- **Shallow vs Deep Copy**
  - `view()`
  - `copy()`
  - `copy.deepcopy()`

- **Shallow vs Deep Copy**
  - `view()`
  - `copy()`
  - `copy.deepcopy()`

- **Dimension Expansion and Reduction**
  - `np.expand_dims()`
  - `np.newaxis`
  - `np.sqeeze()`
- **Array splitting and Merging**
  - Splitting arrays - `split()` , `hsplit()` , `vsplit()`
  - Merging Arrays - `hstack()` , `vstack()` , `concatenate`

- **Array splitting and Merging**
  - Splitting arrays - `split()` , `hsplit()` , `vsplit()`
  - Merging Arrays - `hstack()` , `vstack()` , `concatenate`

In [3]:
```
import numpy as np
a = np.arange(4)
```

In [4]:
```
a
```

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

In [5]:
```
b = a.reshape((2,2))
```

In [6]:
```
b
```

Out[6]:
```
array([[0, 1],
       [2, 3]])
```

In [7]:
```
a[0]=100 #Creates a shallow copy
```

In [8]:
```
a
```

Out[8]:
```
array([100,   1,   2,   3])
```

In [9]:
```
b
```

Out[9]:
```
array([[100,   1],
       [  2,   3]])
```

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

Out[10]:
```
True
```

In [11]:
```
a=np.arange(4)
a
```

Out[11]:
```
array([0, 1, 2, 3])
```

In [12]:
```
c=a+2
c
```

Out[12]:
```
array([2, 3, 4, 5])
```

In [13]:
```
a[0]=100
a
```

Out[13]:
```
array([100,   1,   2,   3])
```

In [14]:
```
c
```

Out[14]:
```
array([2, 3, 4, 5])
```

In [15]:
```
np.shares_memory(a,c)
```

Out[15]:
```
False
```

In [ ]:
```

```

In [17]:
```
a=np.arange(10)
a
```

Out[17]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [18]:
```
view_a=a.view() #shallow copy #recommended
view_a
```

Out[18]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [19]:
```
np.shares_memory(a,view_a)
```

Out[19]:
```
True
```

In [20]:
```
b=a.copy() #deep copy
b
```

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

In [21]:
```
np.shares_memory(a,b)
```

Out[21]:
```
False
```

In [22]:
```
a[0]=100
```

In [23]:
```
a
```

Out[23]:
```
array([100,   1,   2,   3,   4,   5,   6,   7,   8,   9])
```

In [24]:
```
b
```

Out[24]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [25]:
```
#2D
```

In [26]:
```
a=np.array([[1,2,3],[2,3,4]])
a
```

Out[26]:
```
array([[1, 2, 3],
       [2, 3, 4]])
```

In [27]:
```
b=a.copy()
```

In [28]:
```
c=np.array([1,"m",[1,2,3]])
```

```
/var/folders/bs/y1_q644n2msgp7741p7lt4480000gn/T/ipykernel_40233/1843714047.
py:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequ
ences (which is a list-or-tuple of lists-or-tuples-or ndarrays with differen
t lengths or shapes) is deprecated. If you meant to do this, you must specif
y 'dtype=object' when creating the ndarray.
  c=np.array([1,"m",[1,2,3]])
```

In [30]:
```
c=np.array([1,"m",[1,2,3]], dtype=object)
c
```

Out[30]:
```
array([1, 'm', list([1, 2, 3])], dtype=object)
```

```
In [31]: d=c.copy()
```

```
In [32]: c[2][0]=100
```

```
In [33]: c
```

```
Out[33]: array([1, 'm', list([100, 2, 3])], dtype=object)
```

```
In [34]: d
```

```
Out[34]: array([1, 'm', list([100, 2, 3])], dtype=object)
```

```
In [35]: np.shares_memory(c,d)
```

```
Out[35]: False
```

```
In [36]: import copy
```

```
In [37]: c=np.array([1,"m",[1,2,3]],dtype=object)
         c
```

```
Out[37]: array([1, 'm', list([1, 2, 3])], dtype=object)
```

```
In [38]: d=copy.deepcopy(c) #recommended for deep copy
         d
```

```
Out[38]: array([1, 'm', list([1, 2, 3])], dtype=object)
```

```
In [39]: c[2][0]=100
```

```
In [40]: c
```

```
Out[40]: array([1, 'm', list([100, 2, 3])], dtype=object)
```

```
In [41]: d
```

```
Out[41]: array([1, 'm', list([1, 2, 3])], dtype=object)
```

```
In [42]: np.shares_memory(c,d)
```

```
Out[42]: False
```

