itertools

<u>itertools模块提供的全部是处理迭代功能的函数，它们的返回值不是list，而是Iterator，只有用for循环迭代的时候才真正计算</u>

--itertools.count([0])
默认开始是0自然数
创建一个无限的自然数iterator
FE:

```
>>> import itertools
>>> natuals = itertools.count(0)
>>> natuals = itertools.count()
>>> for i in natuals:
 print (i)

0
1
```

--itertools.cycle(iterator)
将一个序列无xian循环下去
FE:

```
>>> cc = itertools.cycle('456')
>>> for c in cc:
 print(c)

4
5
6
4
5
6
```

--itertools.repeat(元素[,最大重复次数])
FE:

```
>>> np = itertools.repeat('1A.', 3)
>>> for i in np:
 print(i)

1A.
1A.
1A.
```

--itertools.takewhile(func,one class of itertools)
无限序列虽然可以无限迭代下去，但是通常我们会通过takewhile()等函数根据条件判断来截取出一个有限的序列
FE:

```
>>> ns = itertools.takewhile(lambda x: x <= 10, [4,5,596])
>>> list(ns)
[4, 5]
>>> natuals = itertools.count(1)
```

这里必须接着使用takewhile！！！若是别的地方使用，则会None

```
>>> ns = itertools.takewhile(lambda x: x <= 10, natuals)
>>> list(ns)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

--itertools.chain(oneiter,twoiter,...,niter)
将这些迭代对象串联起来，形成更大的迭代器

--itertools.groupby(itera[,func])
groupby()把迭代器中相邻的重复元素挑出来放在一起,func作用于迭代对象中的每个元素
FE:

```
>>> for i in itertools.groupby('AAABBBCCCAAA'):
 print(i)


('A', <itertools._grouper object at 0x00000024B071ABA8>)
('B', <itertools._grouper object at 0x00000024B0713EB8>)
('C', <itertools._grouper object at 0x00000024B071ABA8>)
```

```
('A', <itertools._grouper object at 0x00000024B0713EB8>)
>>> for key,group in itertools.groupby('AAABBBCCCAAA'):
 print(key,' is ',list(group))


A  is  ['A', 'A', 'A']
B  is  ['B', 'B', 'B']
C  is  ['C', 'C', 'C']
A  is  ['A', 'A', 'A']

>>> for key, group in itertools.groupby('AaaBBbcCAAa', lambda c: c.upper()):
...     print(key, list(group))
...
A ['A', 'a', 'a']
B ['B', 'B', 'b']
C ['c', 'C']
A ['A', 'A', 'a']
```

=======================2017/3/10==

--组合生成器

迭代器	参数	结果
product()	p, q, ... [repeat=1]	cartesian product, equivalent to a nested for-loop
permutations()	p[, r]	r-length tuples, all possible orderings, no repeated elements
combinations()	p, r	r-length tuples, in sorted order, no repeated elements
combinations_with_replacement()	p, r	r-length tuples, in sorted order, with repeated elements
product('ABCD', repeat=2)		AA AB AC AD BA BB BC BD CA CB CC CD DA DB DC DD
permutations('ABCD', 2)		AB AC AD BA BC BD CA CB CD DA DB DC
combinations('ABCD', 2)		AB AC AD BC BD CD
combinations_with_replacement('ABCD', 2)		AA AB AC AD BB BC BD CC CD DD