Как использовать функцию reshape() библиотеки NumPy в Python

**ОПУБЛИКОВАНО**[**22.05.2021**](https://pythonist.ru/kak-ispolzovat-funkcziyu-reshape-biblioteki-numpy-v-python/)[**ТОП**](https://pythonist.ru/category/topnews/)

Библиотека [NumPy](https://numpy.org/" \t "_blank) имеет множество различных функций для работы с многомерными массивами. [Функция reshape()](https://numpy.org/doc/stable/reference/generated/numpy.reshape.html) – одна из них. Она используется для изменения формы любого существующего массива без изменения данных, которые в нем находятся.

Форма определяется общим количеством элементов в каждом измерении. С помощью функции reshape() можно добавлять или удалять измерения в массиве, а также изменять количество элементов в каждом измерении. Например, можно преобразовать одномерный массив в многомерный (но не наоборот!).

В этой статье мы расскажем, как использовать функцию reshape() и каков принцип ее работы.

**Синтаксис**

Давайте начнем с синтаксиса функции reshape(). Вот так он выглядит в общем виде:

numpy.reshape(np\_array, new\_shape, order='C')

Эта функция может принимать три аргумента. Первый и второй аргументы являются обязательными, а третий — опциональный.

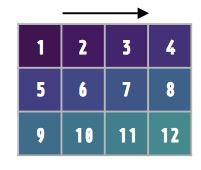
Исходный массив NumPy, форму которого мы хотим изменить, – это значение первого аргумента (np\_array).

Форма массива устанавливается во втором аргументе (new\_shape). Его значение может быть целым числом или кортежем целых чисел.

Значение третьего аргумента определяет порядок заполнения массива и переноса элементов в преобразованном массиве. Возможных значений три: «C», «F» или «A». Давайте разберем, что значит каждый из этих вариантов.

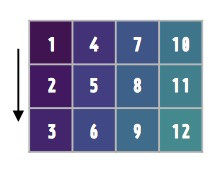
**order=’C’**

Упорядочивание индексов в стиле языка C. Индекс последней оси изменяется быстрее, а индекс первой — медленнее.

Упорядочение в стиле C. [Источник](https://towardsdatascience.com/reshaping-numpy-arrays-in-python-a-step-by-step-pictorial-tutorial-aed5f471cf0b)

**order=’F’**

Упорядочивания индексов в стиле языка Фортран. Индекс первой оси изменяется быстрее, а индекс последней — медленнее.

Упорядочение в стиле Fortran

**order=’A’**

Варианты «C» и «F» не учитывают макет памяти основного массива. Они относятся лишь к порядку индексации. Порядок «A» означает чтение и запись элементов в стиле Фортран, если исходный массив в памяти тоже в стиле Фортран. В противном случае применяется C-подобный стиль.

**Использование функции reshape()**

Первым делом, прежде чем попрактиковаться в использовании данной функции, вам следует импортировать библиотеку NumPy. После этого можно приступать к работе.

Далее мы покажем различные варианты использования функции reshape().

**Пример 1: преобразование одномерного массива в двумерный**

Итак, давайте разберем, как с помощью функции reshape() преобразовать одномерный массив в двумерный.

В этом сценарии для создания одномерного массива из 10 элементов используется функция arange().

Первая функция reshape() используется для преобразования одномерного массива в двумерный, состоящий из 2 строк и 5 столбцов. Здесь функция reshape() вызывается с использованием имени модуля np.

А вторая функция reshape() используется для преобразования одномерного массива в двумерный, состоящий из 5 строк и 2 столбцов. Здесь уже функция reshape() вызывается с использованием массива NumPy с именем np\_array.

