Python TUTORIAL

# NumPy flatten()

**Summary**: in this tutorial, you'll learn how to use the numpy `flatten()` method to return a copy of an array collapsed into one dimension.

## Introduction to the NumPy flatten() method

The `flatten()` is a method of the `ndarray` (https://www.pythontutorial.net/python-numpy/create-numpy-array/) class. The `flatten()` method returns a copy of an array (https://www.pythontutorial.net/python-numpy/numpy-copy/) collapsed into one dimension.

The following shows the syntax of the `flatten()` method:

```
ndarray.flatten(order='C')
```

The order parameter specifies the order of elements of an array in the returned array. It accepts one of the following values:

- 'C' means to flatten array elements into row-major order (C-style).
- 'F' means to flatten array elements into column-major order (Fortran-style).
- 'A' – means to flatten array elements in column-major order if a is Fortran contiguous in memory or row-major otherwise.

- 'K' means to flatten array elements in order of the elements laid out in memory.

By default, the order is 'C' which flattens the array elements into row-major.

## NumPy flatten() method examples

Let's take some examples of using the NumPy `flatten()` method.

### 1) Using flatten() method example with a multidimensional array

The following example uses the `flatten()` method to return a 1-D array from a 2-D array:

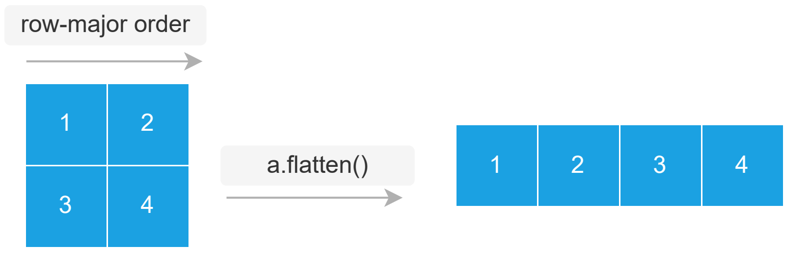
```
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = a.flatten()
print(b)
```

Output:

```
[1 2 3 4]
```

How it works.



First, create a 2-D array that has two rows and two columns:

```
a = np.array([[1, 2], [3, 4]])
```

Second, return a copy of the array with dimensions collapsed into one using the `flatten()` method:

```
b = a.flatten()
```

Third, display the result array:

```
print(b)
```

Note that b is a copy, not a view of the array a. If you change elements in array b, the elements in array a are not changed. For example:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = a.flatten()

# change element at index 0
b[0] = 0
print(b)

# display the array a
print(a)
```

Output:

```
[0 2 3 4]
[[1 2]
 [3 4]]
```

In this example:

First, flatten the array a and assign the result array to b variable:

```
b = a.flatten()
```

Second, change the element at index 0 of b to zero and print out b:

```
b[0] = 0
print(b)
```

Third, display the array a:

```
print(a)
```

The output shows that the element at index 0 of b changes but the element at index 0 of a doesn't change.

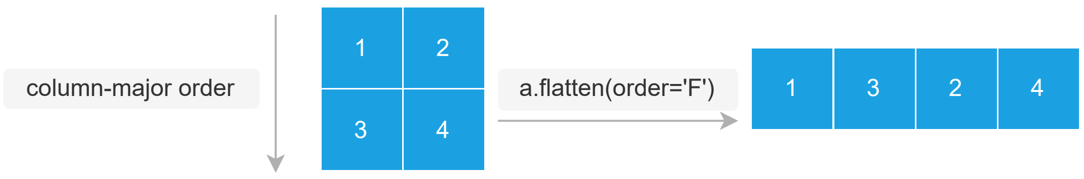
## 2) Using numpy flatten() method to flatten an array using column-major order

The following example uses the numpy `flatten()` method to flatten an array using column-major order:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = a.flatten(order='F')

print(b)
```



Output:

```
[1 3 2 4]
```

## Summary

- Use the numpy array `flatten()` method to return a copy of an array collapsed into one dimension.