


Модуль itertools в Python, эффективные итераторы для циклов



✓ mrqz.me

РЕКЛАМА

Бесплатная стратегия продвижения от сервиса Rookee

Получить предложение

[Стандартная библиотека Python3.](#) / Модуль itertools в Python, эффективные итераторы для циклов

Готовые итераторы на все случаи для эффективных циклов

[Модуль itertools](#) реализует ряд [итераторов](#), основанных на конструкциях из языков программирования APL, Haskell и SML. Каждый был переделан в форму, подходящую для Python.

Модуль стандартизирует основной набор быстрых и эффективных по памяти инструментов, которые полезны сами по себе или в сочетании. Вместе они образуют "алгебру итераторов", позволяющую быстро и эффективно создавать специализированные инструменты в чистом Python.

Например язык SML предоставляет инструмент табулирования: `tabulate(f)`, который генерирует последовательность `f(0)`, `f(1)`, Тот же эффект может быть достигнут в Python путем объединения функций [map\(\)](#) и [itertools.count\(\)](#) в форму `map(f, count())`.

Эти инструменты и их встроенные аналоги также хорошо работают с высокоскоростными функциями в модуле [operator](#). Например, оператор умножения может отображаться на два вектора для формирования эффективного точечного произведения: `sum(map(operator.mul, vector1, vector2))`.

Также читайте "[Введение в модуль itertools Python.](#)"

Бесконечные итераторы:

Итератор	Аргументы	Результат	Пример
<code>count()</code>	<code>start, [step]</code>	<code>start, start+step, start+2*step, ...</code>	<code>count(10) --> 10 11 12 13 14 ...</code>
<code>cycle()</code>	<code>p</code>	<code>p0, p1, ... plast, p0, p1, ...</code>	<code>cycle('ABCD') --> A B C D A B C D ...</code>
<code>repeat()</code>	<code>elem [,n]</code>	<code>elem, elem, elem, ... endlessly or up to n times</code>	<code>repeat(10, 3) --> 10 10 10</code>

Итераторы, оканчивающиеся на самой короткой входной последовательности:

Итератор	Аргументы	Результат	Пример
<code>accumulate()</code>	<code>p [,func]</code>	<code>p0, p0+p1, p0+p1+p2, ...</code>	<code>accumulate([1,2,3,4,5]) --> 1 3 6 10 15</code>
<code>chain()</code>	<code>p, q, ...</code>	<code>p0, p1, ... plast, q0, q1, ...</code>	<code>chain('ABC', 'DEF') --> A B C D E F</code>
<code>chain.from_iterable()</code>	<code>iterable</code>	<code>p0, p1, ... plast, q0, q1, ...</code>	<code>chain.from_iterable(['ABC', 'DEF']) --> A B C D E F</code>
<code>compress()</code>	<code>data, selectors</code>	<code>(d[0] if s[0]), (d[1] if s[1]), ...</code>	<code>compress('ABCDEF', [1,0,1,0,1,1]) --> A C E F</code>
<code>dropwhile()</code>	<code>pred, seq</code>	<code>seq[n], seq[n+1], starting when pred fails</code>	<code>dropwhile(lambda x: x<5, [1,4,6,4,1]) --> 6 4 1</code>
<code>filterfalse()</code>	<code>pred, seq</code>	<code>elements of seq where pred(elem) is false</code>	<code>filterfalse(lambda x: x%2, range(10)) --> 0 2 4 6 8</code>
<code>groupby()</code>	<code>iterable[, key]</code>	<code>sub-iterators grouped by value of key(v)</code>	
<code>islice()</code>	<code>seq, [start,] stop [, step]</code>	<code>elements from seq[start:stop:step]</code>	<code>islice('ABCDEFGH', 2, None) --> C D E F G</code>

starmap()	func, seq	func(*seq[0]), func(*seq[1]), ...	starmap(pow, [(2,5), (3,2), (10,3)]) --> 32 9 1000
takewhile()	pred, seq	seq[0], seq[1], until pred fails	takewhile(lambda x: x<5, [1,4,6,4,1]) --> 1 4
tee()	it, n	it1, it2, ... itn splits one iterator into n	
zip_longest()	p, q, ...	(p[0], q[0]), (p[1], q[1]), ...	zip_longest('ABCD', 'xy', fillvalue='-') --> Ax By C- D-

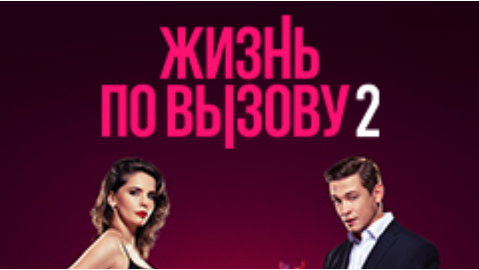
Комбинаторные итераторы:

Iterator	Arguments	Results
product()	p, q, ... [repeat=1]	Декартово произведение, эквивалентное вложенному циклу for
permutations()	p[, r]	Кортежи r-длины, все возможные упорядочения, без повторяющихся элементов
combinations()	p, r	Кортежи r-длины, в отсортированном порядке, без повторяющихся элементов
combinations_with_replacement()	p, r	Кортежи r-длины, в отсортированном порядке, с повторяющимися элементами

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Обзор модуля itertools Python](#)
- [Функция accumulate\(\) модуля itertools](#)
- [Функция chain\(\) модуля itertools](#)
- [Функция chain.from_iterable\(\) модуля itertools](#)
- [Функция combinations\(\) модуля itertools](#)
- [Функция combinations_with_replacement\(\) модуля itertools](#)
- [Функция compress\(\) модуля itertools](#)
- [Функция count\(\) модуля itertools](#)
- [Функция cycle\(\) модуля itertools](#)
- [Функция dropwhile\(\) модуля itertools](#)
- [Функция filterfalse\(\) модуля itertools](#)
- [Функция groupby\(\) модуля itertools](#)
- [Функция islice\(\) модуля itertools](#)
- [Функция permutations\(\) модуля itertools](#)
- [Функция product\(\) модуля itertools](#)
- [Функция repeat\(\) модуля itertools](#)
- [Функция starmap\(\) модуля itertools](#)
- [Функция takewhile\(\) модуля itertools](#)
- [Функция tee\(\) модуля itertools](#)
- [Функция zip_longest\(\) модуля itertools](#)
- [Трюки и рецепты использования модуля itertools](#)

ХОЧУ ПОМОЧЬ
ПРОЕКТУ



[DOCS-Python.ru](#)[™], 2023 г.

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs_python_ru](#)