In [1]:
```
import numpy as np
a=np.arange(15).reshape(3,5)
a
```

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

In [2]:
```
a.shape
```

Out[2]:
```
(3, 5)
```

In [3]:
```
a.ndim
```

Out[3]:
```
2
```

In [4]:
```
a.dtype.name
```

Out[4]:
```
'int32'
```

In [5]:
```
a.itemsize
```

Out[5]:
```
4
```

In [6]:
```
a.size
```

Out[6]:
```
15
```

In [7]:
```
type(a)
```

Out[7]:
```
numpy.ndarray
```

In [8]:
```
a=np.arange(10)**3
```

In [9]:
```
a
```

Out[9]:
```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729], dtype=int32)
```

In [10]:
```
a[2:5]
```

Out[10]:
```
array([ 8, 27, 64], dtype=int32)
```

In [11]:
```
a[:6:2]=1000
```

In [16]:
```
a
```

Out[16]:
```
array([1000,    1, 1000,   27, 1000,  125,  216,  343,  512,  729],
```

```
      dtype=int32)
```

In [17]:
```
a=np.arange(12)**2
i=np.array([1,1,3,8,5])
a[i]
```

Out[17]:
```
array([ 1,  1,  9, 64, 25], dtype=int32)
```

In [18]:
```
a=np.arange(12).reshape(3,4)
a
```

Out[18]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [19]:
```
i=np.array([[0,1],[1,2]])
j=np.array([[2,1],[3,3]])
a[i,j]
```

Out[19]:
```
array([[ 2,  5],
       [ 7, 11]])
```

In [20]:
```
a=np.array([2,3,4,5])
b=np.array([8,5,4])
c=np.array([5,4,6,8,3])
ax,bx,cx=np.ix_(a,b,c)
```

In [21]:
```
ax
```

Out[21]:
```
array([[[2]],

       [[3]],

       [[4]],

       [[5]]])
```

In [22]:
```
bx
```

Out[22]:
```
array([[[8],
        [5],
        [4]]])
```

In [23]:
```
cx
```

Out[23]:
```
array([[[5, 4, 6, 8, 3]]])
```

In [24]:
```
ax.shape,bx.shape,cx.shape
```

Out[24]:
```
((4, 1, 1), (1, 3, 1), (1, 1, 5))
```

In [25]:
```
result=ax+bx+cx
result
```

Out[25]:
```
array([[[15, 14, 16, 18, 13],
```

```
        [12, 11, 13, 15, 10],
        [11, 10, 12, 14,  9]],

       [[16, 15, 17, 19, 14],
        [13, 12, 14, 16, 11],
        [12, 11, 13, 15, 10]],

       [[17, 16, 18, 20, 15],
        [14, 13, 15, 17, 12],
        [13, 12, 14, 16, 11]],

       [[18, 17, 19, 21, 16],
        [15, 14, 16, 18, 13],
        [14, 13, 15, 17, 12]]])
```

In [26]:
```
#import numpy
import numpy as np
a=np.arange(6)
a2=a[np.newaxis,:]
a2.shape
```

Out[26]: (1, 6)

In [28]:
```
a=np.array([[1,2,3,4],[5,6]])
print(a[0])
```

```
[1, 2, 3, 4]
<ipython-input-28-8b6771c64c90>:1: VisibleDeprecationWarning: Creating an ndarray fr
om ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays
with different lengths or shapes) is deprecated. If you meant to do this, you must s
pecify 'dtype=object' when creating the ndarray.
  a=np.array([[1,2,3,4],[5,6]])
```

In [30]:
```
np.zeros(2)
```

Out[30]: array([0., 0.])

In [31]:
```
np.ones(4)
```

Out[31]: array([1., 1., 1., 1.])

In [32]:
```
np.empty(3)
```

Out[32]: array([1.48805998e-311, 0.00000000e+000, 4.94065646e-324])

In [33]:
```
np.arange(4)
```

Out[33]: array([0, 1, 2, 3])

In [34]:
```
np.arange(1,9,3)
```

Out[34]: array([1, 4, 7])

In [35]:
```
np.arange(1,9,2)
```

Out[35]:
```
array([1, 3, 5, 7])
```

In [43]:
```
np.linspace(0,10,num=6)
```

Out[43]:
```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

In [44]:
```
x = np.ones(2, dtype=np.int64)
x
```

Out[44]:
```
array([1, 1], dtype=int64)
```

In [45]:
```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
np.sort(arr)
```

Out[45]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [46]:
```
a = np.array([1, 2, 3, 4])
b = np.array([5, 6, 7, 8])
np.concatenate((a, b))
```

Out[46]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [47]:
```
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6]])
np.concatenate((x, y), axis=0)
```

Out[47]:
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [48]:
```
array_example = np.array([[[0, 1, 2, 3],
[4, 5, 6, 7]],
[[0, 1, 2, 3],
[4, 5, 6, 7]],
[[0 ,1 ,2, 3],
[4, 5, 6, 7]]])
array_example.ndim
```

Out[48]:
```
3
```

In [49]:
```
 array_example.size
```

Out[49]:
```
24
```

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

Out[50]:
```
(3, 2, 4)
```

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

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

In [52]:
```
b = a.reshape(3, 2)
```

In [55]:
```
print(b)
```

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

In [56]:
```
a = np.array([1, 2, 3, 4, 5, 6])
a.shape
```

Out[56]: (6,)

In [57]:
```
a2 = a[np.newaxis, :]
a2.shape
```

Out[57]: (1, 6)

In [58]:
```
row_vector = a[np.newaxis, :]
row_vector.shape
```

Out[58]: (1, 6)

In [59]:
```
col_vector = a[:, np.newaxis]
col_vector.shape
```

Out[59]: (6, 1)

In [60]:
```
a = np.array([1, 2, 3, 4, 5, 6])
a.shape
```

Out[60]: (6,)

In [61]:
```
b = np.expand_dims(a, axis=1)
b.shape
```

Out[61]: (6, 1)

In [62]:
```
data = np.array([1, 2, 3])
```

In [63]:
```
data[1]
```

Out[63]: 2

In [64]:
```
data[0:1]
```

Out[64]: array([1])

In [65]:

```
data[-2:]
```

Out[65]:
```
array([2, 3])
```

In [66]:
```
a = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(a[a < 5])
```

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

In [67]:
```
five_up = (a >= 5)
print(a[five_up])
```

```
[ 5  6  7  8  9 10 11 12]
```

In [68]:
```
divisible_by_2 = a[a%2==0]
print(divisible_by_2)
```

```
[ 2  4  6  8 10 12]
```

In [69]:
```
d= a[(a > 2) & (a < 11)]
print(d)
```

```
[ 3  4  5  6  7  8  9 10]
```

In [70]:
```
five_up = (a > 5) | (a == 5)
print(five_up)
```

```
[[False False False False]
 [ True  True  True  True]
 [ True  True  True  True]]
```

In [ ]: