Chapter 6

Itertools

Chapter 6

Itertools

In This Chapter:

1. itertools.product()
2. itertools.permutations()
3. itertools.combinations()
4. itertools.combinations_with_replacement()
5. Compress the String!

6.1 itertools.product()

This tool computes the cartesian product of input iterables.
It is equivalent to nested *for-loops*.
For example, product(A, B) returns the same as ((x,y) for x in A for y in B).

```
>>> from itertools import product
>>>
>>> print list(product([1,2,3],repeat = 2))
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
>>>
>>> print list(product([1,2,3],[3,4]))
[(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)]
>>>
>>> A = [[1,2,3],[3,4,5]]
>>> print list(product(*A))
[(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 3), (3, 4), (3, 5)]
>>>
>>> B = [[1,2,3],[3,4,5],[7,8]]
>>> print list(product(*B))
[(1, 3, 7), (1, 3, 8), (1, 4, 7), (1, 4, 8), (1, 5, 7), (1, 5, 8), (2, 3, 7), (2,
3, 8), (2, 4, 7), (2, 4, 8), (2, 5, 7), (2, 5, 8), (3, 3, 7), (3, 3, 8), (3, 4,
7), (3, 4, 8), (3, 5, 7), (3, 5, 8)]
```

You are given a two lists A and B. Your task is to compute their cartesian product A X B.

Example

```
A = [1, 2]
B = [3, 4]

AxB = [(1, 3), (1, 4), (2, 3), (2, 4)]
```

Note: A and B are sorted lists, and the cartesian product's tuples should be output in sorted order.

Input Format

The first line contains the space separated elements of list A.
The second line contains the space separated elements of list B.

Both lists have no duplicate integer elements.

Constraints

$0 < A < 30$

$0 < B < 30$

Output Format

Output the space separated tuples of the cartesian product.

Sample Input

```
1 2
3 4
```

Sample Output

```
(1, 3) (1, 4) (2, 3) (2, 4)
```

Code

```
from itertools import product
A = map(int, raw_input().split())
B = map(int, raw_input().split())
print ' '.join(map(str,list(product(A, B))))
```

6.2 itertools.permutations()

itertools.permutations(iterable[, r]) returns successive r length permutations of elements in an iterable.
If r is not specified or is None, then r defaults to the length of the iterable, and all possible full length permutations are generated.

Permutations are printed in a lexicographic sorted order. So, if the input iterable is sorted, the permutation tuples will be produced in a sorted order.

Sample Code

```
>>> from itertools import permutations
>>> print permutations(['1','2','3'])
<itertools.permutations object at 0x02A45210>
>>>
>>> print list(permutations(['1','2','3']))
[('1', '2', '3'), ('1', '3', '2'), ('2', '1', '3'), ('2', '3', '1'), ('3', '1', '2'), ('3', '2', '1')]
>>>
>>> print list(permutations(['1','2','3'],2))
[('1', '2'), ('1', '3'), ('2', '1'), ('2', '3'), ('3', '1'), ('3', '2')]
>>>
>>> print list(permutations('abc',3))
[('a', 'b', 'c'), ('a', 'c', 'b'), ('b', 'a', 'c'), ('b', 'c', 'a'), ('c', 'a', 'b'), ('c', 'b', 'a')]
```

You are given a string S.
Your task is to print all possible permutations of size K of the string in lexicographic sorted order.

Input Format

A single line containing the space separated string S and the integer value K.

Constraints

$0 < k \leq len(S)$

The string contains only *UPPERCASE* characters.

Output Format

Print the permutations of the string S on separate lines.

Sample Input

```
HACK 2
```

Sample Output

```
AC
AH
AK
CA
CH
CK
HA
HC
HK
KA
KC
KH
```

Explanation

All possible size 2 permutations of the string "**HACK**" are printed in lexicographic sorted order.

