**python itertools功能详解**

**介绍**

      itertools是python内置的模块，使用简单且功能强大，这里尝试汇总整理下，并提供简单应用示例；如果还不能满足你的要求，欢迎加入补充。

      使用只需简单一句导入：import itertools

**chain()**

      与其名称意义一样，给它一个列表如 lists/tuples/iterables，链接在一起；返回iterables对象。

letters = ['a', 'b', 'c', 'd', 'e', 'f']

booleans = [1, 0, 1, 0, 0, 1]

    print(list(itertools.chain(letters,booleans)))

#     ['a', 'b', 'c', 'd', 'e', 'f', 1, 0, 1, 0, 0, 1]

    print(tuple(itertools.chain(letters,letters[3:])))

#     ('a', 'b', 'c', 'd', 'e', 'f', 'd', 'e', 'f')

    print(set(itertools.chain(letters,letters[3:])))

#     {'a', 'd', 'b', 'e', 'c', 'f'}

    print(list(itertools.chain(letters,letters[3:])))

#     ['a', 'b', 'c', 'd', 'e', 'f', 'd', 'e', 'f']

    for item in list(itertools.chain(letters,booleans)):

        print(item)

**count()**

      生成无界限序列，count(start=0, step=1) ，示例从100开始，步长为2，循环10，打印对应值；必须手动break，count()会一直循环。

    i = 0

    for item in itertools.count(100,2):

        i += 1

        if i > 10 : break

        print(item)

**filterfalse ()**

      Python filterfalse(contintion,data) 迭代过滤条件为false的数据。如果条件为空，返回data中为false的项；

booleans = [1, 0, 1, 0, 0, 1]

numbers = [23, 20, 44, 32, 7, 12]

print(list(itertools.filterfalse(None,booleans)))

#     [0, 0, 0]

print(list(itertools.filterfalse(lambda x : x < 20,numbers)))

#    [23, 20, 44, 32]

**compress()**

返回我们需要使用的元素，根据b集合中元素真值，返回a集中对应的元素。

print(list(itertools.compress(letters,booleans)))

# ['a', 'c', 'f']

**starmap()**

      针对list中的每一项，调用函数功能。starmap(func,list[]) ；

starmap(pow, [(2,5), (3,2), (10,3)]) --> 32 9 1000

>>> from itertools import \*

>>> x = starmap(max,[[5,14,5],[2,34,6],[3,5,2]])

>>> for i in x:

>>> print (i)

14

34

5

**repeat()**

repeat(*object*[, *times*]) 重复times次；

repeat(10, 3) --> 10 10 10

**dropwhile()**

dropwhile(func, seq );当函数f执行返回假时, 开始迭代序列

dropwhile(lambda x: x<5, [1,4,6,4,1]) --> 6 4 1

**takewhile()**

takewhile(predicate, iterable)；返回序列，当predicate为true是截止。

takewhile(lambda x: x<5, [1,4,6,4,1]) --> 1 4

**islice()**

islice(seq[, start], stop[, step]);返回序列seq的从start开始到stop结束的步长为step的元素的迭代器

for i in islice("abcdef", 0, 4, 2):#a, c

    print i

**product()**

product(iter1,iter2, ... iterN, [repeat=1]);创建一个迭代器，生成表示item1，item2等中的项目的笛卡尔积的元组，repeat是一个关键字参数，指定重复生成序列的次数

    # product('ABCD', 'xy') --> Ax Ay Bx By Cx Cy Dx Dy

    # product(range(2), repeat=3) --> 000 001 010 011 100 101 110 111

for i in product([1, 2, 3], [4, 5], [6, 7]):

    print i

(1, 4, 6)

(1, 4, 7)

(1, 5, 6)

(1, 5, 7)

(2, 4, 6)

(2, 4, 7)

(2, 5, 6)

(2, 5, 7)

(3, 4, 6)

(3, 4, 7)

(3, 5, 6)

(3, 5, 7)

**permutations()**

permutations(p[,r]);返回p中任意取r个元素做排列的元组的迭代器

for i in permutations([1, 2, 3], 3):

    print i

(1, 2, 3)

(1, 3, 2)

(2, 1, 3)

(2, 3, 1)

(3, 1, 2)

(3, 2, 1)

**combinations()**

combinations(iterable,r);创建一个迭代器，返回iterable中所有长度为r的子序列，返回的子序列中的项按输入iterable中的顺序排序

note:不带重复

for i in combinations([1, 2, 3], 2):

    print i

(1, 2)

(1, 3)

(2, 3)

**combinations\_with\_replacement()**

同上, 带重复 例子:

for i in combinations\_with\_replacement([1, 2, 3], 2):

    print i

(1, 1)

(1, 2)

(1, 3)

(2, 2)

(2, 3)

(3, 3)

**应用示例**

求质数序列中1,3,5,7,9,11,13,15三个数之和为35的三个数；

def get\_three\_data(data\_list,amount):

    for data in list(itertools.combinations(data\_list, 3)):

        if sum(data) == amount:

            print(data)

#(7, 13, 15)

#(9, 11, 15)