# 8

# модуль itertools

# модуль itertools

Данное расширение является сборником полезных итераторов, повышающих эффективность работы с циклами и генераторами последовательностей объектов. Это достигается за счет лучшего управления памятью в программе, быстрого выполнения подключаемых функций, а также сокращения и упрощения кода.

- Бесконечная итерация
- Комбинация значений
- Фильтрация последовательности
- Прочие итераторы

# Комбинация значений

**combinations**

Первая функция по комбинированию отдельных элементов последовательности принимает два аргумента. **Первый позволяет задать определенный объект, а второй – количество значений, которые будут присутствовать в каждом новом отрезке.** В данном примере демонстрируется работа в Python функции combinations библиотеки itertools при создании списка.

```
from itertools import combinations

data = list(combinations('DOG', 2))

print(data)
```

output: [('D', 'O'), ('D', 'G'), ('O', 'G')]

# Комбинация значений

**combinations_with_replacement**

Более продвинутая вариация предыдущего итератора предоставляет программе возможность делать выборку из отдельных элементов **с учетом их порядка**. В следующем образце кода показано использование combinations_with_replacement с уже известными аргументами.

```
from itertools import combinations_with_replacement
for i in combinations_with_replacement('DOG', 2):
print(''.join(i))
```

**Output:**

**DD**

**DO**

**DG**

**OO**

**OG**

**GG**

# Комбинация значений

**permutations**

Работа функции permutations модуля itertools в Python похожа на комбинацию со сменой порядка. **Однако в ней не допускается размещение идентичных элементов в одной группе**. Ниже приведен код, демонстрирующий поведение и результат выполнения этого метода в цикле for.

```
from itertools import permutations

for i in permutations('DOG', 2):
    print(''.join(i))
```

**Output:**

**DO**

**DG**

**OD**

**OG**

**GD**

**GO**

# Комбинация значений

**product**

Последний из комбинационных итераторов получает в качестве параметра массив данных, состоящий из нескольких групп значений. Функция product библиотеки itertools в Python 3 позволяет получить из введенной последовательности чисел или символов **новую совокупность групп** во всех возможных вариациях. Следующий пример показывает исполнение этого метода.

```
from itertools import product

data = list(product((0, 1), (2, 3)))

print(data)
```

output: [(0, 2), (0, 3), (1, 2), (1, 3)]

**combinations – комбинации, например,**

```
from itertools import combinations
cmb = list(combinations('ABC', 2))
print( cmb )
// Результат: [('A', 'B'), ('A', 'C'), ('B', 'C')]
```

**permutations – перестановки, например,**

```
from itertools import permutations
cmb = list(permutations('ABC'))
print( cmb )
// Результат: [('A', 'B', 'C'), ('A', 'C', 'B'),
// ('B', 'A', 'C'), ('B', 'C', 'A'), ('C', 'A', 'B'),
// ('C', 'B', 'A')]
```

**product – декартово произведение (все возможные слова заданной длины, составленные из данного алфавита), например:**

```
from itertools import product

cmb = list(product('ABC',repeat=2))

print( cmb )

// Результат: [('A', 'A'), ('A', 'B'), ('A', 'C'), ('B', 'A'),

// ('B', 'B'), ('B', 'C'), ('C', 'A'), ('C', 'B'), ('C', 'C')]
```

**Как видно из этих примеров, результат работы этих трёх функций – массив кортежей. В нём удобно работать с отдельными символами, но неудобно искать сочетания букв. Если нужно работать с сочетаниями букв, нужно «склеить» символы каждого кортежа в строки с помощью метода .join:**

```
from itertools import product

cmb = product('ABC',repeat=2)

cmb = list( map( "".join, cmb ) )

print( cmb )

// Результат: ['AA', 'AB', 'AC', 'BA', 'BB', 'BC', 'CA',

// 'CB', 'CC']
```

Игорь составляет таблицу кодовых слов для передачи сообщений, каждому сообщению соответствует своё кодовое слово. В качестве кодовых слов Игорь использует трёхбуквенные слова, в которых могут быть только буквы Ш, К, О, Л, А, причём буква К появляется ровно 1 раз. Каждая из других допустимых букв может встречаться в кодовом слове любое количество раз или не встречаться совсем. Сколько различных кодовых слов может использовать Игорь?

```
from itertools import product

p = product('ШКОЛА', repeat=3)

n = 0

for x in p:

  if x.count('К') == 1:

    n += 1

print(n)
```

Маша составляет шестибуквенные слова перестановкой букв слова КАПКАН. При этом она избегает слов с двумя подряд одинаковыми буквами. Сколько различных кодов может составить Маша?

```
from itertools import *
# В множестве состаляемых слов "w" автоматически удаляются
# дубликаты кортежей с перестановками "p",
# а команда "if" отсеивает перестановки с двумя подряд
# идущими одинаковыми буквами ("АА" или "КК"):
w = {p for p in permutations('КАПКАН')
    if ''.join(p).count('АА') + ''.join(p).count('КК') == 0}
print(len(w))
```

Маша составляет 5-буквенные коды из букв В, У, А, Л, Ь. Каждую букву нужно использовать ровно 1 раз, при этом буква Ь не может стоять на первом месте и перед гласной. Сколько различных кодов может составить Маша?

# 1 вариант решения. Так как количество букв в слове= количеству букв в алфавите

для построения множества всевозможных слов можно использовать функцию permutations из модуля itertools; затем остаётся выбрать и пересчитать подходящие слова:

```
from itertools import permutations

n = 0

for x in permutations('ВУАЛЬ'):

    s = ''.join(x)

    if s[0] != 'Ь' and 'ЬУ' not in s \

        and 'ЬА' not in s:

        n += 1

print(n)
```

# 2 вариант решения. Так как составляем слова.

для построения множества всевозможных слов можно использовать функцию product из модуля itertools; затем остаётся выбрать и пересчитать подходящие слова:

```
from itertools import product

p = product('ВУАЛЬ',repeat=5)

s = map(lambda x: ''.join(x), p)

n = 0

for x in s:

  if ((x.count('В')==1) and (x.count('У')==1) and

     (x.count('А')==1) and (x.count('Л')==1) and

     (x.count('Ь')==1)) and

     (x[0] != 'Ь') and (x.find('ЬУ')==-1 and

      x.find('ЬА')==-1):

      n += 1

print(n)
```

Все 4-буквенные слова, составленные из букв К, Л, Р, Т, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. КККК

2. КККЛ

3. КККР

4. КККТ

……

Запишите слово, которое стоит на 67-м месте от начала списка.

```
from itertools import product

print(*list(product('КЛРТ',repeat=4))[67-1])
```

Все 5-буквенные слова, составленные из 5 букв А, К, Л, О, Ш, записаны в алфавитном порядке.

Вот начало списка:

1. АААААА

2. ААААК

3. ААААЛ

4. АААОО

5. АААШ

6. ААКА

……

На каком месте от начала списка стоит слово ШКОЛА?

```
from itertools import product
ss=[ ]
for x in product('АКЛОШ',repeat=5):
    s=''.join(x)
    ss.append(s)
print(ss.index('ШКОЛА')+1)
```

Леонид составляет коды перестановкой букв слова ПАРИЖАНКА. При этом в этих кодах ровно один раз встречаются две идущие подряд гласные буквы. Сколько различных кодов может составить Леонид?