In [ ]:

```

```

In [ ]:

```
# Dimension Expansion and Reduction

# np.expand_dims()
# np.newaxis
# np.sqeeze()
```

In [43]:

```
a=np.arange(9)
a
```

Out[43]:

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

In [44]:

```
b=a.reshape((3,3))
b
```

Out[44]:

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

In [45]:

```
b.shape
```

Out[45]:

```
(3, 3)
```

In [46]:

```
c=a.reshape((3,3,1))
```

In [47]:

```
c
```

Out[47]:

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

       [[3],
        [4],
        [5]],

       [[6],
        [7],
        [8]]])
```

In [48]:

```
c.shape
```

Out[48]:

```
(3, 3, 1)
```

In [49]:

```
d=a.reshape((3,3,1,1,1,1,1,1,1))
```

In [50]:
```
d.shape
```

Out[50]:
```
(3, 3, 1, 1, 1, 1, 1, 1, 1)
```

In [51]:
```
d.ndim
```

Out[51]:
```
9
```

In [52]:
```
#Better way
```

In [53]:
```
a=np.arange(9)
print(a)
print(a.shape)
```

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

In [54]:
```
e=np.expand_dims(a,axis=0)
```

In [55]:
```
e
```

Out[55]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8]])
```

In [56]:
```
e.shape
```

Out[56]:
```
(1, 9)
```

In [57]:
```
e=np.expand_dims(a,axis=1)
e
```

Out[57]:
```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8]])
```

In [58]:
```
e.shape
```

Out[58]:
```
(9, 1)
```

In [59]:
```
a=np.arange(1,13).reshape((3,4))
print(a.shape)
```

```
(3, 4)
```

In [62]:
```
np.expand_dims(a,axis=2).shape
```

Out[62]:
```
(3, 4, 1)
```

In [65]:
```
a=np.arange(5)
a.shape
```

Out[65]:
```
(5,)
```

In [64]:
```
a[:,np.newaxis].shape
```

Out[64]:
```
(5, 1)
```

In [66]:
```
a[np.newaxis,:].shape
```

Out[66]:
```
(1, 5)
```

In [68]:
```
a[np.newaxis,np.newaxis,np.newaxis,:,np.newaxis,np.newaxis,np.newaxis].sh
```

Out[68]:
```
(1, 1, 1, 5, 1, 1, 1)
```

In [69]:
```
#Reduce the size using Squeeze
```

In [82]:
```
np.array(np.arange(10).reshape(2,5),ndmin=10).shape
```

Out[82]:
```
(1, 1, 1, 1, 1, 1, 1, 1, 2, 5)
```

In [71]:
```
a=np.arange(1,13).reshape((1,3,1,4,1))
a.shape
```

Out[71]:
```
(1, 3, 1, 4, 1)
```

In [72]:
```
np.squeeze(a).shape
```

Out[72]:
```
(3, 4)
```

In [73]:
```
a=np.arange(1,13).reshape((1,3,1,4,1,1,1,1,1,1,1,1,1,1))
a.ndim
```

Out[73]:
```
14
```

In [74]:
```
np.squeeze(a).shape
```

Out[74]:
```
(3, 4)
```

In [ ]:
```

```

In [ ]:
```
#Array Split and Merge
```

In [83]:
```
a=np.arange(1,13)
a
```

Out[83]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [86]:
```
np.split(a,5)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[86], line 1
----> 1 np.split(a,5)

File <__array_function__ internals>:180, in split(*args, **kwargs)

File ~/anaconda3/lib/python3.10/site-packages/numpy/lib/shape_base.py:872, i
n split(ary, indices_or_sections, axis)
    870     N = ary.shape[axis]
    871     if N % sections:
--> 872         raise ValueError(
    873             'array split does not result in an equal division') from
None
    874 return array_split(ary, indices_or_sections, axis)

ValueError: array split does not result in an equal division
```

Type *Markdown* and LaTeX: α^2

In [87]:
```
np.split(a,[4,5]) # [0:4, 4:5, 5:]
```

Out[87]:
```
[array([1, 2, 3, 4]), array([5]), array([ 6,  7,  8,  9, 10, 11, 12])]
```

