

Массивы — коротко (англ. Generators and List Comprehension)

Генераторы

Если коротко, генераторы это короткая запись, с помощью которой можно получить один итерируемый объект из другого (массив из строки, кортеж из массива из т.п.).

Это может быть применение какой-то операции или функции к каждому элементу массива. Это может быть отбор элементов массива по какому-то критерию. Оба этих подхода можно совмещать.

Вот глава стандартной документации, посвящённая этой теме: [ссылка](#).

Несколько примеров, в которых используются только массивы:

Сделать массив квадратов из массива чисел:

```
x = [2, 3, 5, 7]
y = [elem ** 2 for elem in x]
```

Сделать массив из перевёрнутых задом наперёд слов в прочитанной строке:

```
x = [elem[::-1] for elem in input().split()]
```

Сделать массив из удвоенных слов прочитанной строки:

```
x = [elem * 2 for elem in input().split()]
```

Вот несколько примеров получения массива отбором элементов:

Сделать массив из чётных чисел:

```
x = [3, 4, 3, 45, 6, -65, 2, 34, 33]
y = [elem for elem in x if elem % 2 == 0]
```

Отобразить из строки только слова короче 5 символов и сохранить в массиве:

```
x = [elem for elem in input().split() if len(elem) < 5]
```

Вот пример, где используются оба подхода:

Сделать массив из всех слов длиннее 10 символов, обрезав их многоточием и добавив исходную длину в скобках:

```
x = [elem[:10] + '(' + str(len(elem)) + ')' for elem in input().split() if len(elem) > 10]
```

Если будет введена строка 'Абракадабра это длинноватое слово для пустопорожней беседы', то результат в массиве `x` будет таким:

```
['Абракадабр(11) ', 'длинновато(11)', 'пустопорож(13) ']
```

Срезы

С массивами, так же как и со строками, можно делать срезы. А именно:

`A[i:j]` — срез из $j - i$ элементов `A[i]`, `A[i+1]`, ..., `A[j-1]`.

`A[i:j:k]` — срез с шагом k : `A[i]`, `A[i+k]`, `A[i+2*k]`, ... Если $k < 0$, то элементы следуют в противоположном порядке (от большего индекса к меньшему).

`A[i:j:-1]` — срез из $i - j$ элементов `A[i]`, `A[i-1]`, ..., `A[j+1]` (то есть меняется порядок элементов).

Каждое из чисел i или j может отсутствовать, что означает “начало массива” или “конец массива”.

Массивы, в отличие от строк, являются изменяемыми объектами: можно отдельному элементу списка присвоить новое значение. Но можно менять и целиком срезы. Например:

```
A = [1, 2, 3, 4, 5]
A[2:4] = [7, 8, 9]
```

Получится список, у которого вместо двух элементов среза `A[2:4]` вставлен новый список уже из трёх элементов. Теперь список стал таким `[1, 2, 7, 8, 9, 5]`.

```
A = [1, 2, 3, 4, 5, 6, 7]
A[::-2] = [10, 20, 30, 40]
```

Получится список `[40, 2, 30, 4, 20, 6, 10]`. Здесь `A[::-2]` — это список из элементов `A[-1]`, `A[-3]`, `A[-5]`, `A[-7]`, которым присваиваются значения 10, 20, 30, 40 соответственно.

Если не непрерывному срезу (то есть срезу с шагом k , отличному от 1), присвоить новое значение, то количество элементов в старом и новом срезе обязательно должно совпадать, в противном случае произойдет ошибка

`ValueError`.

Обратите внимание, `A[i]` — это элемент списка, а не срез!

Некоторые операции и методы работы с массивами:

- `x in A`
Проверить, содержится ли элемент в списке. Возвращает `True` или `False`
- `x not in A`
То же самое, что `not(x in A)`
- `min(A)`
Наименьший элемент списка
- `max(A)`
Наибольший элемент списка
- `A.index(x)`
Индекс первого вхождения элемента `x` в список, при его отсутствии генерирует исключение `ValueError`
- `A.count(x)`
Количество вхождений элемента `x` в список
- `A.append(x)`
Добавить в конец списка `A` элемент `x`.
- `A.insert(i, x)`
Вставить в список `A` элемент `x` на позицию с индексом `i`. Элементы списка `A`, которые до вставки имели индексы `i` и больше сдвигаются вправо.
- `A.extend(B)`
Добавить в конец списка `A` содержимое списка `B`.
- `A.pop()`
Удалить из списка последний элемент, возвращается значение удалённого элемента
- `A.pop(i)`
Удалить из списка элемент с индексом `i`, возвращается значение удаленного элемента. Все элементы, стоящие правее удалённого, сдвигаются влево.

