In [4]:

```
1 import numpy as np
```

In [5]:

```
1 dir(np)
```

Out[5]:

```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
```

In [6]:

```
1 len(dir(np))
```

Out[6]:

```
592
```

In [7]:

```
1 np.__version__
```

Out[7]:

```
'1.24.1'
```

In [8]:

```
1 l = [2, 6, 9, 7, 0]
2 a = np.array(l)
3 a
```

Out[8]:

```
array([2, 6, 9, 7, 0])
```

In [9]:

```
a.ndim    # Number of dimensions
```

Out[9]:

```
1
```

In [10]:

```
print(type(a))
```

```
<class 'numpy.ndarray'>
```

In [11]:

```
a.shape
```

Out[11]:

```
(5,)
```

In [12]:

```
l1 = [[1, 2],
      [5, 0]]
a1 = np.array(l1)
a1
```

Out[12]:

```
array([[1, 2],
       [5, 0]])
```

In [13]:

```
a1.ndim
```

Out[13]:

```
2
```

In [14]:

```
a1.shape
```

Out[14]:

```
(2, 2)
```

In [15]:

```
a.size
```

Out[15]:

```
5
```

In [16]:

```
a1.size
```

Out[16]:

```
4
```

In [17]:

```
type(a)
```

Out[17]:

```
numpy.ndarray
```

In [18]:

```
a.dtype
```

Out[18]:

```
dtype('int32')
```

In [19]:

```
l2 = [[[1, 2, 4],
       [5, 0, 4]],

      [[1, 2, 7],
       [5, 0, 3]]]
a2 = np.array(l2)
a2
```

Out[19]:

```
array([[[1, 2, 4],
        [5, 0, 4]],

       [[1, 2, 7],
        [5, 0, 3]]])
```

In [20]:

```
a2.ndim
```

Out[20]:

```
3
```

In [21]:

```
a2.shape
```

Out[21]:

```
(2, 2, 3)
```

In [22]:

```
l2 = [[[1, 2, 4],
       [5, 0, 4]],

      [[1, 2, 7],
       [5, 0, 3]]]
a2 = np.array(l2, dtype = str)
a2
```

Out[22]:

```
array([[['1', '2', '4'],
        ['5', '0', '4']],

       [['1', '2', '7'],
        ['5', '0', '3']]], dtype='<U1')
```

In [23]:

```
list(range(2, 8))
```

Out[23]:

```
[2, 3, 4, 5, 6, 7]
```

In [24]:

```
b = np.arange(2, 8, 2)
b
```

Out[24]:

```
array([2, 4, 6])
```

In [25]:

```
c = np.zeros((2, 3), dtype = int)
c
```

Out[25]:

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

In [26]:

```
d = np.ones((2, 3))
d
```

Out[26]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [27]:

```
1 np.full((2, 4, 3), 9, dtype = str)
```

Out[27]:

```
array([[['9', '9', '9'],
        ['9', '9', '9'],
        ['9', '9', '9'],
        ['9', '9', '9']],

       [['9', '9', '9'],
        ['9', '9', '9'],
        ['9', '9', '9'],
        ['9', '9', '9']]], dtype='<U1')
```

In [28]:

```
1 np.arange(2, 10, 2)
```

Out[28]:

```
array([2, 4, 6, 8])
```

In [29]:

```
1 np.linspace(2, 10, 4)
```

Out[29]:

```
array([ 2.        ,  4.66666667,  7.33333333, 10.        ])
```

In [30]:

```
1 a2
```

Out[30]:

```
array([[['1', '2', '4'],
        ['5', '0', '4']],

       [['1', '2', '7'],
        ['5', '0', '3']]], dtype='<U1')
```

In [31]:

```
1 a2.shape
```

Out[31]:

```
(2, 2, 3)
```

In [32]:

```
a3 = a2.reshape((3, 4))
a3
```

Out[32]:

```
array([['1', '2', '4', '5'],
       ['0', '4', '1', '2'],
       ['7', '5', '0', '3']], dtype='<U1')
```

