itertools模块

> python 3.6.5

- itertools模块能够快速创建迭代器。
- itertools模块里由一堆奇奇怪怪的东西：
  - 无限迭代器
    - count
    - cycle
    - repeat
  - 终止于最短输入序列的迭代器
    - chain
    - compress
    - takewhile和filterfalse
    - zip_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)`

```
>>> for i in itertools.count():
    print(i)

0
1
2
3
4
...
```

- 返回以start为开头，步长step的值。

cycle：复读机一号

- `itertools.cycle(iterable)`

```
>>> for i in itertools.cycle([2,3,4,5]):
    print(i)

2
3
4
5
2
3
4
5
2
...
```

- 保存对象的副本，并无限重复返回每一个元素。

repeat：复读机二号

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

```
>>> for i in itertools.repeat('雪碧'):
    print(i)

雪碧
雪碧
雪碧
雪碧
雪碧
...
```

- 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)`

```
>>> a='把你的心'
>>> b=' 我的心'
>>> c=' 串一串'
>>> for i in itertools.chain(a,b,c):
    print(i)

把
你
的
心

我
的
心

串
一
串
```

- 将所有iterable拼接后迭代返回。

compress

- `itertools.compress(data,selectors)`

```
>>> selec=[True,False,42,0,-42,'shuang']
>>> items=['mole','xiangxiangji','tazhenmei','wodene','migang','shuangsile']
```

```
>>> for i in itertools.compress(items,selec):
    print(i)

mole
tazhenmei
migang
shuangsile
```

- 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。

takewhile和filterfalse

- `itertools.takewhile(predicate,iterable)`
- `itertools.filterfalse(predicate,iterable)`

```
>>> for i in itertools.takewhile(lambda x:x=='moyu',['moyu','jinye']):
    print(i)
moyu

>>> for i in itertools.filterfalse(lambda x:x=='moyu',['moyu','jinye']):
    print(i)
jinye
```

- 返回predicate后结果为True：takewhile/False：filterfalse的iterable元素。

zip_longest

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

```
>>> for i in itertools.zip_longest('Twilight Sparkle','Rainbow Dash','Fluttershy','Apple Bloom',fillvalue='Biu'):
    print(i)


('T', 'R', 'F', 'A')
('w', 'a', 'l', 'p')
('i', 'i', 'u', 'p')
('l', 'n', 't', 'l')
('i', 'b', 't', 'e')
('g', 'o', 'e', ' ')
('h', 'w', 'r', 'B')
('t', ' ', 's', 'l')
(' ', 'D', 'h', 'o')
('S', 'a', 'y', 'o')
('p', 's', 'Biu', 'm')
('a', 'h', 'Biu', 'Biu')
('r', 'Biu', 'Biu', 'Biu')
('k', 'Biu', 'Biu', 'Biu')
('l', 'Biu', 'Biu', 'Biu')
('e', 'Biu', 'Biu', 'Biu')
```

- zip_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])`

```
>>> for i in itertools.product('Tom','Jerry',repeat=1):
    print(i)

('T', 'J')
```

```
('T', 'e')
('T', 'r')
('T', 'r')
('T', 'y')
('o', 'J')
('o', 'e')
('o', 'r')
('o', 'r')
('o', 'y')
('m', 'J')
('m', 'e')
('m', 'r')
('m', 'r')
('m', 'y')
```

- 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)
```