Code

```
from __future__ import print_function
from itertools import permutations
(word, parameter) = raw_input().split()
parameter = int(parameter)

b = list(permutations(word, parameter))
for item in sorted(b):
    print("".join(item))
```

6.3 itertools.combinations()

itertools.combinations(iterable, r) returns the r length subsequences of elements from the input iterable.

Combinations are emitted in lexicographic sorted order. So, if the input iterable is sorted, the combination tuples will be produced in sorted order.

Sample Code

```
>>> from itertools import combinations
>>>
>>> print list(combinations('12345',2))
[('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '3'), ('2', '4'), ('2', '5'), ('3', '4'), ('3', '5'), ('4', '5')]
>>>
>>> A = [1,1,3,3,3]
>>> print list(combinations(A,4))
[(1, 1, 3, 3), (1, 1, 3, 3), (1, 1, 3, 3), (1, 3, 3, 3), (1, 3, 3, 3)]
```

You are given a string S.
Your task is to print all possible combinations, up to size K, of the string in lexicographic sorted order.

Input Format

A single line containing the string S and integer value K separated by a space.

Constraints

$$0 < k \leq len(S)$$

The string contains only *UPPERCASE* characters.

Output Format

Print the different combinations of string S on separate lines.

Sample Input

```
HACK 2
```

Sample Output

```
A
C
H
K
AC
AH
AK
CH
CK
HK
```

Code

```
from itertools import combinations
(word, parameter) = raw_input().split()

word = sorted(word)
parameter = int(parameter)

for i in range(1, parameter+1):
    l = map("".join, list(combinations(word, i)))
    for item in sorted(l):
        print item
```

6.4 itertools.combinations_with_replacement()

itertools.combinations_with_replacement(iterable,r) returns r length subsequences of elements from the input iterable allowing individual elements to be *repeated more than once*.

Combinations are emitted in lexicographic sorted order. So, if the input iterable is sorted, the combination tuples will be produced in sorted order.

Sample Code

```
>>> from itertools import combinations_with_replacement
>>>
>>> print list(combinations_with_replacement('12345',2))
[('1', '1'), ('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '2'), ('2', '3'), ('2', '4'),
('2', '5'), ('3', '3'), ('3', '4'), ('3', '5'), ('4', '4'), ('4', '5'), ('5', '5')]
>>> A = [1,1,3,3,3]
>>> print list(combinations(A,2))
[(1, 1), (1, 3), (1, 3), (1, 3), (1, 3), (1, 3), (1, 3), (3, 3), (3, 3), (3, 3)]
```

You are given a string S. Your task is to print all possible size K replacement combinations of the string in lexicographic sorted order.

Input Format

A single line containing the string S and integer value K separated by a space.

Constraints

$0 < k \leq len(S)$

The string contains only *UPPERCASE* characters.

Output Format

Print the combinations with their replacements of string S on separate lines.

Sample Input

```
HACK 2
```

Sample Output

```
AA
AC
AH
AK
CC
CH
CK
HH
HK
KK
```

Code

```
from itertools import combinations_with_replacement
(word, parameter) = raw_input().split()
word = sorted(word)
parameter = int(parameter)

l = map("".join, list(combinations_with_replacement(word, parameter)))
for item in sorted(l):
        print item
```

6.5 Compress the String!

You need to appreciate the usefulness of the *groupby()* function of *itertools* .

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X,c) in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S.

Output Format

A single line of output consisting of the modified string.

Constraints

$1 \leq \mid S \mid \leq 10^4$

All the characters of S denote integers between 0 and 9.

Sample Input

1222311

Sample Output

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

Explanation

First, the character 1 occurs only once. It is replaced by (1, 1). Then the character 2 occurs three times, and it is replaced by (3, 2) and so on.

Code

```
from itertools import groupby
s = raw_input()
for key, group in groupby(s):
    print (len(list(group)), int(key)),
```

To read more about itertools , please refer to
https://docs.python.org/2/library/itertools.html