Day_32_271123

January 23, 2024

[1]:
```
import numpy as np
```

1 Shallow Copy & Deep Copy

[2]:
```
a = np.arange(1,6)
a
```

[2]:
```
array([1, 2, 3, 4, 5])
```

- Stride is nothing but step size in array

Shallow Copy - The other variable shares the memory address of a variable

[3]:
```
b = a
```

[4]:
```
b
```

[4]:
```
array([1, 2, 3, 4, 5])
```

- Here assigning a to b is a shallow Copy
- But when we made any changes to original array it will create a new adress

[5]:
```
x = a+1
```

[6]:
```
x
```

[6]:
```
array([2, 3, 4, 5, 6])
```

[7]:
```
b = a[::2]
```

[8]:
```
b
```

[8]:
```
array([1, 3, 5])
```

- Here it is shallow copy, The changes will be made in header
- Initially (a) ### Metadata of a | Headers | | |———|———| | Shape | (5,) | | ndim | 1 | | Size | 5 | | Stride | 1 |
- After assigning (b) ### Metadata of b | Headers | | |———|———| | Shape | (5,) | | ndim | 1 | | Size | 5 | | Stride | 2 |

[9]:
```
a = np.array([1,2,3,4,5])
b = a
```

2 Function to check whether Memory Sharing happens or not

[10]:
```
np.shares_memory(a,b)
```

[10]:
```
True
```

[11]:
```
b = a+1
np.shares_memory(a,b)
```

[11]:
```
False
```

3 Splitting in 1D

[12]:
```
a = np.arange(9)
print(a)
b = np.split(a,3) # Equal Splits
```

```
[0 1 2 3 4 5 6 7 8]
```

[13]:
```
b
```

[13]:
```
[array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8])]
```

[14]:
```
b = np.split(a,(2,4,7)) # Split using Index
b
```

[14]:
```
[array([0, 1]), array([2, 3]), array([4, 5, 6]), array([7, 8])]
```

4 Splitting in 2D

[15]:
```
x = np.arange(1,17).reshape(4,4)
x
```

[15]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

[16]:
```
np.split(x,2) #Divides into equal rows defaulty with axis = 0
```

[16]:
```
[array([[1, 2, 3, 4],
        [5, 6, 7, 8]]),
 array([[ 9, 10, 11, 12],
        [13, 14, 15, 16]])]
```

```
[17]: np.hsplit(x,2) #Horizontal split
```

```
[17]: [array([[ 1,  2],
              [ 5,  6],
              [ 9, 10],
              [13, 14]]),
       array([[ 3,  4],
              [ 7,  8],
              [11, 12],
              [15, 16]])]
```

```
[18]: np.vsplit(x,2) #Vertical Split
```

```
[18]: [array([[1, 2, 3, 4],
              [5, 6, 7, 8]]),
       array([[ 9, 10, 11, 12],
              [13, 14, 15, 16]])]
```

5 Stacking

```
[19]: a = np.arange(10)
```

```
[20]: np.vstack((a,a,a))
```

```
[20]: array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
             [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
             [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

```
[21]: np.hstack((a,a,a))
```

```
[21]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1,
             2, 3, 4, 5, 6, 7, 8, 9])
```

```
[22]: z = np.arange(9).reshape(3,3)
      z
```

```
[22]: array([[0, 1, 2],
             [3, 4, 5],
             [6, 7, 8]])
```

```
[23]: np.vstack((z,z,z))
```

```
[23]: array([[0, 1, 2],
             [3, 4, 5],
             [6, 7, 8],
        [0, 1, 2],
        [3, 4, 5],
        [6, 7, 8],
        [0, 1, 2],
        [3, 4, 5],
        [6, 7, 8]])
```

```
[24]: np.hstack((z,z,z))
```

```
[24]: array([[0, 1, 2, 0, 1, 2, 0, 1, 2],
             [3, 4, 5, 3, 4, 5, 3, 4, 5],
             [6, 7, 8, 6, 7, 8, 6, 7, 8]])
```

```
[25]: np.concatenate((z,z,z),axis=0) #Stacking using concatenate function
```

```
[25]: array([[0, 1, 2],
             [3, 4, 5],
             [6, 7, 8],
             [0, 1, 2],
             [3, 4, 5],
             [6, 7, 8],
             [0, 1, 2],
             [3, 4, 5],
             [6, 7, 8]])
```

```
[26]: g = np.arange(0,4)
      g
```

```
[26]: array([0, 1, 2, 3])
```

```
[27]: g = np.vstack(g)
      g
```

```
[27]: array([[0],
             [1],
             [2],
             [3]])
```

```
[28]: g = np.concatenate((g,g,g),axis=1)
      g
```

```
[28]: array([[0, 0, 0],
             [1, 1, 1],
             [2, 2, 2],
             [3, 3, 3]])
```

6 Broadcasting

[29]:
```
a = np.arange(0,40,10)
a
```

[29]:
```
array([ 0, 10, 20, 30])
```

[30]:
```
a = np.vstack((a,a,a))
```

[31]:
```
np.hstack((a,a,a))
```

[31]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30]])
```

[32]:
```
np.concatenate((a,a),axis=1)
```

[32]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

[33]:
```
np.tile(a,(3,3))
```

[33]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30]])
```

[34]:
```
np.tile(a,(3,2))
```

[34]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

6.1 Case 1

```
[35]: a
```

```
[35]: array([[ 0, 10, 20, 30],
              [ 0, 10, 20, 30],
              [ 0, 10, 20, 30]])
```

```
[36]: v = np.vstack(a)
      v
```

```
[36]: array([[ 0, 10, 20, 30],
              [ 0, 10, 20, 30],
              [ 0, 10, 20, 30]])
```