In [ ]:
```

```

In [88]:

```
np.split(a,[4,5,7,5])
```

Out[88]:

```
[array([1, 2, 3, 4]),
 array([5]),
 array([6, 7]),
 array([], dtype=int64),
 array([ 6,  7,  8,  9, 10, 11, 12])]
```

In [89]:

```
a
```

Out[89]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [90]:

```
np.split(a,[-5,-2])
```

Out[90]:

```
[array([1, 2, 3, 4, 5, 6, 7]), array([ 8,  9, 10]), array([11, 12])]
```

In [ ]:

```

```

In [ ]:

```
#Splitting 2D array
```

In [91]:

```
a=np.arange(1,13).reshape((3,4))
a
```

Out[91]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [92]:

```
np.split(a,3)
```

Out[92]:

```
[array([[1, 2, 3, 4]]), array([[5, 6, 7, 8]]), array([[ 9, 10, 11, 12]])]
```

In [93]:

```
np.split(a,3,axis=0)
```

Out[93]:

```
[array([[1, 2, 3, 4]]), array([[5, 6, 7, 8]]), array([[ 9, 10, 11, 12]])]
```

In [95]:

```
np.split(a,2,axis=1)
```

Out[95]:

```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12]])]
```

In [96]:

```
np.vsplit(a,3)
```

Out[96]:

```
[array([[1, 2, 3, 4]]), array([[5, 6, 7, 8]]), array([[ 9, 10, 11, 12]])]
```

In [97]:

```
a
```

Out[97]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [98]:

```
np.hsplit(a,2)
```

Out[98]:

```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12]])]
```

In [ ]:

```
#Stacking - vstack , hstack
```

In [99]:

```
a=np.arange(1,5)
b=np.arange(1,5)
```

In [100]:

```
a
```

Out[100]:

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

In [101]:

```
b
```

Out[101]:

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

In [102]:

```
np.hstack((a,b))
```

Out[102]:

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

In [103]:

```
np.vstack((a,b))
```

Out[103]:

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

In [104]:

```
np.vstack((a,b)).shape
```

Out[104]:

```
(2, 4)
```

In [105]:

```
#2D examples
```

In [106]:

```
a=np.arange(1,13).reshape((3,4))
b=np.arange(1,13).reshape((3,4))
```

In [107]:

```
print(a)
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [108]:

```
np.hstack((a,b))
```

Out[108]:

```
array([[ 1,  2,  3,  4,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  5,  6,  7,  8],
       [ 9, 10, 11, 12,  9, 10, 11, 12]])
```

In [109]:

```
np.vstack((a,b))
```

Out[109]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [110]:

```
a=np.arange(1,13).reshape((3,4))
b=np.arange(1,5).reshape((1,4))
print(a)
print(b)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[1 2 3 4]]
```

In [111]:
```
np.hstack((a,b))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[111], line 1
----> 1 np.hstack((a,b))

File <__array_function__ internals>:180, in hstack(*args, **kwargs)

File ~/anaconda3/lib/python3.10/site-packages/numpy/core/shape_base.py:345,
in hstack(tup)
    343     return _nx.concatenate(arrs, 0)
    344 else:
--> 345     return _nx.concatenate(arrs, 1)

File <__array_function__ internals>:180, in concatenate(*args, **kwargs)

ValueError: all the input array dimensions for the concatenation axis must m
atch exactly, but along dimension 0, the array at index 0 has size 3 and the
array at index 1 has size 1
```

In [112]:
```
np.vstack((a,b))
```

Out[112]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [ 1,  2,  3,  4]])
```

In [113]:
```
#np.tile
```

In [114]:
```
a=np.arange(1,13).reshape((3,4))
a
```

Out[114]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [115]:
```
np.tile(a,(3,2))
```

Out[115]:
```
array([[ 1,  2,  3,  4,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  5,  6,  7,  8],
       [ 9, 10, 11, 12,  9, 10, 11, 12],
       [ 1,  2,  3,  4,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  5,  6,  7,  8],
       [ 9, 10, 11, 12,  9, 10, 11, 12],
       [ 1,  2,  3,  4,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  5,  6,  7,  8],
       [ 9, 10, 11, 12,  9, 10, 11, 12]])
```

In [ ]: