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

In [3]:
```
np.array([2,4,56,422,32,1])
```

Out[3]:
```
array([  2,   4,  56, 422,  32,   1])
```

In [9]:
```
a=np.array([2,4,56,422,32,1])
print(a)
```

```
[  2   4  56 422  32   1]
```

In [11]:
```
type(a)
```

Out[11]:
```
numpy.ndarray
```

In [15]:
```
#2d array


new=np.array([[45,34,22,2],[24,55,3,22]])
print(new)
```

```
[[45 34 22  2]
 [24 55  3 22]]
```

In [21]:
```
# 3d array

np.array([[2,3,33,4,45],[23,45,56,66,2],[357,523,32,24,2],[32,32,44,33,234]])
```

Out[21]:
```
array([[  2,   3,  33,   4,  45],
       [ 23,  45,  56,  66,   2],
       [357, 523,  32,  24,   2],
       [ 32,  32,  44,  33, 234]])
```

In [25]:
```
np.array([11,23,44],dtype=float)
```

Out[25]:
```
array([11., 23., 44.])
```

In [27]:
```
np.array([11,23,44],dtype=bool)
```

Out[27]:
```
array([ True,  True,  True])
```

In [29]:
```
np.array([11,23,44],dtype=complex)
```

Out[29]:
```
array([11.+0.j, 23.+0.j, 44.+0.j])
```

In [33]:
```
np.arange(1,25)
```

Out[33]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24])
```

In [35]:
```
np.arange(1,25,2)
```

Out[35]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23])
```

In [39]:
```
np.arange(1,11).reshape(5,2)
```

Out[39]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

In [41]:
```
np.arange(1,13).reshape(3,4)
```

Out[41]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [43]:
```
np.ones((3,4))
```

Out[43]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [45]:
```
np.zeros((3,4))
```

Out[45]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [47]:
```
np.random.random((4,3))
```

Out[47]:
```
array([[0.18377237, 0.39089979, 0.19865015],
       [0.16753346, 0.26468191, 0.4557862 ],
       [0.28534927, 0.8769399 , 0.4575609 ],
       [0.93195717, 0.5207416 , 0.84634075]])
```

In [51]:
```
np.linspace(-10,10,10)
```

Out[51]:
```
array([-10.        ,  -7.77777778,  -5.55555556,  -3.33333333,
        -1.11111111,   1.11111111,   3.33333333,   5.55555556,
         7.77777778,  10.        ])
```

In [53]:
```
np.linspace(-2,12,6)
```

Out[53]:
```
array([-2. ,  0.8,  3.6,  6.4,  9.2, 12. ])
```

In [55]:
```
np.identity(3)
```

Out[55]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [57]:
```
np.identity(6)
```

Out[57]:
```
array([[1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1.]])
```

In [63]:
```
a1 = np.arange(10)
a1
```

Out[63]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [73]:
```
a2 = np.arange(12, dtype =float).reshape(3,4)
a2
```

Out[73]:
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]])
```

In [77]:
```
a3 = np.arange(8).reshape(2,2,2)
a3
```

Out[77]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [79]:
```
a1.ndim
```

Out[79]:
```
1
```

In [81]:
```
a2.ndim
```

Out[81]:
```
2
```

In [83]:
```
a3.ndim
```

Out[83]:
```
3
```

In [87]:
```
a1.shape
```

Out[87]:
```
(10,)
```

In [90]:
```
a2.shape
```

Out[90]:
```
(3, 4)
```

a3.shape

In [94]:
```
a3
```

Out[94]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [96]:
```
a3
```

Out[96]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [98]:
```
a3.size
```

Out[98]: 8

In [100…
```
a2
```

Out[100…
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]])
```

In [102…
```
a2.size
```

Out[102… 12

In [104…
```
a1
```

Out[104…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [106…
```
a1.size
```

Out[106… 10

In [108…
```
a1
```

Out[108…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [110…
```
a1.itemsize
```

Out[110… 4

In [112…
```
a2
```

Out[112…
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]])
```

In [114…
```
a2.itemsize
```

Out[114… 8

In [116…
```
a3
```

Out[116…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [118…
```
a3.itemsize
```

Out[118… 4

In [120…
```
a1
```

Out[120…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [122…
```
print(a1.dtype)
```

```
int32
```

In [124…
```
a2
```

Out[124…
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]])
```

In [126…
```
print(a2.dtype)
```

```
float64
```

In [128…
```
a3
```

Out[128…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [130…
```
print(a3.dtype)
```

```
int32
```

In [136…
```
x=np.array([33,22,2.5])
x
```

Out[136…
```
array([33. , 22. ,  2.5])
```

In [138…
```
x.astype(int)
```

Out[138…
```
array([33, 22,  2])
```

z1 = np.arange(12).reshape(3,4) z2 = np.arange(12,24).reshape(3,4)

In [144…
```
z1
```

Out[144…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [146…
```
z2
```

Out[146…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [148…
```
#arithmetic
z1+2
```

Out[148…
```
array([[ 2,  3,  4,  5],
       [ 6,  7,  8,  9],
       [10, 11, 12, 13]])
```

In [160…
```
z2+2
```

Out[160…
```
array([[14, 15, 16, 17],
       [18, 19, 20, 21],
       [22, 23, 24, 25]])
```

In [152…
```
#subtraction
```

```
z1-2
```

Out[152…
```
array([[-2, -1,  0,  1],
       [ 2,  3,  4,  5],
       [ 6,  7,  8,  9]])
```

In [162…
```
z2-2
```

Out[162…
```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

In [154…
```
#multiplication
z1*2
```

Out[154…
```
array([[ 0,  2,  4,  6],
       [ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

In [156…
```
#power
z1**2
```

Out[156…
```
array([[  0,   1,   4,   9],
       [ 16,  25,  36,  49],
       [ 64,  81, 100, 121]])
```

In [164…
```
z2**2
```

Out[164…
```
array([[144, 169, 196, 225],
       [256, 289, 324, 361],
       [400, 441, 484, 529]])
```

In [158…
```
##modulo
z1%2
```

Out[158…
```
array([[0, 1, 0, 1],
       [0, 1, 0, 1],
       [0, 1, 0, 1]], dtype=int32)
```

In [166…
```
z2%2
```

Out[166…
```
array([[0, 1, 0, 1],
       [0, 1, 0, 1],
       [0, 1, 0, 1]], dtype=int32)
```

In [168…
```
z2
```

Out[168…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [172…
```
z2 > 2
```

Out[172…
```
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

In [174…
```
z2 >20
```

Out[174…
```
array([[False, False, False, False],
       [False, False, False, False],
       [False,  True,  True,  True]])
```

In [176…
```
z1
```

Out[176…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [178…
```
z1 >2
```

Out[178…
```
array([[False, False, False,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

In [180…
```
z1 >20
```

Out[180…
```
array([[False, False, False, False],
       [False, False, False, False],
       [False, False, False, False]])
```

In [182…
```
z1
```

Out[182…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [184…
```
z2
```

Out[184…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [186…
```
#arithemetic
z1+z2
```

Out[186…
```
array([[12, 14, 16, 18],
       [20, 22, 24, 26],
       [28, 30, 32, 34]])
```

In [188…
```
z1-z2
```

Out[188…
```
array([[-12, -12, -12, -12],
       [-12, -12, -12, -12],
       [-12, -12, -12, -12]])
```

In [190…
```
z1*z2
```

Out[190…
```
array([[  0,  13,  28,  45],
       [ 64,  85, 108, 133],
       [160, 189, 220, 253]])
```

In [192…
```
z1%z2
```

Out[192…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [196…
```
z1**z2
```

Out[196…
```
array([[          0,           1,       16384,    14348907],
       [          0, -1564725563,  1159987200,   442181591],
       [          0,  1914644777, -1304428544,  -122979837]])
```

In [198…
```
z1/z2
```

Out[198…
```
array([[0.        , 0.07692308, 0.14285714, 0.2       ],
       [0.25      , 0.29411765, 0.33333333, 0.36842105],
       [0.4       , 0.42857143, 0.45454545, 0.47826087]])
```

In [206…
```
k1 = np.random.random((3,3))
k1 = np.round(k1*100)
k1
```

Out[206…
```
array([[ 18.,  71.,  68.],
       [ 61.,  13., 100.],
       [ 44.,  39.,  57.]])
```

In [208…
```
#max
np.max(k1)
```

Out[208…
```
100.0
```

In [210…
```
#min
np.min(k1)
```

Out[210…
```
13.0
```

In [212…
```
#sum
np.sum(k1)
```

Out[212…
```
471.0
```

In [214…
```
#product---->multiplication
np.prod(k1)
```

Out[214…
```
674070146006400.0
```

In [218…
```
#if we want maximum of every row
np.max(k1,axis = 1)
```

Out[218…
```
array([ 71., 100.,  57.])
```

In [220…
```
#maximum of every column
np.max(k1, axis = 0)
```

Out[220…
```
array([ 61.,  71., 100.])
```

In [226…
```
#product of every column
np.prod(k1, axis = 0)
```

Out[226…
```
array([ 48312.,  35997., 387600.])
```

In [228…
```
#mean
k1
```

Out[228…
```
array([[ 18.,  71.,  68.],
       [ 61.,  13., 100.],
       [ 44.,  39.,  57.]])
```

In [236…
```
np.mean(k1)
```

Out[236…
```
52.333333333333336
```

In [238…
```
#mean of every column
k1.mean(axis=0)
```

Out[238…
```
array([41., 41., 75.])
```

In [240…
```
#median
np.median(k1)
```

Out[240…
```
57.0
```

In [242…
```
np.median(k1,axis = 1)
```

Out[242…
```
array([68., 61., 44.])
```

In [244…
```
#standard deviation
np.std(k1)
```

Out[244…
```
25.681813712344294
```

In [246…
```
np.std(k1,axis = 0)
```

Out[246…
```
array([17.68238295, 23.72059583, 18.23915203])
```

In [248…
```
#variance
np.var(k1)
```

Out[248…
```
659.5555555555554
```

In [250…
```
# TRIGNOMETRIC FUNCTIONS
np.sin(k1)
```

Out[250…
```
array([[-0.75098725,  0.95105465, -0.89792768],
       [-0.96611777,  0.42016704, -0.50636564],
       [ 0.01770193,  0.96379539,  0.43616476]])
```

In [252…
```
np.cos(k1)
```

Out[252…
```
array([[ 0.66031671, -0.30902273,  0.44014302],
       [-0.25810164,  0.90744678,  0.86231887],
       [ 0.99984331,  0.26664293,  0.89986683]])
```

In [256…
```
np.tan(k1)
```

Out[256…
```
array([[-1.13731371, -3.0776204 , -2.0400816 ],
       [ 3.74316794,  0.46302113, -0.58721392],
       [ 0.0177047 ,  3.61455441,  0.48469923]])
```

In [258…
```
#DOT PRODUCT
s2 = np.arange(12).reshape(3,4)
```

In [260…
```
s2
```

Out[260…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [268…
```
s3 = np.arange(12,24).reshape(4,3)
```

In [270…
```
s3
```

Out[270…
```
array([[12, 13, 14],
       [15, 16, 17],
       [18, 19, 20],
       [21, 22, 23]])
```

In [272…
```
np.dot(s2,s3)
```

Out[272…
```
array([[114, 120, 126],
       [378, 400, 422],
       [642, 680, 718]])
```

In [274…
```
#LOG AND EXPONENT
np.exp(s2)
```

Out[274…
```
array([[1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01],
       [5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03],
       [2.98095799e+03, 8.10308393e+03, 2.20264658e+04, 5.98741417e+04]])
```

In [276…
```
#ROUND TO THE NEAREST INTEGER
arr = np.array([1.2,2.7,3.5,4.9])
rounded_arr = np.round(arr)
print(rounded_arr)
```

```
[1. 3. 4. 5.]
```

In [278…
```
#ROUND TO TWO DECIMALS
arr = np.array([1.234,2.567,3.891])
rounded_arr = np.round(arr,decimals=2)
print(rounded_arr)
```

```
[1.23 2.57 3.89]
```

In [280…
```
#RANDOMLY
np.round(np.random.random((2,3))*100)
```

Out[280…
```
array([[ 6., 74.,  3.],
       [53., 86., 70.]])
```

In [282…
```
#FLOOR OPERATION
arr = np.array([1.2,2.7,3.5,4.9])
floored_arr = np.floor(arr)
print(floored_arr)
```

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

In [284…
```
np.floor(np.random.random((2,3))*100)
```

Out[284…
```
array([[71., 89., 50.],
       [38.,  2., 44.]])
```

In [286…
```
#CELL
arr = np.array([1.2,2.7,3.5,4.9])
ceiled_arr = np.ceil(arr)
print(ceiled_arr)
```

```
[2. 3. 4. 5.]
```

In [288…
```
np.ceil(np.random.random((2,3))*100)
```

Out[288…
```
array([[98., 17., 36.],
       [ 5., 65., 22.]])
```

In [290…
```
#INDEXING AND SLICING
p1 = np.arange(10)
p2 = np.arange(12).reshape(3,4)
p3 = np.arange(8).reshape(2,2,2)
```

In [292…
```
p1
```

Out[292…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [294…
```
p2
```

Out[294…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [296…
```
p3
```

Out[296…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [298…
```
#INDEXING ON 1D ARRAY
p1
```

Out[298…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [300…
```
#FETCHING LAST ITEM
p1[-1]
```

Out[300…
```
9
```

In [302…
```
#FETCHING FIRST IETM
p1[0]
```

Out[302…
```
0
```

In [304…
```
#INDEXING ON 2D ARRAY
p2
```

Out[304…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [306…
```
#FETCHING DESIRED ELEMENT : 6
p2[1,2]
```

Out[306…
```
6
```

In [308…
```
#FETCHING DESIRED ELEMENT : 11
p2[2,3]
```

Out[308…
```
11
```

In [310…
```
#FETCHINH DESIRED ELEMENT : 4
p2[1,0]
```

Out[310…
```
4
```

In [312…
```
#INDEXING ON 3D ELEMENT
p3
```

Out[312…
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [314…
```
#FETCHING DESIRED ELEMENT : 5
p3[1,0,1]
```

Out[314…
```
5
```

In [318…
```
#FETCHING DESIRED ELEMENT : 2
p3[0,1,0]
```

Out[318…
```
2
```

In [320…
```
#FETCHING DESIRED ELEMENT : 0
p3[0,0,0]
```

Out[320…
```
0
```

In [322…
```
#FETCHING DESIRED ELEMENT : 6
p3[1,1,0]
```

Out[322…
```
6
```

In [324…
```
#SLICING
p1
```

Out[324…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [326…
```
# fetching desired elements are : 2,3,4
p1[2:5]
```

Out[326…
```
array([2, 3, 4])
```

In [328…
```
# Alternate
p1[2:5:2]
```

Out[328… array([2, 4])

In [330…
```
# SLICING ON 2D
p2
```

Out[330…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [332…
```
#fetching total first row
p2[0,:]
```

Out[332… array([0, 1, 2, 3])

In [340…
```
#FETCHING TOTAL THIRD COLUMN
p2[:,2]
```

Out[340… array([2, 6, 10])

In [342…
```
#fetch 5,6 and 9,10
p2
```

Out[342…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [344…
```
p2[1:3]
```

Out[344…
```
array([[ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [346…
```
p2[1:3,1:3]
```

Out[346…
```
array([[ 5,  6],
       [ 9, 10]])
```

In [348…
```
#fetch 0,3 and 8,11
p2
```

Out[348…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [350…
```
p2[::2,::3]
```

Out[350…
```
array([[ 0,  3],
       [ 8, 11]])
```

In [352…
```
#fetch 1,3 and9,11
p2
```

Out[352…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [354…
```
p2[::2]
```

Out[354…
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [356…
```
p2[::2,1::2]
```

Out[356…
```
array([[ 1,  3],
       [ 9, 11]])
```

In [358…
```
p2[::2,1::2]
```

Out[358…
```
array([[ 1,  3],
       [ 9, 11]])
```

In [360…
```
#fetch only 4,7
p2
```

Out[360…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [362…
```
p2[1]
```

Out[362…
```
array([4, 5, 6, 7])
```

In [364…
```
p2[1,::3]
```

Out[364…
```
array([4, 7])
```

In [368…
```
#fetch 1,2,3 and 5,6,7
p2
```

Out[368…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [370…
```
p2[0:2]
```

Out[370…
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [372…
```
p2[0:2,1:]
```

Out[372…
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [374…
```
#fetch 1,3 and 5,7
p2
```

Out[374…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [376…
```
p2[0:2]
```

Out[376…
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [378…
```
p2[0:2,1::2]
```

Out[378…
```
array([[1, 3],
       [5, 7]])
```

In [380…
```
#slicing in 3d
p3 = np.arange(27).reshape(3,3,3)
p3
```

Out[380…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [382…
```
#fetch second matrix
p3[1]
```

Out[382…
```
array([[ 9, 10, 11],
       [12, 13, 14],
       [15, 16, 17]])
```

In [384…
```
#fetch first and last
p3[::2]
```

Out[384…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [386…
```
#fetch 1 2d array's 2 row-->3,4,5
p3
```

Out[386…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [388…
```
p3[0]
```

Out[388…
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [390…
```
p3[0,1,:]
```

Out[390…
```
array([3, 4, 5])
```

In [392…
```
#fetch 2 numpy array,middle column-->10,13,16
p3
```

Out[392…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [394…
```
p3[1]
```

Out[394…
```
array([[ 9, 10, 11],
       [12, 13, 14],
       [15, 16, 17]])
```

In [396…
```
p3[1,:,1]
```

Out[396…
```
array([10, 13, 16])
```

In [398…
```
#fetch 3 array-->22,23,25,26
p3
```

Out[398…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [400…
```
p3[2]
```

Out[400…
```
array([[18, 19, 20],
       [21, 22, 23],
       [24, 25, 26]])
```

In [406…
```
p3[2,1::]
```

Out[406…
```
array([[21, 22, 23],
       [24, 25, 26]])
```

In [408…
```
p3[2,1: ,1:]
```

Out[408…
```
array([[22, 23],
       [25, 26]])
```

In [410…
```
#fetch 0,2,18,20
p3
```

Out[410…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [412…
```
p3[0::2]
```

Out[412…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [414…
```
p3[0::2,0]
```

Out[414…
```
array([[ 0,  1,  2],
       [18, 19, 20]])
```

In [420…
```
p3[0::2 ,0 , ::2]
```

Out[420…
```
array([[ 0,  2],
       [18, 20]])
```

In [422…
```
 # ITERATING
p1
```

Out[422…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [424…
```
#Looping on 1d array
for i in p1:
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [426…
```
p2
```

Out[426…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [428…
```
## Looping on 2D array
for i in p2:
```

```
    print(i)
```

```
[0 1 2 3]
[4 5 6 7]
[ 8  9 10 11]
```

In [430…
```
p3
```

Out[430…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [434…
```
for i in p3:
     print(i)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[ 9 10 11]
 [12 13 14]
 [15 16 17]]
[[18 19 20]
 [21 22 23]
 [24 25 26]]
```

In [440…
```
for i in np.nditer(p3):
   print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

In [442…
```
#Transpose
p2
```

Out[442…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [444…
```
np.transpose(p2)
```

Out[444…
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [448…
```
#Another method
p2.T
```

Out[448…
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [450…
```
p3
```

Out[450…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [452…
```
p3.T
```

Out[452…
```
array([[[ 0,  9, 18],
        [ 3, 12, 21],
        [ 6, 15, 24]],

       [[ 1, 10, 19],
        [ 4, 13, 22],
        [ 7, 16, 25]],

       [[ 2, 11, 20],
        [ 5, 14, 23],
        [ 8, 17, 26]]])
```

In [454…
```
#Ravel
p2
```

Out[454…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [456…
```
p2.ravel()
```

Out[456…
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [458…
```
p3
```

Out[458…
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [460…
```
p3.ravel()
```

Out[460…
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26])
```

In [468…
```
#STACKING
##horizontal stacking
```

```
w1 = np.arange(12).reshape(3,4)
w2 = np.arange(12,24).reshape(3,4)
```

In [472…
```
w1
```

Out[472…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [474…
```
w2
```

Out[474…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [478…
```
np.hstack((w1,w2))
```

Out[478…
```
array([[ 0,  1,  2,  3, 12, 13, 14, 15],
       [ 4,  5,  6,  7, 16, 17, 18, 19],
       [ 8,  9, 10, 11, 20, 21, 22, 23]])
```

In [480…
```
# Vertical stacking
w1
```

Out[480…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [482…
```
w2
```

Out[482…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [484…
```
np.vstack((w1,w2))
```

Out[484…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [486…
```
#Splitting
##Horizontal splitting
w1
```

Out[486…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [488…
```
np.hsplit(w1,2)
```

Out[488…
```
[array([[0, 1],
        [4, 5],
        [8, 9]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]])]
```

In [490…
```
np.hsplit(w1,4)
```

Out[490…
```
[array([[0],
        [4],
        [8]]),
 array([[1],
        [5],
        [9]]),
 array([[ 2],
        [ 6],
        [10]]),
 array([[ 3],
        [ 7],
        [11]])]
```

In [492…
```
#Vertical splitting
w2
```

Out[492…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [494…
```
np.vsplit(w2,3)
```

Out[494…
```
[array([[12, 13, 14, 15]]),
 array([[16, 17, 18, 19]]),
 array([[20, 21, 22, 23]])]
```

In [502…
```
#Element-wise addition

a = [ i for i in range(10000000)]
b = [i for i in range(10000000,20000000)]

c=[]

import time

start = time.time()
for i in range(len(a)):
    c.append(a[i] + b[i])

print(time.time()-start)
```

```
6.609962224960327
```

In [504…
```
#Element-wise addition
import numpy as np
a =  np.arange(10000000)
b =  np.arange(10000000,20000000)
start =  time.time()
c=a+b
print(time.time()-start)
```

```
0.2544853687286377
```

In [506…
```
2.7065064907073975 / 0.02248692512512207
```

Out[506…
```
120.35911871666826
```

In [508…
```
#Memory used for list vs numpy
##List
p = [i for i in range(10000000)]

import sys

sys.getsizeof(p)
```

Out[508…
```
89095160
```

In [510…
```
#Numpy
R = np.arange(10000000)

sys.getsizeof(R)
```

Out[510…
```
40000112
```

In [512…
```
#we can decrease more in numpy

R = np.arange(10000000,dtype = np.int16)

sys.getsizeof(R)
```

Out[512…
```
20000112
```

In [514…
```
#Normal indexing and slicing

w = np.arange(12).reshape(4,3)
w
```

Out[514…
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [518…
```
#fetching 5 from array

w[1,2]
```

Out[518…
```
5
```

In [522…
```
#fetching4,5,7,8
w[1:3]
```

Out[522…
```
array([[3, 4, 5],
       [6, 7, 8]])
```

In [524…
```
w[1:3,1:3]
```

Out[524…
```
array([[4, 5],
       [7, 8]])
```

In [526…
```
#FANCY INDEXING
w
```

Out[526…
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [528…
```
#fetch 1,3,4

w[[0,2,3]]
```

Out[528…
```
array([[ 0,  1,  2],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [532…
```
#new array
z = np.arange(24).reshape(6,4)
z
```

Out[532…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [534…
```
#fetch 1,3,4,6

z[[0,2,3,5]]
```

Out[534…
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [20, 21, 22, 23]])
```

In [536…
```
#fetch 1,3,4

z[:,[0,2,3]]
```

Out[536…
```
array([[ 0,  2,  3],
       [ 4,  6,  7],
       [ 8, 10, 11],
       [12, 14, 15],
       [16, 18, 19],
       [20, 22, 23]])
```

In [540…
```
#BOOLEAN INDEXING
G = np.random.randint(1,100,24).reshape(6,4)
G
```

Out[540…
```
array([[47, 98, 92, 73],
       [29, 20, 74, 77],
       [91, 31, 99, 92],
       [30, 23, 35, 96],
       [83, 59, 81, 20],
       [51, 46, 78, 36]])
```

In [542…
```
#find all numbers greater than 50

G > 50
```

Out[542…
```
array([[False,  True,  True,  True],
       [False, False,  True,  True],
       [ True, False,  True,  True],
       [False, False, False,  True],
       [ True,  True,  True, False],
       [ True, False,  True, False]])
```

In [544…
```
#where is true , it gives result , everything other that removed.we get value

G[G>50]
```

Out[544…
```
array([98, 92, 73, 74, 77, 91, 99, 92, 96, 83, 59, 81, 51, 78])
```

In [546…
```
#find out even number
G % 2 ==0
```

Out[546…
```
array([[False,  True,  True, False],
       [False,  True,  True, False],
       [False, False, False,  True],
       [ True, False, False,  True],
       [False, False, False,  True],
       [False,  True,  True,  True]])
```

In [548…
```
#gives only the even numbers

G [ G % 2 == 0]
```

Out[548…
```
array([98, 92, 20, 74, 92, 30, 96, 20, 46, 78, 36])
```

In [552…
```
# find all numbers greater than 50 and are even

G [(G > 50) &(G % 2 ==0)]
```

Out[552…
```
array([98, 92, 74, 92, 96, 78])
```

In [556…
```
#Find all numbers not divisible by 7

G % 7 == 0
```

Out[556…
```
array([[False,  True, False, False],
       [False, False, False,  True],
       [ True, False, False, False],
       [False, False,  True, False],
       [False, False, False, False],
       [False, False, False, False]])
```

In [562…
```
#result
G[~(G % 7 ==0)]
```

Out[562…
```
array([47, 92, 73, 29, 20, 74, 31, 99, 92, 30, 23, 96, 83, 59, 81, 20, 51,
       46, 78, 36])
```

In [566…
```
#BROADCASTING
#same shape
a = np.arange(6).reshape(2,3)
b = np.arange(6,12).reshape(2,3)

print(a)
print(b)
```

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

```
[[0 1 2]
 [3 4 5]]
[[ 6  7  8]
 [ 9 10 11]]
[[ 6  8 10]
 [12 14 16]]
```

In [568…
```
#diff shape
a = np.arange(6).reshape(2,3)
b = np.arange(3).reshape(1,3)

print(a)
print(b)

print(a+b)
```

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

In [570…
```
a = np.arange(12).reshape(4,3)
b = np.arange(3)

print(a)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

In [572…
```
print(b)
```

```
[0 1 2]
```

In [574…
```
print(a+b)
```

```
[[ 0  2  4]
 [ 3  5  7]
 [ 6  8 10]
 [ 9 11 13]]
```

In [580…
```
#could not broadcast

a = np.arange(12).reshape(3,4)
b = np.arange(3)

print(a)
print(b)

print(a+b)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[0 1 2]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[580], line 9
      6 print(a)
      7 print(b)
----> 9 print(a+b)

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

In [582…
```
a = np.arange(3).reshape(1,3)
b = np.arange(3).reshape(3,1)
print(a)
print(b)
print(a+b)
```

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

In [584…
```
a = np.arange(3).reshape(1,3)
b = np.arange(4).reshape(4,1)
print(a)
print(b)
print(a+b)
```

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

In [586…
```
a = np.array([1])
b = np.arange(4).reshape(2,2)
print(a)
print(b)
print(a+b)
```

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

In [588…
```
#doesn't work
a = np.arange(12).reshape(3,4)
b = np.arange(12).reshape(4,3)
print(a)
print(b)
print(a+b)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[588], line 6
      4 print(a)
      5 print(b)
----> 6 print(a+b)

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

In [590…
```
#not work
a = np.arange(16).reshape(4,4)
b = np.arange(4).reshape(2,2)
print(a)
print(b)
print(a+b)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[[0 1]
 [2 3]]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[590], line 6
      4 print(a)
      5 print(b)
----> 6 print(a+b)

