随笔 - 227  文章 - 1  评论 - 84

# [Python迭代器，可迭代对象，生成器](https://www.cnblogs.com/wswang/p/6047994.html)

#### 迭代器

迭代器（iterator）有时又称游标（cursor）是程式设计的软件设计模式，可在容器物件（container，例如链表或阵列）上遍访的界面，设计人员无需关心容器物件的内存分配的实现细节。

摘自维基百科

也就是说迭代器类似于一个游标，卡到哪里就是哪里，可以通过这个来访问某个可迭代对象的元素；同时，也不是只有Python有这个特性。比如C++的STL中也有这个，如vector<int>::iterator it。下面主要说一下Python中的可迭代对象和迭代器吧。

#### Python可迭代对象（Iterable）

Python中经常使用for来对某个对象进行遍历，此时被遍历的这个对象就是可迭代对象，像常见的list,tuple都是。如果给一个准确的定义的话，就是只要它定义了可以返回一个迭代器的\_\_iter\_\_方法，或者定义了可以支持下标索引的\_\_getitem\_\_方法(这些双下划线方法会在其他章节中全面解释)，那么它就是一个可迭代对象。

#### Python迭代器（iterator）

迭代器是通过next()来实现的，每调用一次他就会返回下一个元素，当没有下一个元素的时候返回一个StopIteration异常，所以实际上定义了这个方法的都算是迭代器。可以用通过下面例子来体验一下迭代器：

In [38]: s = 'ab'

In [39]: it = iter(s)

In [40]: it

Out[40]: <iterator at 0x1068e6d50>

In [41]: print it

<iterator object at 0x1068e6d50>

In [42]: it.next()

Out[42]: 'a'

In [43]: it.next()

Out[43]: 'b'

In [44]: it.next()

---------------------------------------------------------------------------

StopIteration Traceback (most recent call last)

<ipython-input-44-54f0920595b2> in <module>()

----> 1 it.next()

StopIteration:

自己实现一个迭代器，如下（参见官网文档）：

class Reverse:

"""Iterator for looping over a sequence backwards."""

def \_\_init\_\_(self, data):

self.data = data

self.index = len(data)

def \_\_iter\_\_(self):

return self

def next(self):

if self.index == 0:

raise StopIteration

self.index = self.index - 1

return self.data[self.index]

rev = Reverse('spam')

for char in rev:

print char

[output]

m

a

p

s

#### 生成器（Generators）

生成器是构造迭代器的最简单有力的工具，与普通函数不同的只有在返回一个值的时候使用yield来替代return，然后yield会自动构建好next()和iter()。是不是很省事。例如：

def reverse(data):

for index in range(len(data)-1, -1, -1):

yield data[index]

>>> for char in reverse('golf'):

... print char

...

f

l

o

g

生成器最佳应用场景是：你不想同一时间将所有计算出来的大量结果集分配到内存当中，特别是结果集里还包含循环。比方说，循环打印1000000个数，我们一般会使用xrange()而不是range()，因为前者返回的是生成器，后者返回的是列表（列表消耗大量空间）。

Help on built-in function range in module \_\_builtin\_\_:

range(...)

range(stop) -> list of integers

range(start, stop[, step]) -> list of integers

Return a list containing an arithmetic progression of integers.

range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0.

When step is given, it specifies the increment (or decrement).

For example, range(4) returns [0, 1, 2, 3]. The end point is omitted!

These are exactly the valid indices for a list of 4 elements.

class xrange(object)

| xrange(stop) -> xrange object

| xrange(start, stop[, step]) -> xrange object

|

| Like range(), but instead of returning a list, returns an object that

| generates the numbers in the range on demand. For looping, this is

| slightly faster than range() and more memory efficient.

#### iter()

将可迭代对象转化为迭代器。

In [113]: s = 'abc'

In [114]: s.next()

---------------------------------------------------------------------------

AttributeError Traceback (most recent call last)

<ipython-input-114-5e5e6532ea26> in <module>()

----> 1 s.next()

AttributeError: 'str' object has no attribute 'next'

In [115]: it = iter(s)

In [116]: it.next()

Out[116]: 'a'

#### 生成器表达式

和列表推导式唯一的区别就是中括号换成了小括号，如下：

In [119]: num = (i for i in range(10))

In [120]: sum(num)

Out[120]: 45

#### 参考

1. [Python官方文档](https://docs.python.org/2/tutorial/classes.html#iterators)
2. [Python进阶](http://wiki.jikexueyuan.com/project/interpy-zh/Generators/Iterable.html)