```
[37]: j = np.concatenate((v,v),axis =1)
      j
```

```
[37]: array([[ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30]])
```

```
[38]: i = np.arange(0,3)
      i
```

```
[38]: array([0, 1, 2])
```

```
[39]: j+i
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[39], line 1
----> 1 j+i

ValueError: operands could not be broadcast together with shapes (3,12) (3,)
```

6.2 Case 2

```
[40]: j
```

```
[40]: array([[ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30]])
```

```
[41]: k = np.vstack((i,i,i,i))
      k
```

```
[41]: array([[0, 1, 2],
              [0, 1, 2],
              [0, 1, 2],
              [0, 1, 2]])
```

```
[42]: j+k
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[42], line 1
----> 1 j+k

ValueError: operands could not be broadcast together with shapes (3,12) (4,3)
```

6.3 Case 3

```
[43]: i
```

```
[43]: array([0, 1, 2])
```

```
[44]: v
```

```
[44]: array([[ 0, 10, 20, 30],
              [ 0, 10, 20, 30],
              [ 0, 10, 20, 30]])
```

```
[45]: o = np.concatenate((v,v,v),axis = 1)
      o
```

```
[45]: array([[ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30],
              [ 0, 10, 20, 30,  0, 10, 20, 30,  0, 10, 20, 30]])
```

```
[46]: p = np.vstack((i,i,i,i))
      p
```

```
[46]: array([[0, 1, 2],
              [0, 1, 2],
              [0, 1, 2],
              [0, 1, 2]])
```

```
[47]: p+o
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[47], line 1
----> 1 p+o
```

```
ValueError: operands could not be broadcast together with shapes (4,3) (3,12)
```

7 Rule 1 : If two array differ in the number of dimesions, the shape of one with fewer dimensions is padded with ones on its leading (Left side)

8 Rule 2 : If the shape of two arrays doesnot match in any dimensions, the array with shape equal to 1 is stretched to match the other shape.

[48]:
```
array = np.arange(16).reshape(4,4)
array
```

[48]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

[49]:
```
arr = np.array([0,1,2,3])
```

[50]:
```
array+arr
```

[50]:
```
array([[ 0,  2,  4,  6],
       [ 4,  6,  8, 10],
       [ 8, 10, 12, 14],
       [12, 14, 16, 18]])
```

[51]:
```
a = np.arange(6)
a
```

[51]:
```
array([0, 1, 2, 3, 4, 5])
```

[52]:
```
a.shape
```

[52]:
```
(6,)
```

9 Day 33 28-11-23

10 Other function to increase the dimensions

[58]:
```
b = np.expand_dims(a,axis=0)
```

[60]:
```
print(b)
b.shape
```

```
[[0 1 2 3 4 5]]
```

[60]:
```
(1, 6)
```

[61]:
```
c = a[np.newaxis,:]
c.shape
```

[61]:
```
(1, 6)
```

[62]:
```
c
```

[62]:
```
array([[0, 1, 2, 3, 4, 5]])
```

[63]:
```
d = a[:,np.newaxis]
d
```

[63]:
```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
```

[64]:
```
e = np.arange(6).reshape(2,3)
e
```

[64]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

[65]:
```
np.expand_dims(e,axis=0).shape
```

[65]:
```
(1, 2, 3)
```

[68]:
```
f = np.arange(6).reshape(2,3)
np.expand_dims(f,axis=2)
```

[68]:
```
array([[[0],
        [1],
        [2]],

       [[3],
        [4],
        [5]]])
```

11 Removing Dimensions

```
[78]: a = np.arange(5)
      b = np.expand_dims(a,axis=1)
      b
```

```
[78]: array([[0],
             [1],
             [2],
             [3],
             [4]])
```

```
[79]: np.squeeze(b,axis=1)
```

```
[79]: array([0, 1, 2, 3, 4])
```

```
[92]: k = np.arange(12).reshape(1,12)
      k
```

```
[92]: array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]])
```

```
[86]: np.squeeze(k).shape
```

```
[86]: (12,)
```

11.0.1 NumPy cannot remove the original dimension but can remove the fake dimension

```
[93]: np.squeeze(k,axis=1).shape
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[93], line 1
----> 1 np.squeeze(k,axis=1).shape

File C:\Data\env\Lib\site-packages\numpy\core\fromnumeric.py:1558, in squeeze(a
  ↪axis)
   1556     return squeeze()
   1557 else:
-> 1558     return squeeze(axis=axis)

ValueError: cannot select an axis to squeeze out which has size not equal to on
```

```
[94]: a = np.arange(12).reshape(12,1,1)
      a
```

```
[94]: array([[[ 0]],

             [[ 1]],

             [[ 2]],

             [[ 3]],

             [[ 4]],

             [[ 5]],

             [[ 6]],

             [[ 7]],

             [[ 8]],

             [[ 9]],

             [[10]],

             [[11]]])
```

```
[95]: np.squeeze(a,axis=-1)
```

```
[95]: array([[ 0],
             [ 1],
             [ 2],
             [ 3],
             [ 4],
             [ 5],
             [ 6],
             [ 7],
             [ 8],
             [ 9],
             [10],
             [11]])
```

```
[98]: np.squeeze(a,axis=-2)
```

```
[98]: array([[ 0],
             [ 1],
             [ 2],
             [ 3],
             [ 4],
             [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11]])
```

[99]:
```
array = np.arange(10)
```

[100]:
```
array2 = array.view
```

[102]:
```
np.shares_memory(array2,array)
```

[102]:
```
False
```

[101]:
```
array3 = array.copy
```

[103]:
```
np.shares_memory(array3,array)
```

[103]:
```
False
```

[ ]:
```

```