迭代器和生成器（iterator&generator）

2018年5月22日

9:18

**什么是迭代器：**

**迭代器是指用iter（可迭代对象）函数返回的对象（实例）**

**迭代器可以用next（it）函数获取可迭代对象的数据**

**迭代器函数：**

**iter（iterable）从可迭代对象中返回一个迭代器，iterable必须是一个能提供迭代器的可迭代对象**

**next（iterator）从迭代器iterator中获取下一条记录，如果无法获取下一条记录，则触发stopiteration异常**

**说明：**

**迭代器是访问可迭代对象的一种方式**

**迭代器只能向前取值，不会后退**

**用iter函数可以返回一个可迭代对象的迭代器**

**问题：**

**能否用迭代器访问range（100，1000）中的前三个数**

**A = range(100,1000)**

**R = iter(A)**

**Print("第一个数是：", next(R))**

**Print("第二个数是：", next(R))**

**Print("第三个数是：", next(R))**

**迭代器的用途：**

**迭代器对象能用next函数获取下一元素**

**示例：**

**用while循环语句来访问如下列表**

**L = [2, 3, 5, 7]**

it = iter(L) 
while True: 
try: 
x = next (it) 
print (x) 
except Stoplteration: 
print ( " " ) 
break 

**练习：**

**有一个集合：**

**S = {'唐僧'， ‘悟空’， ‘八戒’， ‘沙僧’}**

**用for语句来打印集合内的信息：**

**For x in s:**

**Print(x)**

**Else:**

**Print("遍历结束")**

**请将上面的for 语句改写为while,next,iter函数组合方式实现上述功能**

s 
for 
in 
x 
s: 
print (x) 
print ( ) 
print ( I' DI Fitwhilejüff- - 
it 
= iter(s) 
while True: 
try 
= next (it) 
x 
print (x) 
except Stoplteration: 
print ( 
break 

**获取迭代器需要在while循环外**

**生成器：（Generator）（python2.5及以后）**

**什么是生成器  
 生成器是能够动态提供数据的对象，生成器对象也是可迭代对象（实例）**

**生成器有两种：**

**生成器函数**

**生成器表达式**

**生成器函数定义：**

**含有yield**

**语句的函数是生成器函数，此函数被调用时将返回一个生成器对象**

**注：yield翻译为产生（或生成）**

**Yield 语句**

**语法：**

**yield表达式**

**说明：**

**yield用于def 函数中，目的是将此函数作为生成器函数使用**

**yield用来生成数据，供迭代器next（it）函数使用**

**示例：**

计算机生成了可选文字:
defmyylel
yield
2
yield
3
yielld
5
for
myyield()：
X
工n
print
（x）

**生成器函数说明：**

**生成器函数的调用将返回一个生成器对象，生成器对象是一个可迭代对象**

**在生成器函数调用return时会出现一个stopiteration异常来通知next（it）函数不再能提供数据**

**练习：**

**写一个生成器函数：**

**Def myinterger(n):**

**…**

5 def myinterger(n): 
while i < 
yield 
return 

**此生成器函数可以生成从0开始，的一系列的整数，到n结束（不包含n）**

**For x in myinterger(3)**

**Print(x)**

**It = iter(myinterger(2))**

**Print(next(it))**

**Print(next(it))**

**Print(next(it))**

**练习：**

**写一个生成器函数myodd（x）来生成一系列技术**

**如**

**myodd（10） 可以生成1,3,5,7,9**

 def myodd(x):  
 for i in range(x+1):  
 if i % 2 != 0:  
 yield i  
  
 return i  
  
a = myodd(10)  
while True:  
 try:  
 print(next(a))  
 except StopIteration:  
 print('over')  
 break

**生成器表达式：**

**语法：**

**（表达式 for 变量 in 可迭代对象 [if 真值表达式]）**

**注：[]里的内容可以省略**

**作用：**

**用推导式的形式生成一个新的生成器**

**示例：**

gen = (x \*\* 2 for x in range(1, 4))  
it = iter(gen)  
  
print(next(it)) #1  
print(next(it)) #4  
print(next(it)) #9  
print(next(it)) #StopIteration

**优点：**

**不占用内存空间**

**列表推导式和生成表达式的区别**

L = [1, 2, 3, 4]  
  
gen = (x for x in L) #gen绑定生成器  
lst = [x for x in L] # lst绑定了列表  
  
L[1] = 222  
  
for x in lst:  
 print(x) # 1 2 3 4 不变  
  
for x in gen:  
 print(x) #1 222 3 4，第二个数是222

**迭代工具函数：**

**迭代工具函数的作用是生成一个个性化的可迭代对象**

**函数**

**zip（iter1 [, iter2, iter3, …]）**

**返回一个zip对象，此对象用于生成一个元组，此元组的个数由最小的可迭代对象决定**

**Enumerate(iterable[,start])**

**生成待索引的枚举对象，返回迭代类型为索引-值对（index，value）对，默认索引从零开始，也可以使用start绑定**

**示例：**

number = [10086, 10000, 10010, 95588]  
names = ['中国移动', '中国电信', '中国联通']  
  
for n, a in zip(number, names):  
 print(a,'的客服号码是:', n)  
  
for x in zip(number, names):  
 print(x)  
  
d = dict(zip(names, number))  
print(d)  
  
for x in zip(range(10), number, names):  
 print(x)

中国移动 的客服号码是: 10086

中国电信 的客服号码是: 10000

中国联通 的客服号码是: 10010

(10086, '中国移动')

(10000, '中国电信')

(10010, '中国联通')

{'中国移动': 10086, '中国电信': 10000, '中国联通': 10010}

(0, 10086, '中国移动')

(1, 10000, '中国电信')

(2, 10010, '中国联通')

**此示例示意zip函数的内部实现机制**

def myzip(iter1, iter2):  
 it1 = iter(iter1)  
 it2 = iter(iter2)  
 while True:  
 x = next(it1)  
 y = next(it2)  
 yield (x, y)  
  
number = [10086, 10000, 10010, 95588]  
names = ['中国移动', '中国电信', '中国联通']  
  
for x in myzip(names, number):  
 print(x)

**枚举函数enumerate函数**

**格式**

**Enumerate(iterable[, start])**

**示例：**

**Names = ['中国移动', '中国电信', '中国联通']**

**For x in enumerate(names):**

**Print(x)**

python3 iterator.py 
1, 

**For x in enumerate(names, start=100):**

**Print(x)**

python3 iterat 
(100, 
(101, 
(102, 

**练习：**

**写一个程序，读入任意行的文字，当输入空行时结束输入**

**打印带有行号的输入结果：**

**如：python3 mytest.py**

**请输入：hello<回车>**

**请输入：world<回车>**

**请输入：bye<回车>**

**请输入：<回车>**

**打印如下：**

**第1行：hello**

**第2行：world**

**第3行：bye**

def mytest(): 
names=[] 
while True: 
= input 
I) 
a 
if 
break 
else 
names . append (a) 
for 
in enumerate(names,start=l) : 
x 
index, value = x 
print( , index, 
mytest() 
value ) 

**字节串bytes和字节数组bytearray：**

**字节串 bytes**

**作用：**

**存储以字节为单位的数据**

**说明：**

**字节串是不可变的字节序列**

**字节是0~255之间的整数**

**创建空字节串的字面值**

**b''**

**b""**

**b''''''**

**B''**

**B""**

**B''''''**

**B""""""**

**创建非空字节串的字面值：**

**b'ABCD'**

**b'\x41\x41'**

**字节串的构造函数bytes**

**Bytes() 生成一个空的字符串 等同于b''**

**Bytes(整型可迭代对象) # 用可迭代对象初始化一个字节串**

**bytes（整数n） 生成n个值为零的 字节串**

**bytes（字符串，encoding='utf-8'）**

**用字符串的转换编码生成一个字节串**

**字节串的运算**

**+ += \* \*=**

**< <= > >= == !=**

**In / not in**

**索引/切片**

**示例：**

**B = b'abc' + b '123' # b= b'abc123'**

**B += b'ABC' #b=b'abc123ABC'**

**b'ABD' > b'ABC #True**

**B = b'ABCD'**

**65 in b #True**

**b'A' in b #True**

**用于序列函数：**

**Len(x), max(x), min(x),sum(x), all(x),any(x)**

**比较bytes与str的区别**

**bytes存储字节（0~255）**

**str存储Unicode字符（0~65535）**

**bytes与str转换**

**编码（encode）**

**Str -----------> bytes**

**b=s.encode('urf-8')**

**解码（decode）**

**Bytes--------------------> str**

**S = b.decode('utf-8')**

**练习：**

**1.用生成器函数primes(begin,end)，生成素数，给出其起始值begin和终止值end，生成此范围内的全部素数，不包含（stop）**

**如 ：**

**L = [x for x in primes(10,20)]**

**将得到列表L = [11, 13, 17, 19]**

def primes(begin, end):  
 while begin <= end:  
 for i in range(2, begin):  
 if begin % i == 0:  
 break  
 else:  
 yield begin  
 begin += 1  
  
L = [x for x in primes(10,20)]  
print(L)

**2,仿制range函数的功能，写一个生成器函数myrange，要求功能与range功能相近，能实现一个，两个，三个参数传参，生成正向的整数**

**如：**

**For x in myrange(1,10,3):**

**Print(x) #1, 4, 7**

def myrange(start, end = None, step = 1):  
 if end is None:  
 end = start  
 start = 0  
  
 if start < end and step > 0:  
 while start < end:  
 yield start  
 start += step  
 elif start > end and step < 0:  
 while start < end:  
 yield start  
 start += step  
  
for x in myrange(1,10,3):  
 print(x)