ValueError: operands could not be broadcast together with shapes (4,4) (2,2)
```

In [592…
```
#working with mathematical formaulas
k = np.arange(10)
```

In [594…
```
k
```

Out[594…
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [596…
```
np.sum(k)
```

Out[596…
```
45
```

In [598…
```
np.sin(k)
```

Out[598…
```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849])
```

In [600…
```
def signoid(array):
    return 1/(1+np.exp(-(array)))
k = np.arange(10)
signoid(k)
```

Out[600…
```
array([0.5       , 0.73105858, 0.88079708, 0.95257413, 0.98201379,
       0.99330715, 0.99752738, 0.99908895, 0.99966465, 0.99987661])
```

In [604…
```
k = np.arange(100)
signoid(k)
```

Out[604…
```
array([0.5       , 0.73105858, 0.88079708, 0.95257413, 0.98201379,
       0.99330715, 0.99752738, 0.99908895, 0.99966465, 0.99987661,
       0.9999546 , 0.9999833 , 0.99999386, 0.99999774, 0.99999917,
       0.99999969, 0.99999989, 0.99999996, 0.99999998, 0.99999999,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ,
       1.        , 1.        , 1.        , 1.        , 1.        ])
```

In [606…
```
actual = np.random.randint(1,50,25)
predicted = np.random.randint(1,50,25)
```

In [608…
```
actual
```

Out[608…
```
array([49,  1, 43, 11, 36, 25,  9, 48, 34, 26, 42, 19, 32, 25, 21, 20, 40,
       38,  7, 33, 38, 33,  4, 17, 24])
```

In [610…
```
predicted
```

Out[610…
```
array([22,  8, 40, 19, 39, 29, 45,  6, 12, 24, 29, 17, 28, 31, 30, 35, 41,
       29, 36,  5, 28, 20, 16, 13,  2])
```

In [614…
```
def mse(actual,predicted):
    return np.mean((actual-predicted)**2)

mse(actual,predicted)
```

Out[614…
```
303.0
```

In [616…
```
#detailed

actual-predicted
```

Out[616…
```
array([ 27,  -7,   3,  -8,  -3,  -4, -36,  42,  22,   2,  13,   2,   4,
        -6,  -9, -15,  -1,   9, -29,  28,  10,  13, -12,   4,  22])
```

In [618…
```
(actual-predicted)**2
```

Out[618…
```
array([ 729,   49,    9,   64,    9,   16, 1296, 1764,  484,    4,  169,
          4,   16,   36,   81,  225,    1,   81,  841,  784,  100,  169,
        144,   16,  484])
```

In [620…
```
np.mean((actual-predicted)**2)
```

Out[620…
```
303.0
```

In [626…
```
#working with missing values
#working with missing values-> np.nan

s=np.array([1,2,3,4,np.nan,6])
s
```

Out[626…
```
array([ 1.,  2.,  3.,  4., nan,  6.])
```

In [630…
```
np.isnan(s)
```

Out[630…
```
array([False, False, False, False,  True, False])
```

In [638…
```
s[np.isnan(s)]
```

Out[638…
```
array([nan])
```

In [640…
```
s[~np.isnan(s)]
```

Out[640…
```
array([1., 2., 3., 4., 6.])
```

In [644…
```
    #PLOTTING GRAPHS
#plotting a 2d plot
#x = y
x=np.linspace(-10,10,100)
x
```

Out[644…
```
array([-10.        ,  -9.7979798 ,  -9.5959596 ,  -9.39393939,
        -9.19191919,  -8.98989899,  -8.78787879,  -8.58585859,
        -8.38383838,  -8.18181818,  -7.97979798,  -7.77777778,
        -7.57575758,  -7.37373737,  -7.17171717,  -6.96969697,
        -6.76767677,  -6.56565657,  -6.36363636,  -6.16161616,
        -5.95959596,  -5.75757576,  -5.55555556,  -5.35353535,
        -5.15151515,  -4.94949495,  -4.74747475,  -4.54545455,
        -4.34343434,  -4.14141414,  -3.93939394,  -3.73737374,
        -3.53535354,  -3.33333333,  -3.13131313,  -2.92929293,
        -2.72727273,  -2.52525253,  -2.32323232,  -2.12121212,
        -1.91919192,  -1.71717172,  -1.51515152,  -1.31313131,
        -1.11111111,  -0.90909091,  -0.70707071,  -0.50505051,
        -0.3030303 ,  -0.1010101 ,   0.1010101 ,   0.3030303 ,
         0.50505051,   0.70707071,   0.90909091,   1.11111111,
         1.31313131,   1.51515152,   1.71717172,   1.91919192,
         2.12121212,   2.32323232,   2.52525253,   2.72727273,
         2.92929293,   3.13131313,   3.33333333,   3.53535354,
         3.73737374,   3.93939394,   4.14141414,   4.34343434,
         4.54545455,   4.74747475,   4.94949495,   5.15151515,
         5.35353535,   5.55555556,   5.75757576,   5.95959596,
         6.16161616,   6.36363636,   6.56565657,   6.76767677,
         6.96969697,   7.17171717,   7.37373737,   7.57575758,
         7.77777778,   7.97979798,   8.18181818,   8.38383838,
         8.58585859,   8.78787879,   8.98989899,   9.19191919,
         9.39393939,   9.5959596 ,   9.7979798 ,  10.        ])
```

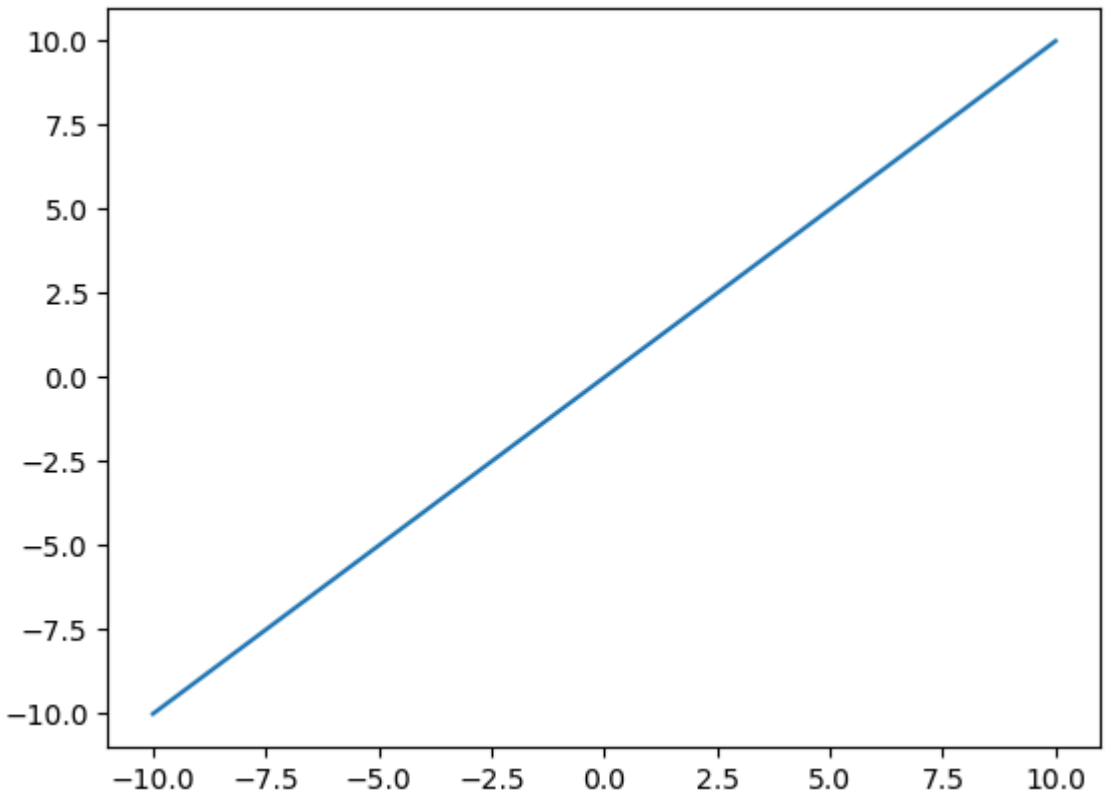
In [646…
```
y=x
```

In [648…
```
y
```

Out[648…
```
array([-10.        ,  -9.7979798 ,  -9.5959596 ,  -9.39393939,
        -9.19191919,  -8.98989899,  -8.78787879,  -8.58585859,
        -8.38383838,  -8.18181818,  -7.97979798,  -7.77777778,
        -7.57575758,  -7.37373737,  -7.17171717,  -6.96969697,
        -6.76767677,  -6.56565657,  -6.36363636,  -6.16161616,
        -5.95959596,  -5.75757576,  -5.55555556,  -5.35353535,
        -5.15151515,  -4.94949495,  -4.74747475,  -4.54545455,
        -4.34343434,  -4.14141414,  -3.93939394,  -3.73737374,
        -3.53535354,  -3.33333333,  -3.13131313,  -2.92929293,
        -2.72727273,  -2.52525253,  -2.32323232,  -2.12121212,
        -1.91919192,  -1.71717172,  -1.51515152,  -1.31313131,
        -1.11111111,  -0.90909091,  -0.70707071,  -0.50505051,
        -0.3030303 ,  -0.1010101 ,   0.1010101 ,   0.3030303 ,
         0.50505051,   0.70707071,   0.90909091,   1.11111111,
         1.31313131,   1.51515152,   1.71717172,   1.91919192,
         2.12121212,   2.32323232,   2.52525253,   2.72727273,
         2.92929293,   3.13131313,   3.33333333,   3.53535354,
         3.73737374,   3.93939394,   4.14141414,   4.34343434,
         4.54545455,   4.74747475,   4.94949495,   5.15151515,
         5.35353535,   5.55555556,   5.75757576,   5.95959596,
         6.16161616,   6.36363636,   6.56565657,   6.76767677,
         6.96969697,   7.17171717,   7.37373737,   7.57575758,
         7.77777778,   7.97979798,   8.18181818,   8.38383838,
         8.58585859,   8.78787879,   8.98989899,   9.19191919,
         9.39393939,   9.5959596 ,   9.7979798 ,  10.        ])
```

In [650…
```
import matplotlib.pyplot as plt

