# itertools

## Введение

Основанный на итераторах код обеспечивает лучшие характеристики использования памяти, чем код, основанный на использовании списков. Поскольку итераторы не возвращает данные до тех пор, пока они не потребуются, отпадает необходимиость в хранении всего набора данных в памяти. Такая модель отложенной обработки сглаживает отрицательное влияние подкачки и других побочных эффектов, связанных с обработкой больших объёмов данных, на производительность.

### Определение

itertools - стандартный модуль Python, включающий в себя ряд функций, предназначенных для обработки последовательностей.

itertools относят к функциональным модулям, ведь он был вдохновлён такими языками программирования, как APL, Haskell, SML.

## Функции

### Бесконечные итераторы

Бесконечные итераторы - бесконечные последовательности, которые имеют паттерн.

| Итератор | Аргументы | Результат | Пример |
| --- | --- | --- | --- |
| count() | начало, [шаг] | начало, начало+шаг, начало+2\*шаг, … | count(10) --> 10 11 12 13 14 ... |
| cycle() | с | с0, с1, …, с0, с1, … | cycle('СТРОКА') --> С Т Р О К А С Т Р О К А |
| repeat() | элемент, [количество] | элемент, элемент, … бесконечно или столько сколько указано в количестве | repeat(10, 3) --> 10 10 10 10 |

#### Пример 1

Используется, когда необходим вечный цикл со знанием количествa пройденных итераций.

Ниже представлено сравнение скорости работы скрипта с count() и без него.

Без count() скорость выполнения составит 0.0007404560001305072 секунды.

i = 0  
while True:  
 if i == 200:  
 break  
 print(i)  
 i += 1

С count() скорость выполнения составит 0.0006238849996407225 секунды.

import itertools  
for i in itertools.count():  
 if i == 200:  
 break  
 print(i)

### Итераторы, завершающиеся на кратчайшей входной последовательности

| Итератор | Аргументы | Результат | Пример |
| --- | --- | --- | --- |
| accumulate() | последовательность, [функция] | начало, начало+шаг, начало+2\*шаг, … | accumulate([1, 2, 3, 4, 5]) --> 1 3 6 10 15 |
| chain() | p, …, q | p0, p1, p конечный, …, q1, …, q конечный | chain('abc', 'абв') --> a b c а б в |
| chain.from\_iterable() | итерационный | p0, p1, p конечный, …, q1, …, q конечный | chain.from\_iterable(['abc', 'абв']) --> a b c а б в |
| compress() | данные, выбор | (d[0] если s[0]), (d[1] if s[1]), … | compress('компр', [1, 0, 1]) --> к м |
| dropwhile() | предикат, последовательность | начинается при ложном предикате | dropwhile(lambda x: x<5), [1, 4, 7, 8, 9]) --> 7, 8, 9 |
| filterfalse() | предикат, последовательность | элементы последовательности, где предикат ложь | filterfalse(lambda x: x%2, range(10)) --> 0 2 4 6 8 |
| takewhile() | предикат, последовательность | seq[0], seq[1]? пока предикат ложь | takewhile(lambda x: x<5, [1, 4, 5]) --> 1 4 |
| groupby() | последовательность, ключ-функция | подитераторы, сгруппированные по значению ключа |  |
| islice() | последовательность, [начало,] конец [, шаг] | элементы последовательности[начало:конец:шаг] | islice('ABCDEFG', 2, None) --> C D E F G |
| pairwise() | последовательность | (p[0], p[1]), (p[2], p[3]), … | pairwise('мишка') --> ми иш шк ка |
| starmap() | func, seq | func(*seq[0]), func(*seq[1]), … | starmap(pow, [(2, 5), (3, 2)]) --> 32 9 |
| tee() | func, seq | func(*seq[0]), func(*seq[1]), … | starmap(pow, [(2, 5), (3, 2)]) --> 32 9 |
| zpi\_longest() | p, q, … | (p[0], q[0]), (p[1]), (p[1], q[1]), … | itertools.zip\_longest('abcd', 'xy', fillvalue) --> ax by c- d- |