In [33]:

```
a2
```

Out[33]:

```
array([[['1', '2', '4'],
        ['5', '0', '4']],

       [['1', '2', '7'],
        ['5', '0', '3']]], dtype='<U1')
```

In [34]:

```
a4 = a2.flatten()
a4
```

Out[34]:

```
array(['1', '2', '4', '5', '0', '4', '1', '2', '7', '5', '0', '3'],
      dtype='<U1')
```

In [35]:

```
a3
```

Out[35]:

```
array([['1', '2', '4', '5'],
       ['0', '4', '1', '2'],
       ['7', '5', '0', '3']], dtype='<U1')
```

In [36]:

```
a5 = np.asarray(a3, dtype = int)
a5
```

Out[36]:

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

Indexing in Array

In [37]:

```
1 a5[0]
```

Out[37]:

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

In [38]:

```
1 a5[0][2]
```

Out[38]:

```
4
```

Integer Array Indexing

In [39]:

```
1 a5[[1,2],[1,3]]
```

Out[39]:

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

Boolean Array Indexing

In [40]:

```
1 a5[a5 > 4]
```

Out[40]:

```
array([5, 7, 5])
```

In [41]:

```
1 a5
```

Out[41]:

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

In [42]:

```
1 a5[a5 % 2 == 0]
```

Out[42]:

```
array([2, 4, 0, 4, 2, 0])
```

Arithmetic Operations on Array

In [43]:

```
1 l
```

Out[43]:

```
[2, 6, 9, 7, 0]
```

In [44]:

```
1 # l + 1
```

In [45]:

```
1 a
```

Out[45]:

```
array([2, 6, 9, 7, 0])
```

In [46]:

```
1 a + 1
```

Out[46]:

```
array([ 3,  7, 10,  8,  1])
```

In [47]:

```
1 a = a + 1   # a += 1
2 a
```

Out[47]:

```
array([ 3,  7, 10,  8,  1])
```

In [48]:

```
1 a
```

Out[48]:

```
array([ 3,  7, 10,  8,  1])
```

In [49]:

```
1 a1
```

Out[49]:

```
array([[1, 2],
       [5, 0]])
```

In [109]:

```
# modulus
arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.mod(arr1, arr2)

print(newarr)

# power
arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 5, 6, 8, 2, 33])

newarr = np.power(arr1, arr2)

print(newarr)

```

```
[ 1  6  3  0  0 27]
[      1000    3200000  729000000 -520093696       2500          0]
```

Slicing

In [50]:

```
a5
```

Out[50]:

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

In [51]:

```
a6 = a5[1:,1:3]
a6
```

Out[51]:

```
array([[4, 1],
       [5, 0]])
```

In [52]:

```
a5[:2, 1::2]
```

Out[52]:

```
array([[2, 5],
       [4, 2]])
```

Element-wise Arithmetic Operations

In [53]:

```
1 a1
```

Out[53]:

```
array([[1, 2],
       [5, 0]])
```

In [54]:

```
1 a6
```

Out[54]:

```
array([[4, 1],
       [5, 0]])
```

In [55]:

```
1 a16 = a1 + a6
2 a16
```

Out[55]:

```
array([[ 5,  3],
       [10,  0]])
```

In [56]:

```
1 a1 * a6           # Element-wise multiplication
```

Out[56]:

```
array([[ 4,  2],
       [25,  0]])
```

In [57]:

```
1 a1.dot(a6)        # Matrix Multiplication
```

Out[57]:

```
array([[14,  1],
       [20,  5]])
```

In [58]:

```
1 a1
```

Out[58]:

```
array([[1, 2],
       [5, 0]])
```

In [59]:

```
1 a1.T
```

Out[59]:

```
array([[1, 5],
       [2, 0]])
```

In [60]:

```
1 a3
```

Out[60]:

```
array([['1', '2', '4', '5'],
       ['0', '4', '1', '2'],
       ['7', '5', '0', '3']], dtype='<U1')
```

