numpy.reshape

numpy.**reshape**(*a*, *newshape*, *order='C'*) [source] (http://github.com/numpy/numpy/blob/v1.10.1/numpy/core/fromnumeric.py#L128-L225)

Gives a new shape to an array without changing its data.

Parameters: **a** : array_like

Array to be reshaped.

newshape : int or tuple of ints

The new shape should be compatible with the original shape. If an integer, then the result will be a 1-D array of that length. One shape dimension can be -1. In this case, the value is inferred from the length of the array and remaining dimensions.

如果某一维度是-1，说明这个维度是随动调整

vt. 推断; 猜想，推理; 暗示; 意指;
vi. 作出推论;

order : {'C', 'F', 'A'}, optional

Read the elements of a using this index order, and place the elements into the reshaped array using this index order. 'C' means to read / write the elements using C-like index order, with the last axis index changing fastest, back to the first axis index changing slowest. 'F' means to read / write the elements using Fortran-like index order, with the first index changing fastest, and the last index changing slowest. Note that the 'C' and 'F' options take no account of the memory layout of the underlying array, and only refer to the order of indexing. 'A' means to read / write the elements in Fortran-like index order if a is Fortran contiguous in memory, C-like order otherwise.

Returns: **reshaped_array** : ndarray

This will be a new view object if possible; otherwise, it will be a copy. Note there is no guarantee of the memory layout (C- or Fortran- contiguous) of the returned array.

See also:

ndarray.reshape (numpy.ndarray.reshape.html#numpy.ndarray.reshape)
Equivalent method.

Previous topic

numpy.flipud (numpy.flipud.html)

Next topic

numpy.roll (numpy.roll.html)

Notes

It is not always possible to change the shape of an array without copying the data. If you want an error to be raise if the data is copied, you should assign the new shape to the shape attribute of the array:

```
>>> a = np.zeros((10, 2))
# A transpose make the array non-contiguous
>>> b = a.T
# Taking a view makes it possible to modify the shape without
modifying
# the initial object.
>>> c = b.view()
>>> c.shape = (20)
AttributeError: incompatible shape for a non-contiguous array
```

```
a
array([[ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.]])
```

The order keyword gives the index ordering both for fetching the values from a, and then placing the values into the output array. For example, let's say you have an array:

```
>>> b=a.T  #转置！！！
>>> b
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

```
>>> c=b.view()
>>> c
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

```
>>> a = np.arange(6).reshape((3, 2))
>>> a
array([[0, 1],
       [2, 3],
       [4, 5]])
```

You can think of reshaping as first raveling the array (using the given index order), then inserting the elements from the raveled array into the new array using the same kind of index ordering as was used for the raveling.

使纠缠，变得错综复杂，这里面是"遍历"的意思？

```
>>> np.reshape(a, (2, 3)) # C-like index ordering
array([[0, 1, 2],
       [3, 4, 5]])
>>> np.reshape(np.ravel(a), (2, 3)) # equivalent to C ravel th
en C reshape
array([[0, 1, 2],
       [3, 4, 5]])
>>> np.reshape(a, (2, 3), order='F') # Fortran-like index orde
ring
array([[0, 4, 3],
       [2, 1, 5]])
>>> np.reshape(np.ravel(a, order='F'), (2, 3), order='F')
array([[0, 4, 3],
       [2, 1, 5]])
```

Examples

```
>>> a = np.array([[1,2,3], [4,5,6]])
>>> np.reshape(a, 6)
array([1, 2, 3, 4, 5, 6])
>>> np.reshape(a, 6, order='F')
array([1, 4, 2, 5, 3, 6])
```

```
>>> np.reshape(a, (3,-1))       # the unspecified value is inf
erred to be 2
array([[1, 2],
       [3, 4],
       [5, 6]])
```