
由[阿里云](#)托管
[广告合作](#)



友情链接: [中华诗词](#) - [阿里云](#) - [SICP](#) - [4closure](#)