В этом листке встречаются некоторые задачи предыдущего листка (массивы). Но если в предыдущем листке задачи необходимо было решать без использования срезов, дополнительных списков, методов списков, то в этом листке, напротив, надо обойтись встроенными средствами работы с массивами и **во всех задачах нельзя использовать циклы, кроме задач H, I**).

Для многих упражнений написано, какое наибольшее число строк может быть в программе.

Если программа решается в одну строку, то необходимо использовать функции внутри функций. Например, вот так можно вычислить сумму всех чисел, введенных в строку, используя стандартную функцию `sum`:

```
print(sum(map(int, input().split())))
```

Обратите внимание, в однострочном решении нельзя сохранять список в переменной — нужно сразу же его обработать и вывести результат.

Решение в две строки, как правило, должно иметь следующий вид:

```
A = input().split()
print(' '.join(...))
```

При этом зачастую не требуется преобразовывать элементы списка к типу `int`.

Решение в три строки, как правило, должно иметь следующий вид:

```
A = input().split()
A = ...
print(' '.join(...))
```

A. *Чётные индексы*

Выведите все элементы списка с чётными индексами (то есть $A[0]$, $A[2]$, $A[4]$, ...).

Решите эту задачу в одну строку.

Input	Output
1 2 3 4 5	1 3 5

B. *Чётные элементы*

Выведите все чётные элементы списка.

Решите эту задачу в одну строку.

Input	Output
1 2 2 3 3 3 4	2 2 4

C. *Количество положительных элементов списка*

Выведите количество положительных элементов списка.

Решите эту задачу в одну строку.

Input	Output
1 -2 3 -4 5	3

D. *Наибольший элемент*

Дан список чисел. Выведите значение наибольшего элемента в списке, а затем индекс этого элемента в списке. Если наибольших элементов несколько, выведите индекс первого из них.

Решите эту задачу в две строки.

Input	Output
1 2 3 2 1	3 2

E. *Вывести в обратном порядке*

Выведите элементы данного списка в обратном порядке, не изменяя сам список.

Решите эту задачу в одну строку.

Input	Output
1 2 3 4 5	5 4 3 2 1

F. *Переставить соседние*

Переставьте соседние элементы списка ($A[0]$ с $A[1]$, $A[2]$ с $A[3]$ и т.д.). Если элементов нечётное число, то последний элемент остаётся на своём месте.

Решите эту задачу в три строки.

Input	Output
1 2 3 4 5	2 1 4 3 5

G. *Циклический сдвиг вправо*

Циклически сдвиньте элементы списка вправо ($A[0]$ переходит на место $A[1]$, $A[1]$ на место $A[2]$, и так далее, последний элемент переходит на место $A[0]$).

Решите эту задачу в две строки.

Input	Output
1 2 3 4 5	5 1 2 3 4

Н. Кузнечики

N кузнечиков стоят в ряд. Для каждого кузнечика задана числовая характеристика — длина его прыжка. Если длина прыжка кузнечика равна L , то он за один прыжок перепрыгивает через L других кузнечиков.

Каждую секунду последний кузнечик прыгает в направлении начала ряда, перепрыгивая через столько кузнечиков, чему равна длина его прыжка, и становится между двумя другими кузнечиками или в начало ряда.

В первой строке входных данных задана расстановка кузнечиков (длины их прыжков). Во второй строке входных данных задано число секунд T . Определите и выведите на экран расстановку кузнечиков через T секунд. Все длины прыжков — натуральные числа, меньшие, чем число кузнечиков в ряду.

Решите эту задачу в четыре строки.

Первая строка — считывание списка. Вторая строка — цикл `for` и считывание числа повторений. Третья строка — модификация списка в цикле. Четвертая строка — вывод результата.

Input	Output
1 2 3 4 2	4 1 2 2 3
2	

I. Числа k -боначчи

Назовём последовательность чисел последовательностью k -боначчи, если каждый элемент этой последовательности является суммой k предыдущих членов последовательности. В частности, последовательность 2-боначчи является последовательностью Фибоначчи.

Более формально, i -й элемент последовательности ϕ_i равен 1, если $0 \leq i \leq k - 1$ и равен сумме k предыдущих членов последовательности $\phi_{i-1} + \phi_{i-2} + \dots + \phi_{i-k}$ при $i \geq k$.

Даны два числа k и n ($k \geq 2, n \geq 0$). Вычислите n -й член последовательности k -боначчи ϕ_n .

Решите эту задачу в пять строк.

Первая строка — считывание данных: `k, n = map(int, input().split())`. Вторая строка — создание списка. Третья строка — цикл `for`. Четвертая строка — добавление нового элемента в список. Пятая строка — вывод результата. Для суммирования среза списка используйте функцию `sum`.

Input	Output
3 6	17
100 0	1