In [61]:

```
1 a3.dtype = int
```

In [62]:

```
1 a3
```

Out[62]:

```
array([[49, 50, 52, 53],
       [48, 52, 49, 50],
       [55, 53, 48, 51]])
```

In [63]:

```
1 a4 = np.array(a3, dtype = int)
2 a4
```

Out[63]:

```
array([[49, 50, 52, 53],
       [48, 52, 49, 50],
       [55, 53, 48, 51]])
```

In [64]:

```
1 a4.max()
```

Out[64]:

```
55
```

In [65]:

```
1 a4.max(axis = 0)
```

Out[65]:

```
array([55, 53, 52, 53])
```

In [66]:

```
1 a4.max(axis = 1)
```

Out[66]:

```
array([53, 52, 55])
```

In [67]:

```
1 a4.min()
```

Out[67]:

```
48
```

In [68]:

```
1 a4.sum()
```

Out[68]:

```
610
```

In [69]:

```
1 a4.sum(axis = 0)
```

Out[69]:

```
array([152, 155, 149, 154])
```

In [70]:

```
1 a4.mean()
```

Out[70]:

```
50.833333333333336
```

In [71]:

```
1 a4
```

Out[71]:

```
array([[49, 50, 52, 53],
       [48, 52, 49, 50],
       [55, 53, 48, 51]])
```

In [72]:

```
1 a4.cumsum(axis = 0)
```

Out[72]:

```
array([[ 49,  50,  52,  53],
       [ 97, 102, 101, 103],
       [152, 155, 149, 154]])
```

In [73]:

```
1 a4.cumprod(axis = 0)
```

Out[73]:

```
array([[    49,     50,     52,     53],
       [  2352,   2600,   2548,   2650],
       [129360, 137800, 122304, 135150]])
```

In [74]:

```
1 a4
```

Out[74]:

```
array([[49, 50, 52, 53],
       [48, 52, 49, 50],
       [55, 53, 48, 51]])
```

In [75]:

```
1 np.pi
```

Out[75]:

```
3.141592653589793
```

In [76]:

```
1 x = np.arange(0, 2*np.pi, 0.1)
2 x
```

Out[76]:

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2,
       1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5,
       2.6, 2.7, 2.8, 2.9, 3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8,
       3.9, 4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. , 5.1,
       5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 6. , 6.1, 6.2])
```

In [77]:

```
1 y = np.sin(x)
2 y
```

Out[77]:

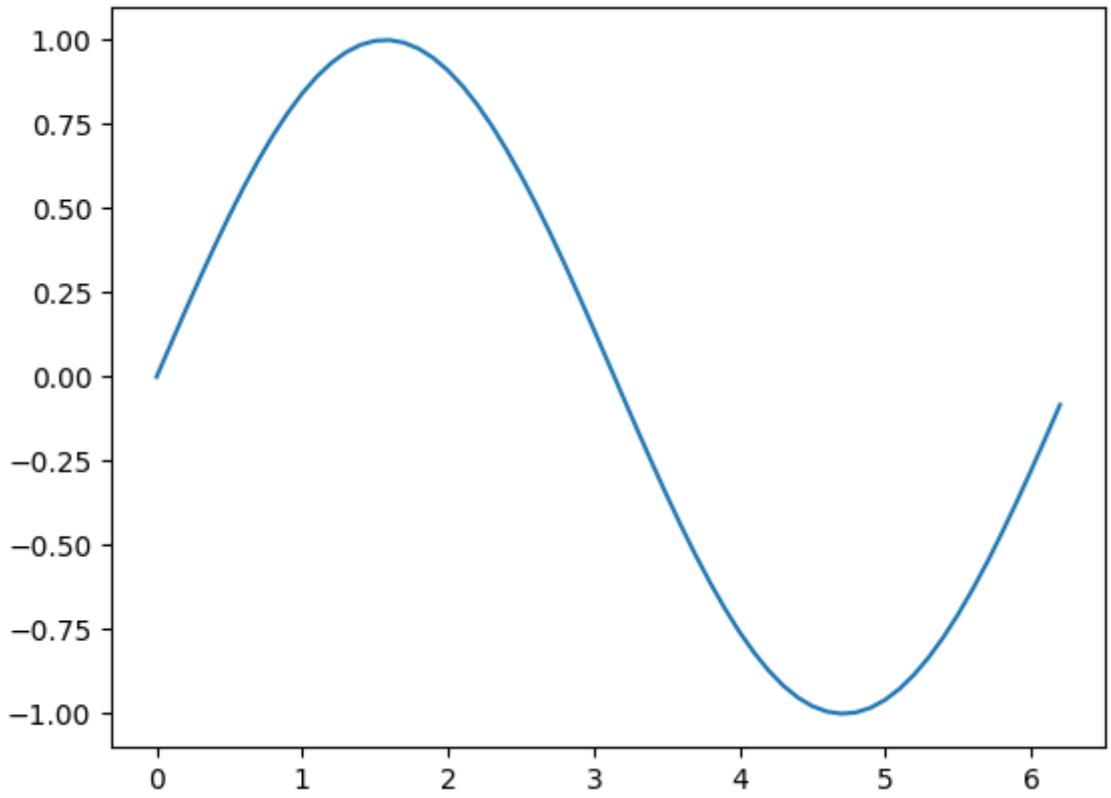
```
array([ 0.        ,  0.09983342,  0.19866933,  0.29552021,  0.38941834,
        0.47942554,  0.56464247,  0.64421769,  0.71735609,  0.78332691,
        0.84147098,  0.89120736,  0.93203909,  0.96355819,  0.98544973,
        0.99749499,  0.9995736 ,  0.99166481,  0.97384763,  0.94630009,
        0.90929743,  0.86320937,  0.8084964 ,  0.74570521,  0.67546318,
        0.59847214,  0.51550137,  0.42737988,  0.33498815,  0.23924933,
        0.14112001,  0.04158066, -0.05837414, -0.15774569, -0.2555411 ,
       -0.35078323, -0.44252044, -0.52983614, -0.61185789, -0.68776616,
       -0.7568025 , -0.81827711, -0.87157577, -0.91616594, -0.95160207,
       -0.97753012, -0.993691  , -0.99992326, -0.99616461, -0.98245261,
       -0.95892427, -0.92581468, -0.88345466, -0.83226744, -0.77276449,
       -0.70554033, -0.63126664, -0.55068554, -0.46460218, -0.37387666,
       -0.2794155 , -0.1821625 , -0.0830894 ])
```

In [78]:

```
1 import matplotlib.pyplot as plt
2 # from matplotlib import pyplot as plt
3 plt.plot(x, y)
```

Out[78]:

```
[<matplotlib.lines.Line2D at 0x1c88322e1a0>]
```



In [79]:

```
1 np.cos(x)
```

Out[79]:

```
array([ 1.        ,  0.99500417,  0.98006658,  0.95533649,  0.92106099,
        0.87758256,  0.82533561,  0.76484219,  0.69670671,  0.62160997,
        0.54030231,  0.45359612,  0.36235775,  0.26749883,  0.16996714,
        0.0707372 , -0.02919952, -0.12884449, -0.22720209, -0.32328957,
       -0.41614684, -0.5048461 , -0.58850112, -0.66627602, -0.73739372,
       -0.80114362, -0.85688875, -0.90407214, -0.94222234, -0.97095817,
       -0.9899925 , -0.99913515, -0.99829478, -0.98747977, -0.96679819,
       -0.93645669, -0.89675842, -0.84810003, -0.79096771, -0.7259323 ,
       -0.65364362, -0.57482395, -0.49026082, -0.40079917, -0.30733287,
       -0.2107958 , -0.11215253, -0.01238866,  0.08749898,  0.18651237,
        0.28366219,  0.37797774,  0.46851667,  0.55437434,  0.63469288,
        0.70866977,  0.77556588,  0.83471278,  0.88551952,  0.92747843,
        0.96017029,  0.98326844,  0.9965421 ])
```

In [80]:

```
1 np.tan(x)
```

Out[80]:

```
array([ 0.00000000e+00,  1.00334672e-01,  2.02710036e-01,  3.09336250e-01,
        4.22793219e-01,  5.46302490e-01,  6.84136808e-01,  8.42288380e-01,
        1.02963856e+00,  1.26015822e+00,  1.55740772e+00,  1.96475966e+00,
        2.57215162e+00,  3.60210245e+00,  5.79788372e+00,  1.41014199e+01,
       -3.42325327e+01, -7.69660214e+00, -4.28626167e+00, -2.92709751e+00,
       -2.18503986e+00, -1.70984654e+00, -1.37382306e+00, -1.11921364e+00,
       -9.16014290e-01, -7.47022297e-01, -6.01596613e-01, -4.72727629e-01,
       -3.55529832e-01, -2.46405394e-01, -1.42546543e-01, -4.16166546e-02,
        5.84738545e-02,  1.59745748e-01,  2.64316901e-01,  3.74585640e-01,
        4.93466730e-01,  6.24733075e-01,  7.73556091e-01,  9.47424650e-01,
        1.15782128e+00,  1.42352648e+00,  1.77777977e+00,  2.28584788e+00,
        3.09632378e+00,  4.63733205e+00,  8.86017490e+00,  8.07127630e+01,
       -1.13848707e+01, -5.26749307e+00, -3.38051501e+00, -2.44938942e+00,
       -1.88564188e+00, -1.50127340e+00, -1.21754082e+00, -9.95584052e-01,
       -8.13943284e-01, -6.59730572e-01, -5.24666222e-01, -4.03110900e-01,
       -2.91006191e-01, -1.85262231e-01, -8.33777149e-02])
```

In [81]:

```
1 np.sqrt(x)
```

Out[81]:

```
array([0.        , 0.31622777, 0.4472136 , 0.54772256, 0.63245553,
       0.70710678, 0.77459667, 0.83666003, 0.89442719, 0.9486833 ,
       1.        , 1.04880885, 1.09544512, 1.14017543, 1.18321596,
       1.22474487, 1.26491106, 1.30384048, 1.34164079, 1.37840488,
       1.41421356, 1.44913767, 1.4832397 , 1.51657509, 1.54919334,
       1.58113883, 1.61245155, 1.64316767, 1.67332005, 1.70293864,
       1.73205081, 1.76068169, 1.78885438, 1.81659021, 1.84390889,
       1.87082869, 1.8973666 , 1.92353841, 1.94935887, 1.97484177,
       2.        , 2.02484567, 2.04939015, 2.07364414, 2.0976177 ,
       2.12132034, 2.14476106, 2.16794834, 2.19089023, 2.21359436,
       2.23606798, 2.25831796, 2.28035085, 2.30217289, 2.32379001,
       2.34520788, 2.36643191, 2.38746728, 2.40831892, 2.42899156,
       2.44948974, 2.46981781, 2.48997992])
```

In [82]:

```
1 np.power(x, 4)
```

Out[82]:

```
array([0.0000000e+00, 1.0000000e-04, 1.6000000e-03, 8.1000000e-03,
       2.5600000e-02, 6.2500000e-02, 1.2960000e-01, 2.4010000e-01,
       4.0960000e-01, 6.5610000e-01, 1.0000000e+00, 1.4641000e+00,
       2.0736000e+00, 2.8561000e+00, 3.8416000e+00, 5.0625000e+00,
       6.5536000e+00, 8.3521000e+00, 1.0497600e+01, 1.3032100e+01,
       1.6000000e+01, 1.9448100e+01, 2.3425600e+01, 2.7984100e+01,
       3.3177600e+01, 3.9062500e+01, 4.5697600e+01, 5.3144100e+01,
       6.1465600e+01, 7.0728100e+01, 8.1000000e+01, 9.2352100e+01,
       1.0485760e+02, 1.1859210e+02, 1.3363360e+02, 1.5006250e+02,
       1.6796160e+02, 1.8741610e+02, 2.0851360e+02, 2.3134410e+02,
       2.5600000e+02, 2.8257610e+02, 3.1116960e+02, 3.4188010e+02,
       3.7480960e+02, 4.1006250e+02, 4.4774560e+02, 4.8796810e+02,
       5.3084160e+02, 5.7648010e+02, 6.2500000e+02, 6.7652010e+02,
       7.3116160e+02, 7.8904810e+02, 8.5030560e+02, 9.1506250e+02,
       9.8344960e+02, 1.0556001e+03, 1.1316496e+03, 1.2117361e+03,
       1.2960000e+03, 1.3845841e+03, 1.4776336e+03])
```

In [83]:

```
1 # np.insert?
```

Random

In [102]:

```
1 from numpy import random
2 np.random.randint(3, 9)
```

Out[102]:

```
5
```

In [85]:

```
1 np.random.randint(3, 9, (2, 3))
```

Out[85]:

```
array([[3, 5, 5],
       [5, 8, 6]])
```

In [86]:

```
1 a
```

Out[86]:

```
array([ 3,  7, 10,  8,  1])
```

In [87]:

```
1 np.random.choice(a, (2, 3))
```

Out[87]:

```
array([[10,  3, 10],
       [ 3,  1,  1]])
```

In [88]:

```
1 a5
```

Out[88]:

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

In [89]:

```
1 np.random.choice(a5.flatten(), (2, 3))
```

Out[89]:

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

In [90]:

```
1 a5
```

Out[90]:

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

nditer()

In [91]:

```
1
2
3 arr = np.array([[[3, 6], [2, 1]], [[6, 4], [2, 5]]])
4
5 for x in np.nditer(arr):
6     print(x)
7
```

```
1
2
3
4
5
6
7
8
```

denumerate()

In [92]:

```
arr = np.array([6,4,6])

for idx, x in np.ndenumerate(arr):
    print(idx, x)

```

```
(0,) 6
(1,) 4
(2,) 6
```

statistical analysis in numpy

In [100]:

```
arr1 = np.array([[1,2],[3,4]])

print(np.mean(arr1))   #mean

arr2 = np.array([[4,6],[2,4]])
print(np.median(arr2))    #median

arr3 = [1,2,3,3,3,4,4,4,5,3,5,6]
# to find mode you have to import a  library
from scipy import stats as st
print(st.mode(arr3))

print(np.std(arr1))   #stadars deviation
```

```
2.5
4.0
ModeResult(mode=array([3]), count=array([4]))
1.118033988749895

C:\Users\Parn\AppData\Local\Temp\ipykernel_5212\1372092990.py:11: FutureWa
rning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the def
ault behavior of `mode` typically preserves the axis it acts along. In Sci
Py 1.11.0, this behavior will change: the default value of `keepdims` will
become False, the `axis` over which the statistic is taken will be elimina
ted, and the value None will no longer be accepted. Set `keepdims` to True
or False to avoid this warning.
  print(st.mode(arr3))
```

suffling in array

In [103]:

```
arr = np.array([1, 2, 3, 4, 5])

random.shuffle(arr)

print(arr)

```

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

intersection and union

In [106]:

```
# Example 3:

import numpy as np

arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([4, 5, 6, 7])

InterSection = np.intersect1d(arr1, arr2, assume_unique=True)  #intersection

print(newarr)

Union = np.union1d(arr1, arr2)
print(Union)
```

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

In [ ]:

```

```