### Комбинационные итераторы

Комбинационные итераторы - последовательности, которые перебирают комбинации.

| Итератор | Аргументы | Результат |
| --- | --- | --- |
| product() | последовательность, [повторы=1] | декартово произведение, эквивалентное вложенному циклу for |
| permutations() | последовательность, [п=1] | кортеж длиной r, все возможные порядки без повторяющихся элементов, позиция важна |
| combinations\_with\_replacement() | последовательность, [п=1] | кортеж длиной r, все возможные порядки c повторяющимися элементами, позиция не важна |

#### Пример 1

Ниже приведена таблица с примерами комбинационных итераторов.

| Пример | Резуьтат |
| --- | --- |
| product(‘abcd’, repeat=2) | aa ab ac ad ba bb bc bd ca cb cc cd da db dc dd |
| permutations(‘abcd’, 2) | ab ac ad ba bc bd ca cb cd da db dc |
| combinations(‘abcd’, 2) | ab ac ad bc bd cd |
| combinations\_with\_replacement(‘abcd’, 2) | aa ab ac ad bb bc bd cc cd dd |

#### Пример 2

Получить последовательность, состоящую из последовательностей, в которых первый элемент - масть карты, второй элемент её достоинство.

Решение:

import itertools  
  
dignities = [\*map(str, range(1, 11)), 'валет', 'дама', 'король', 'туз']  
suits = ['пики', 'трефы', 'черви', 'бубны']  
  
print(\*itertools.product(suits, dignities))  
  
"""  
Результат:  
 ('пики', '1') ('пики', '2') ('пики', '3')  
 ('пики', '4') ('пики', '5') ('пики', '6')  
 ('пики', '7') ('пики', '8') ('пики', '9')  
 ('пики', '10') ('пики', 'валет') ('пики', 'дама')  
 ('пики', 'король') ('пики', 'туз') ('трефы', '1')  
 ...  
 ('бубны', 'дама') ('бубны', 'король') ('бубны', 'туз')  
"""

#### Пример 3

Решим задание 8 задание ЕГЭ, используя разбираемый модуль.

Задание: (№ 3538) (Е. Джобс) Все 4-буквенные слова, составленные из букв П, Р, В, Д, А, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. АААА  
2. АААВ  
3. АААД  
4. АААП  
5. АААР  
6. ААВА  
...

Найдите номер первого слова в этом списке, которое не содержит гласных и одинаковых букв.

Решение:

import itertools  
  
seq = 'авдпр'  
without\_repeat = tuple(itertools.combinations(seq, 4))  
  
for idx, elem in enumerate(itertools.product(seq, repeat=4), 1):  
 if 'а' not in elem and elem in without\_repeat:  
 print(idx, elem)  
 break

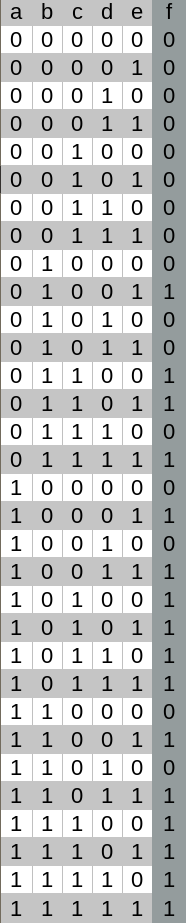
Результат: 195 ('в', 'д', 'п', 'р').

#### Пример 4

Создадим таблицу истинности и загрузим её в csv файл.

import itertools  
import csv  
  
variables = 'a b c d e'  
func = 'not(a + b) \* (c + e) <= d \* b'  
  
with open("test.csv", 'w', newline='\n') as csvfile:  
 writer = csv.writer(csvfile, delimiter=',')  
 writer.writerow(variables.replace(' ', '') + 'f')  
  
 for row in itertools.product('01', repeat=len(variables.replace(' ', '')):  
 exec(f'{variables.replace(" ", ", ")} = map(int, row)')  
 writer.writerow([\*row] + ['1' if eval(func) else '0'])

Результат:



comb\_iters\_ex4