**import** numpy **as** np

np\_array = np.arange(10)

print("Исходный массив : \n", np\_array)

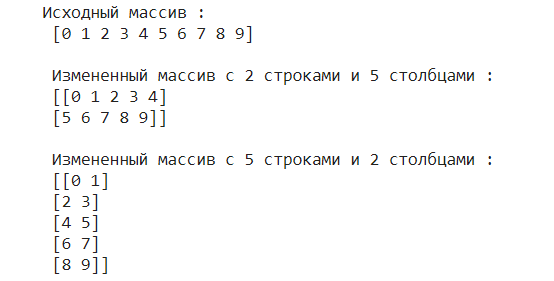
new\_array = np.reshape(np\_array, (2, 5))

print("\n Измененный массив с 2 строками и 5 столбцами : \n", new\_array)

new\_array = np\_array.reshape(5, 2)

print("\n Измененный массив с 5 строками и 2 столбцами : \n", new\_array)

Если вы запустите описанную выше программу, то получите результат, как на следующем скриншоте. Первый print() показывает исходный массив, а второй и третий выводят преобразованные массивы.



**Пример 2: преобразование одномерного массива в трехмерный**

Теперь давайте посмотрим, как при помощи функции reshape() преобразовать одномерный массив в трехмерный.

Воспользуемся функцией array() для создания одномерного массива из 12 элементов.

Функция reshape() преобразует созданный одномерный массив в трехмерный размером 2х2х3. Здесь функция reshape() вызывается с использованием NumPy-массива np\_array.

**import** numpy **as** np

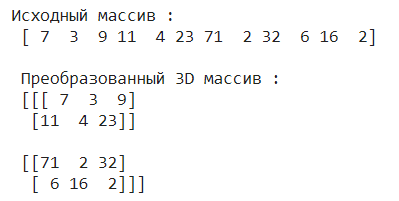
np\_array = np.array([7, 3, 9, 11, 4, 23, 71, 2, 32, 6, 16, 2])

print("Исходный массив : \n", np\_array)

new\_array = np\_array.reshape(2, 2, 3)

print("\n Преобразованный 3D массив : \n", new\_array)

Выполнив данный код, вы получите следующий вывод. Как и в прошлый раз, первый print() показывает изначальный массив, второй – преобразованный массив.



**Пример 3: изменение формы массива NumPy с учетом порядка**

Как вы помните, у функции reshape() есть третий — опциональный — аргумент, задающий порядок индексации. Давайте посмотрим, как он применяется на практике.

Как и в первом примере, воспользуемся функцией arange()для создания одномерного массива из 15 элементов.

Первая функция reshape() используется для создания двумерного массива из 3 строк и 5 столбцов с упорядочением в стиле C. В то время как вторая функция reshape() используется для создания двумерного массива из 3 строк и 5 столбцов с упорядочением в стиле Фортрана.

**import** numpy **as** np

np\_array = np.arange(15)

print("Исходный массив : \n", np\_array)

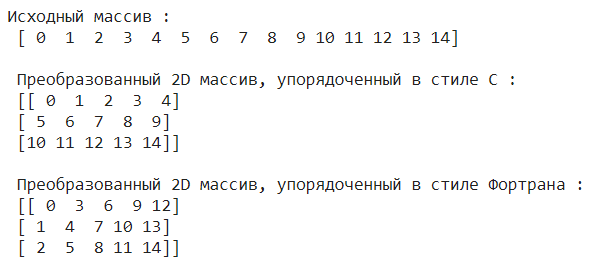
new\_array1 = np.reshape(np\_array, (3, 5), order='C')

print("\n Преобразованный 2D массив, упорядоченный в стиле С : \n", new\_array1)

new\_array2 = np.reshape(np\_array, (3, 5), order='F')

print("\n Преобразованный 2D массив, упорядоченный в стиле Фортрана : \n", new\_array2)

Давайте выполним наш код. Вот, что мы получим. Как и раньше, первый print() показывает исходный массив значений. Второй print() показывает значения массива, упорядоченного по строкам. Третий – упорядоченного по столбцам.



**Итоги**

Итак, в этой статье мы подробно описали, как использовать функцию reshape() для преобразования массива из одной формы в другую. Цели и особенности использования данной функции становятся достаточно прозрачными после изучение примеров, которые мы привели выше.

Возможно, вам будет интересно также почитать [о сравнении функций numpy.reshape() и numpy.flatten()](https://pythonist.ru/funkczii-numpy-reshape-i-numpy-flatten/).