plt.plot(x,y)
```

Out[650…
```
[<matplotlib.lines.Line2D at 0x21c082dcb30>]
```

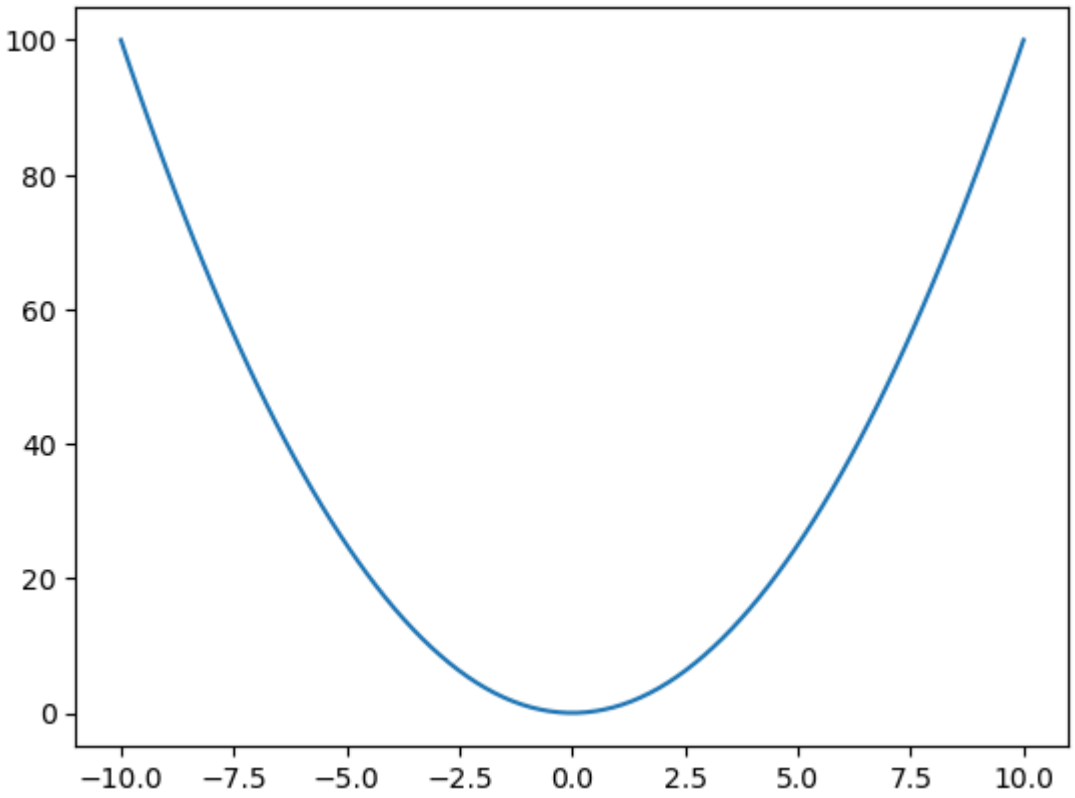


In [652…

```
#y =x^2

x = np.linspace(-10,10,100)
y = x**2

plt.plot(x,y)
```

Out[652…

```
[<matplotlib.lines.Line2D at 0x21c08b17560>]
```

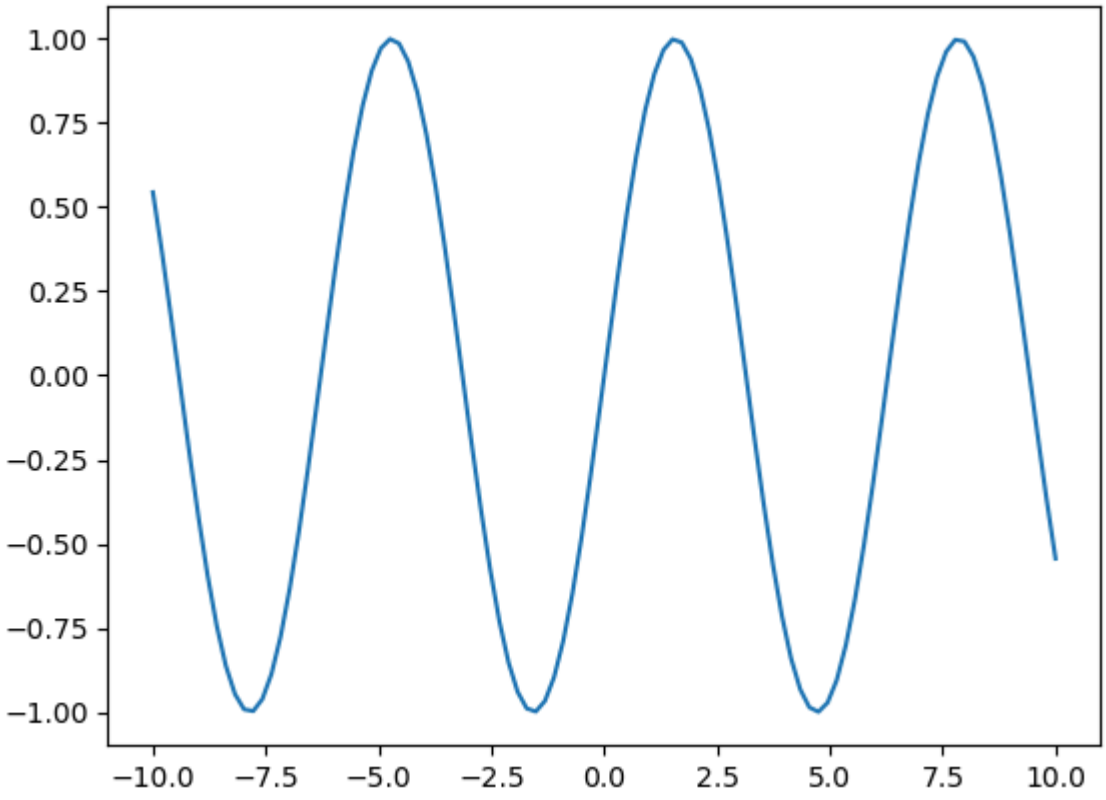


In [654…

```
#y = sin(x)

x = np.linspace(-10,10,100)
y = np.sin(x)

plt.plot(x,y)
```

Out[654…

```
[<matplotlib.lines.Line2D at 0x21c08345d60>]
```



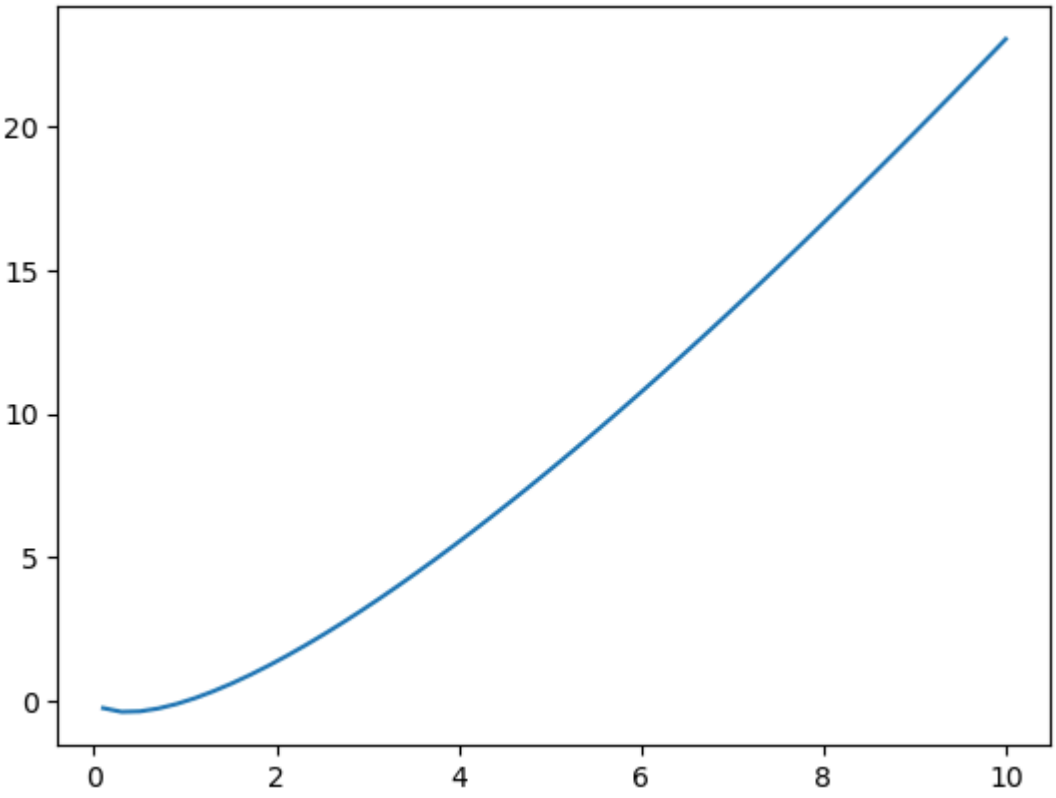
In [656…

