# itertools模块

- itertools模块能够快速创建迭代器。
- itertools模块一般分为三类，列举了一些常用的迭代器：
  - 无限迭代器
    - count
    - cycle
    - repeat
  - 终止于最短输入序列的迭代器
    - chain
    - compress
    - ifilter和ifilterfalse
    - imap
    - izip和izip_longest
  - 组合生成器
    - product
    - permutations
    - combinations和combinations_with_replacement

## 无限迭代器

### count

```
itertools.count(start=0,step=1)
```

- 返回以start为开头，步长step的值。

### cycle

```
itertools.cycle(iterable)
```

- 保存对象的副本，并无限重复返回每一个元素。

### repeat

```
itertools.repeat(object[,times])
```

- 重复返回对象[次]。

## 终止于最短输入序列的迭代器

- 这部分包括如下：

```
chain()
compress()
dropwhile()
groupby()
ifilter()
ifilterfalse()
islice()
imap()
starmap()
tee()
takewhile()
izip()
izip_longest()
```

### chain

```
itertools.chain(*iterables)
```

- 将所有iterable拼接后迭代返回。

### compress

```
itertools.compress(data,selectors)
```

- 返回data中对应selectors为True的元素。
- 相当于把selectors当做滤镜套在了data上。

### ifilter和ifilterfalse

```
itertools.ifilter(predicate,iterable)
itertools.ifilterfalse(predicate,iterable)
```

- 返回predicate后结果为True：ifilter/False：ifilterfalse的iterable元素。

### imap

```
itertools.imap(function,*iterables)
```

- 返回迭代序列中每一个元素经过function处理后的值。

### izip和izip_longest

```
itertools.izip(*iterables)
```

```
itertools.izip_longest(*iterables[,fillvalue=none])
```

- izip:用最短iterable来zip
- izip_longest:用最长序列来zip，短序列填充fillvalue

## 组合生成器

### product

```
itertools.product(*iterables[,repeat=1])
```

- 对*iterables进行笛卡尔积运算。

### permutations

```
itertools.permutations(iterable[,r])
```

- 返回连续长度为r（默认为最大长度）的迭代对象。

```
import itertools

digi=[1,2,3]
for item in itertools.permutations(digi,2):
    print item
for item in itertools.permutations(range(3)):
    print item

(1, 2)
(1, 3)
(2, 1)
(2, 3)
(3, 1)
(3, 2)
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

### combinations和combinations_with_replacement

```
itertools.combinations(iterable, r)
itertools.combinations_with_replacement(iterable, r)
```

- combinataions与permutations类似，但由前到后返回不重复（索引组合）的迭代。
- combinations_with_replacement与combinataions类似，但是将自身索引也作为一次对象。

```
import itertools

digi=[1,2,3]
for item in itertools.combinations(digi,2):
    print item
print "\n"
for item in itertools.combinations(range(3),2):
    print item

(1, 2)
(1, 3)
(2, 3)

(0, 1)
(0, 2)
(1, 2)
```

```
import itertools

digi=[1,2,3]
for item in itertools.combinations_with_replacement(digi,2):
    print item

(1, 1)
(1, 2)
(1, 3)
(2, 2)
(2, 3)
(3, 3)
```