```
#y = xlog(x)
x = np.linspace(-10,10,100)
y = x * np.log(x)
plt.plot(x,y)
```

```
C:\Users\arpan\AppData\Local\Temp\ipykernel_20100\3633574392.py:3: RuntimeWarnin
g: invalid value encountered in log
  y = x * np.log(x)
```

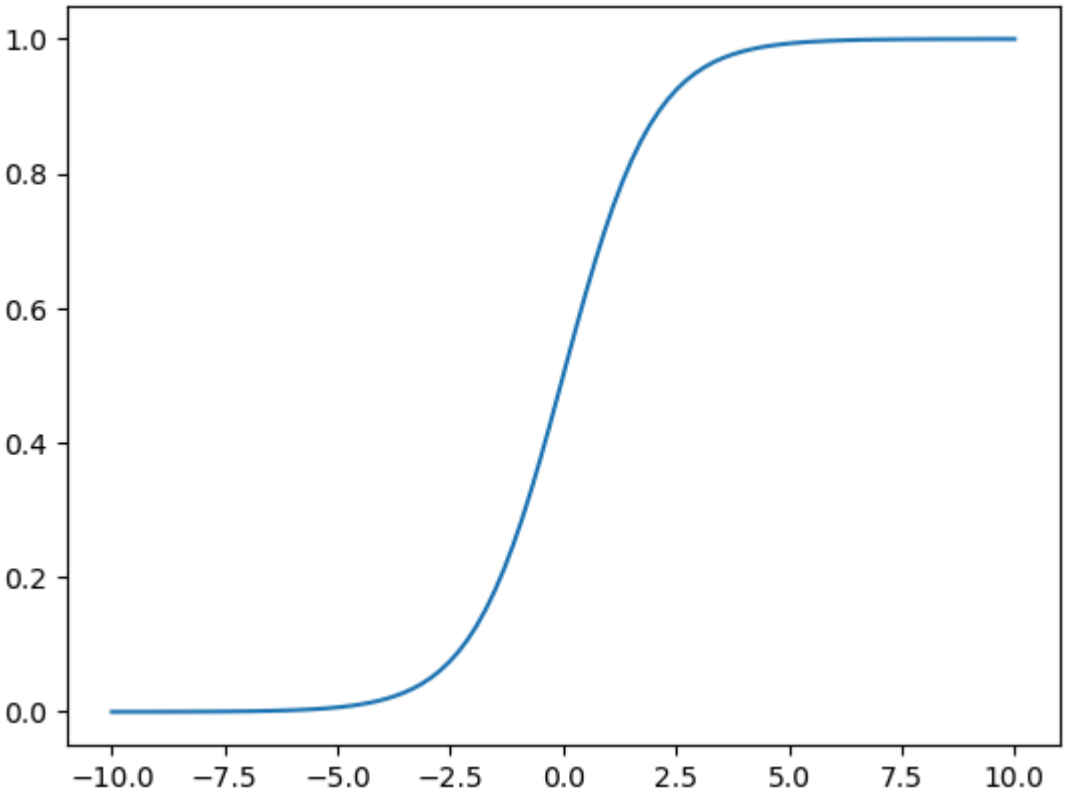
Out[656…

```
[<matplotlib.lines.Line2D at 0x21c083c99a0>]
```



In [658…
```
#signoid
x = np.linspace(-10,10,100)
y = 1/(1+np.exp(-x))
plt.plot(x,y)
```

Out[658…
```
[<matplotlib.lines.Line2D at 0x21c0843ce90>]
```

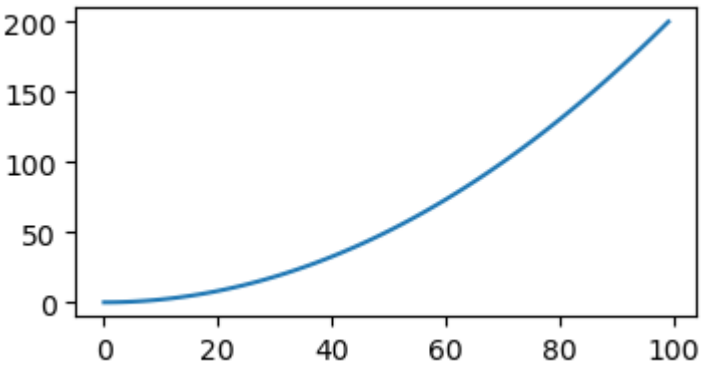


In [662…
```
# MEDHGRID
import numpy as np
import matplotlib.pyplot as plt
```

In [664…
```
x = np.linspace(0,10,100)
y = np.linspace(0,10,100)
```

In [668…
```
f = x**2+y**2
```

In [670…
```
plt.figure(figsize=(4,2))
plt.plot(f)
plt.show()
```



In [672…
```
x = np.arange(3)
y = np.arange(3)
```

In [674…
```
x
```

Out[674…
```
array([0, 1, 2])
```

In [676…
```
y
```

Out[676…
```
array([0, 1, 2])
```

In [684…
```
#Generating a meshgrid

xv ,yv = np.meshgrid(x,y)
```

In [686…
```
xv
```

Out[686…
```
array([[0, 1, 2],
       [0, 1, 2],
       [0, 1, 2]])
```

In [688…
```
yv
```

Out[688…
```
array([[0, 0, 0],
       [1, 1, 1],
       [2, 2, 2]])
```

In [690…
```
p = np.linspace(-4,4,9)
v = np.linspace(-5,5,11)
print(p)
print(v)
```

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

In [692…
```
p_1,v_1 = np. meshgrid(p,v)
```

In [694…
```
print(p_1)
```

```
[[-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]
 [-4. -3. -2. -1.  0.  1.  2.  3.  4.]]
```

In [696…
```
print(v_1)
```

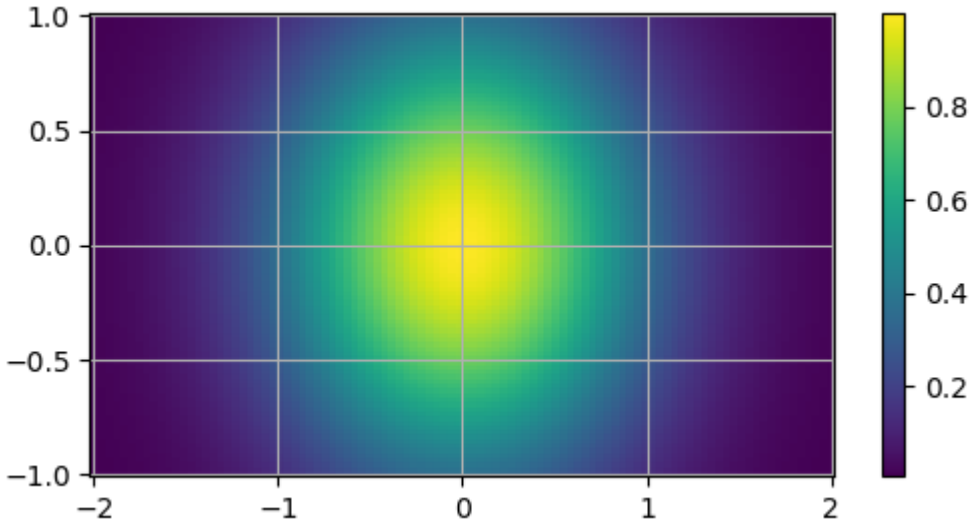
```
[[-5. -5. -5. -5. -5. -5. -5. -5. -5.]
 [-4. -4. -4. -4. -4. -4. -4. -4. -4.]
 [-3. -3. -3. -3. -3. -3. -3. -3. -3.]
 [-2. -2. -2. -2. -2. -2. -2. -2. -2.]
 [-1. -1. -1. -1. -1. -1. -1. -1. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  1.  1.  1.  1.  1.  1.  1.  1.]
 [ 2.  2.  2.  2.  2.  2.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

In [698…
```
xv**2 + y**2
```

Out[698…
```
array([[0, 2, 8],
       [0, 2, 8],
       [0, 2, 8]])
```

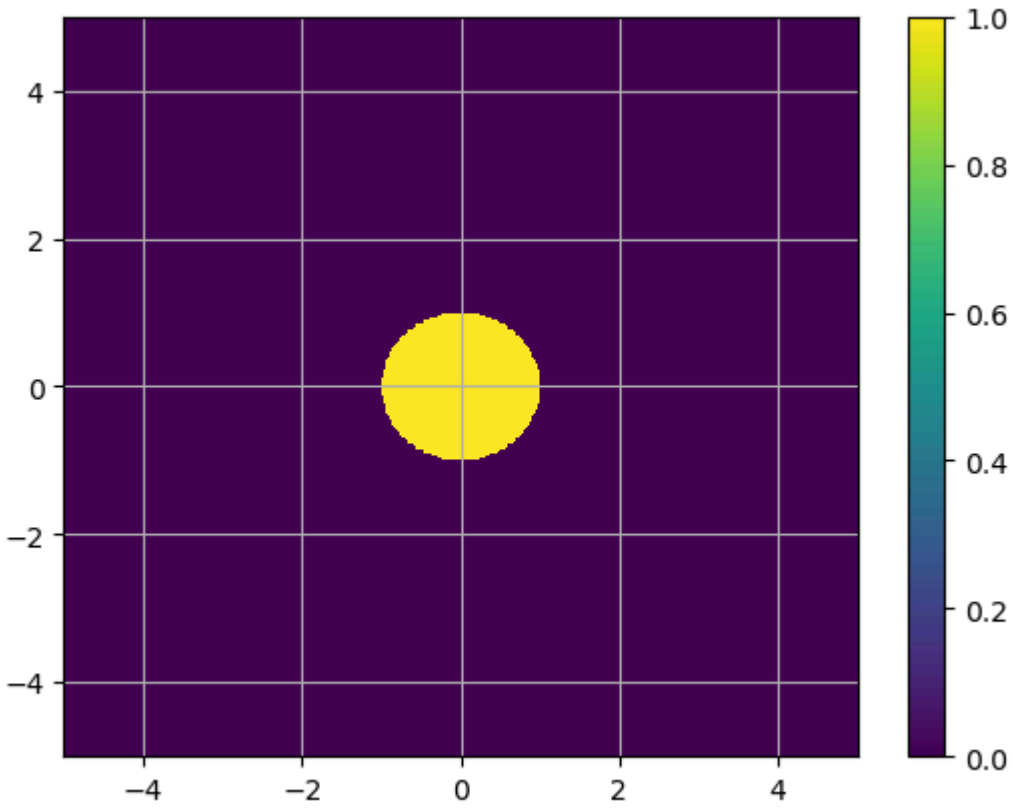
In [700…
```
x = np.linspace(-2,2,100)
y = np.linspace(-1,1,100)
xv,yv = np.meshgrid(x,y)
f = np.exp(-xv**2-yv**2)
```

In [704…
```
plt.figure(figsize=(6,3))
plt.pcolormesh(xv,yv ,f ,shading='auto')
plt.colorbar()
plt.grid()
plt.show()
```



In [708…
```
import numpy as np
import matplotlib.pyplot as plt

def f(x,y):
    return np.where((x**2 + y**2 < 1), 1.0,0.0)

x = np.linspace(-5,5,500)
y = np.linspace(-5,5,500)
xv,yv = np.meshgrid(x,y)
rectangular_mask = f(xv,yv)

plt.pcolormesh(xv,yv,rectangular_mask,shading='auto')
plt.colorbar()
plt.grid()
```

```
plt.show()
```

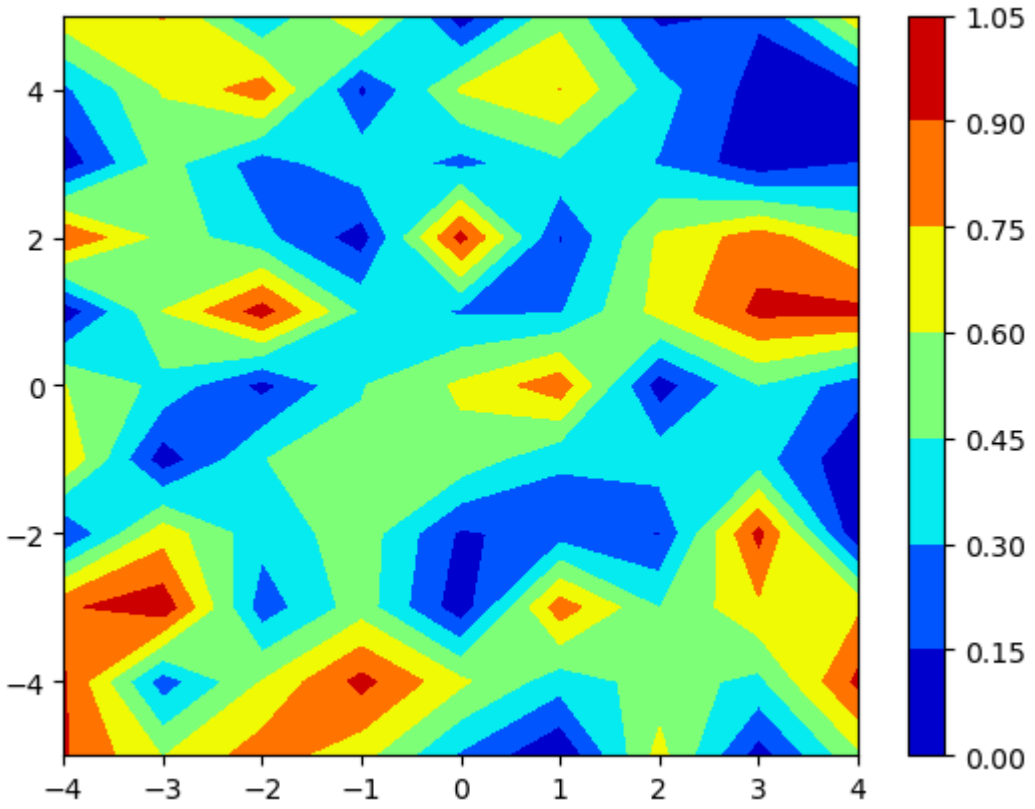


In [710…
```
x = np.linspace(-4,4,9)
```

In [712…
```
y = np.linspace(-5,5,11)
```

In [714…
```
x_1,y_1 = np.meshgrid(x,y)
```

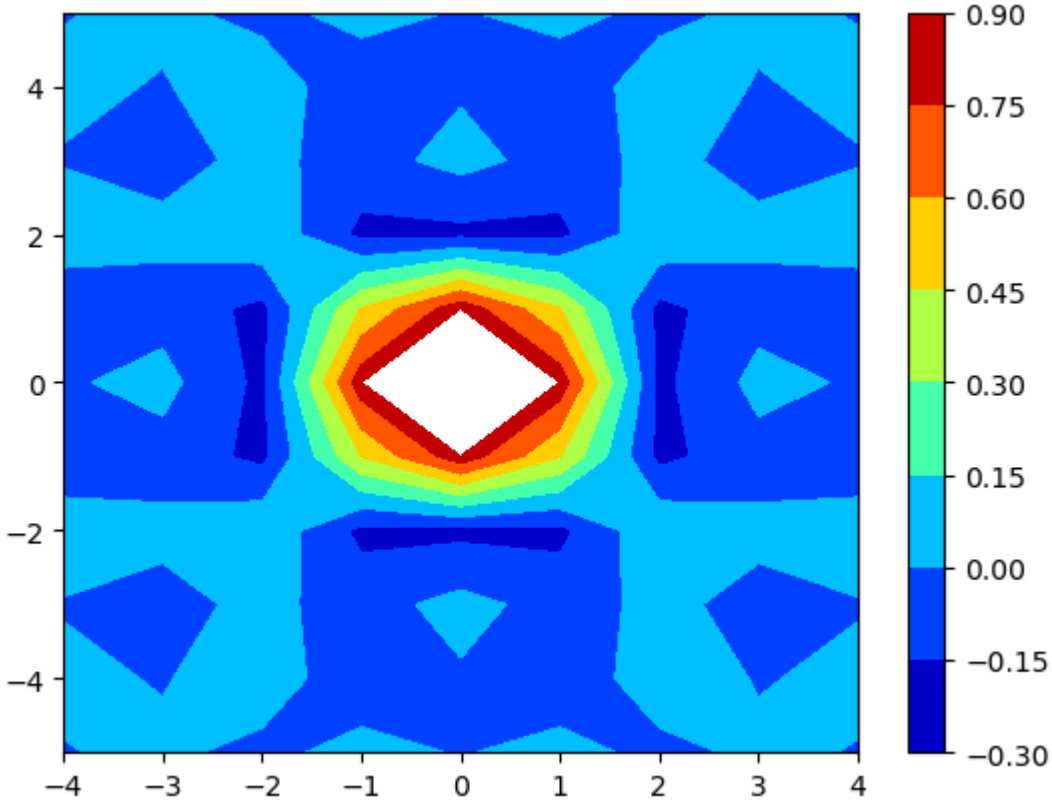
In [716…
```
random_data = np.random.random((11,9))
plt.contourf(x_1,y_1,random_data,cmap = 'jet')

plt.colorbar()
plt.show()
```



In [718…

```
sine = (np.sin(x_1**2 + y_1**2))/(x_1**2 + y_1**2)
plt.contourf(x_1, y_1, sine, cmap = 'jet')
plt.colorbar()
plt.show()
```

```
C:\Users\arpan\AppData\Local\Temp\ipykernel_20100\824154955.py:1: RuntimeWarning:
invalid value encountered in divide
  sine = (np.sin(x_1**2 + y_1**2))/(x_1**2 + y_1**2)
```



In [722…
```
x_1,y_1 = np.meshgrid(x,y,sparse = True)
```

In [724…
```
x_1
```

Out[724…
```
array([[-4., -3., -2., -1.,  0.,  1.,  2.,  3.,  4.]])
```

In [726…
```
y_1
```

Out[726…
```
array([[-5.],
       [-4.],
       [-3.],
       [-2.],
       [-1.],
       [ 0.],
       [ 1.],
       [ 2.],
       [ 3.],
       [ 4.],
       [ 5.]])
```

In [728…
```
a = np.random.randint(1,100,15)
a
```

Out[728…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [732…
```
b = np.random.randint(1,100,24).reshape(6,4)
b
```

Out[732…
```
array([[ 4, 73, 75, 13],
       [74, 28, 15, 93],
       [95, 94, 48,  6],
       [90, 34, 58, 94],
       [ 4, 53, 94, 48],
       [82, 62, 47, 24]])
```

In [734…
```
np.sort(a)
```

Out[734…
```
array([ 3,  4,  5, 28, 31, 31, 46, 49, 53, 69, 69, 77, 84, 98, 99])
```

In [736…
```
np.sort(a)[::-1]
```

Out[736…
```
array([99, 98, 84, 77, 69, 69, 53, 49, 46, 31, 31, 28,  5,  4,  3])
```

In [738…
```
np.sort(b)
```

Out[738…
```
array([[ 4, 13, 73, 75],
       [15, 28, 74, 93],
       [ 6, 48, 94, 95],
       [34, 58, 90, 94],
       [ 4, 48, 53, 94],
       [24, 47, 62, 82]])
```

In [740…
```
np.sort(b,axis = 0)
```

Out[740…
```
array([[ 4, 28, 15,  6],
       [ 4, 34, 47, 13],
       [74, 53, 48, 24],
       [82, 62, 58, 48],
       [90, 73, 75, 93],
       [95, 94, 94, 94]])
```

In [742…
```
#np.append
#code
a
```

Out[742…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [744…
```
np.append(a,200)
```

Out[744…
```
array([ 46,   3,  84,  28,  49,  69,   5,  31,  77,  31,  69,  99,  98,
         4,  53, 200])
```

In [746…
```
b
```

Out[746…
```
array([[ 4, 73, 75, 13],
       [74, 28, 15, 93],
       [95, 94, 48,  6],
       [90, 34, 58, 94],
       [ 4, 53, 94, 48],
       [82, 62, 47, 24]])
```

In [748…
```
np.append(b,np.ones((b.shape[0],1)))
```

Out[748…
```
array([ 4., 73., 75., 13., 74., 28., 15., 93., 95., 94., 48.,  6., 90.,
       34., 58., 94.,  4., 53., 94., 48., 82., 62., 47., 24.,  1.,  1.,
        1.,  1.,  1.,  1.])
```

In [750…
```
np.append(b,np.ones((b.shape[0],1)),axis=1)
```

Out[750…
```
array([[ 4., 73., 75., 13.,  1.],
       [74., 28., 15., 93.,  1.],
       [95., 94., 48.,  6.,  1.],
       [90., 34., 58., 94.,  1.],
       [ 4., 53., 94., 48.,  1.],
       [82., 62., 47., 24.,  1.]])
```

In [752…
```
np.append(b,np.random.random((b.shape[0],1)),axis=1)
```

Out[752…
```
array([[4.00000000e+00, 7.30000000e+01, 7.50000000e+01, 1.30000000e+01,
        9.56861707e-01],
       [7.40000000e+01, 2.80000000e+01, 1.50000000e+01, 9.30000000e+01,
        1.05448497e-01],
       [9.50000000e+01, 9.40000000e+01, 4.80000000e+01, 6.00000000e+00,
        8.29747630e-01],
       [9.00000000e+01, 3.40000000e+01, 5.80000000e+01, 9.40000000e+01,
        1.24766102e-02],
       [4.00000000e+00, 5.30000000e+01, 9.40000000e+01, 4.80000000e+01,
        2.41801280e-01],
       [8.20000000e+01, 6.20000000e+01, 4.70000000e+01, 2.40000000e+01,
        9.38894725e-01]])
```

In [756…
```
c = np.arange(6).reshape(2,3)
d =np.arange(6,12).reshape(2,3)
```

In [761…
```
c
```

Out[761…
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [759…
```
d
```

Out[759…
```
array([[ 6,  7,  8],
       [ 9, 10, 11]])
```

In [763…
```
np.concatenate((c,d))
```

Out[763…
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [765…
```
np.concatenate((c,d),axis = 1)
```

Out[765…
```
array([[ 0,  1,  2,  6,  7,  8],
       [ 3,  4,  5,  9, 10, 11]])
```

In [767…
```
e = np.array([1,1,2,2,3,3,4,4,5,5,6,6,])
```

In [769…
```
np.unique(e)
```

Out[769…
```
array([1, 2, 3, 4, 5, 6])
```

In [771…
```
a
```

Out[771…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [773…
```
a.shape
```

Out[773…
```
(15,)
```

In [775…
```
np.expand_dims(a,axis = 0)
```

Out[775…
```
array([[46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53]])
```

In [777…
```
np.expand_dims(a,axis = 0).shape
```

Out[777…
```
(1, 15)
```

In [781…
```
np.expand_dims(a,axis = 1)
```

Out[781…
```
array([[46],
       [ 3],
       [84],
       [28],
       [49],
       [69],
       [ 5],
       [31],
       [77],
       [31],
       [69],
       [99],
       [98],
       [ 4],
       [53]])
```

In [783…
```
np.expand_dims(a,axis = 1).shape
```

Out[783…
```
(15, 1)
```

In [785…
```
#np.where
a
```

Out[785…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [787…
```
np.where(a>50)
```

Out[787…
```
(array([ 2,  5,  8, 10, 11, 12, 14], dtype=int64),)
```

In [789…
```
np.where(a%2 == 0,0,a)
```

Out[789…
```
array([ 0,  3,  0,  0, 49, 69,  5, 31, 77, 31, 69, 99,  0,  0, 53])
```

In [791…
```
#np.argmax
a
```

Out[791…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [793…
```
np.argmax(a)
```

Out[793…
```
11
```

In [795…
```
b
```

Out[795…
```
array([[ 4, 73, 75, 13],
       [74, 28, 15, 93],
       [95, 94, 48,  6],
       [90, 34, 58, 94],
       [ 4, 53, 94, 48],
       [82, 62, 47, 24]])
```

In [797…
```
np.argmax(b,axis =1)
```

Out[797…
```
array([2, 3, 0, 3, 2, 0], dtype=int64)
```

In [799…
```
np.argmax(b,axis =0)
```

Out[799… array([2, 2, 4, 3], dtype=int64)

In [801…
```
#np.argmin
a
```

Out[801… array([46, 3, 84, 28, 49, 69, 5, 31, 77, 31, 69, 99, 98, 4, 53])

In [803…
```
np.argmin(a)
```

Out[803… 1

In [805…
```
#np.cumsum
a
```

Out[805… array([46, 3, 84, 28, 49, 69, 5, 31, 77, 31, 69, 99, 98, 4, 53])

In [807…
```
np.cumsum(a)
```

Out[807…
```
array([ 46,  49, 133, 161, 210, 279, 284, 315, 392, 423, 492, 591, 689,
       693, 746])
```

In [809…
```
b
```

Out[809…
```
array([[ 4, 73, 75, 13],
       [74, 28, 15, 93],
       [95, 94, 48,  6],
       [90, 34, 58, 94],
       [ 4, 53, 94, 48],
       [82, 62, 47, 24]])
```

In [811…
```
np.cumsum(b)
```

Out[811…
```
array([   4,   77,  152,  165,  239,  267,  282,  375,  470,  564,  612,
        618,  708,  742,  800,  894,  898,  951, 1045, 1093, 1175, 1237,
       1284, 1308])
```

In [813…
```
np.cumsum(b,axis=1)
```

Out[813…
```
array([[  4,  77, 152, 165],
       [ 74, 102, 117, 210],
       [ 95, 189, 237, 243],
       [ 90, 124, 182, 276],
       [  4,  57, 151, 199],
       [ 82, 144, 191, 215]])
```

In [815…
```
np.cumsum(b,axis=0)
```

Out[815…
```
array([[  4,  73,  75,  13],
       [ 78, 101,  90, 106],
       [173, 195, 138, 112],
       [263, 229, 196, 206],
       [267, 282, 290, 254],
       [349, 344, 337, 278]])
```

In [817…
```
#np.cumsum-->multiply
a
```

Out[817… array([46, 3, 84, 28, 49, 69, 5, 31, 77, 31, 69, 99, 98, 4, 53])

In [819…
```
np.cumsum(a)
```

Out[819…
```
array([ 46,  49, 133, 161, 210, 279, 284, 315, 392, 423, 492, 591, 689,
       693, 746])
```

In [821…
```
#np.percentile
a
```

Out[821…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [823…
```
np.percentile(a,100)
```

Out[823…
```
99.0
```

In [825…
```
np.percentile(a,0)
```

Out[825…
```
3.0
```

In [827…
```
np.percentile(a,50)
```

Out[827…
```
49.0
```

In [829…
```
np.median(a)
```

Out[829…
```
49.0
```

In [831…
```
#np.histogram
a
```

Out[831…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [833…
```
np.histogram(a, bins=[10,20,30,40,50,60,70,80,90,100])
```

Out[833…
```
(array([0, 1, 2, 2, 1, 2, 1, 1, 2], dtype=int64),
 array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100]))
```

In [837…
```
np.histogram(a, bins= [0,50,100])
```

Out[837…
```
(array([8, 7], dtype=int64), array([  0,  50, 100]))
```

In [957…
```
#np.corr coef
salary = np.array([20000,25000,35000,60000])
experience = np.array([1,3,2,4,2])
```

In [841…
```
salary
```

Out[841…
```
array([20000, 25000, 35000, 60000])
```

In [843…
```
experience
```

Out[843…
```
array([1, 3, 2, 4, 2])
```

In [955…
```
np.corrcoef(salary,experience)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[955], line 1
----> 1 np.corrcoef(salary,experience)

File ~\anaconda3\Lib\site-packages\numpy\lib\function_base.py:2889, in corrcoef
(x, y, rowvar, bias, ddof, dtype)
   2885 if bias is not np._NoValue or ddof is not np._NoValue:
   2886     # 2015-03-15, 1.10
   2887     warnings.warn('bias and ddof have no effect and are deprecated',
   2888                   DeprecationWarning, stacklevel=2)
-> 2889 c = cov(x, y, rowvar, dtype=dtype)
   2890 try:
   2891     d = diag(c)

File ~\anaconda3\Lib\site-packages\numpy\lib\function_base.py:2683, in cov(m, y,
rowvar, bias, ddof, fweights, aweights, dtype)
   2681     if not rowvar and y.shape[0] != 1:
   2682         y = y.T
-> 2683     X = np.concatenate((X, y), axis=0)
   2685 if ddof is None:
   2686     if bias == 0:

ValueError: all the input array dimensions except for the concatenation axis must
match exactly, but along dimension 1, the array at index 0 has size 4 and the arr
ay at index 1 has size 5
```

In [855…
```
#np.isin
a
```

Out[855…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [863…
```
items=([46,53,14,44,33,39,76,60,68,12,87,66,74,10,98])
np.isin(a,items)
```

Out[863…
```
array([ True, False, False, False, False, False, False, False, False,
       False, False, False,  True, False,  True])
```

In [867…
```
a[np.isin(a,items)]
```

Out[867…
```
array([46, 98, 53])
```

In [869…
```
#np.flip
a
```

Out[869…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [871…
```
np.flip(a)
```

Out[871…
```
array([53,  4, 98, 99, 69, 31, 77, 31,  5, 69, 49, 28, 84,  3, 46])
```

In [873…
```
b
```

Out[873…
```
array([[ 4, 73, 75, 13],
       [74, 28, 15, 93],
       [95, 94, 48,  6],
       [90, 34, 58, 94],
       [ 4, 53, 94, 48],
       [82, 62, 47, 24]])
```

In [875…
```
np.flip(b)
```

Out[875…
```
array([[24, 47, 62, 82],
       [48, 94, 53,  4],
       [94, 58, 34, 90],
       [ 6, 48, 94, 95],
       [93, 15, 28, 74],
       [13, 75, 73,  4]])
```

In [877…
```
np.flip(b,axis = 1)
```

Out[877…
```
array([[13, 75, 73,  4],
       [93, 15, 28, 74],
       [ 6, 48, 94, 95],
       [94, 58, 34, 90],
       [48, 94, 53,  4],
       [24, 47, 62, 82]])
```

In [879…
```
np.flip(b,axis = 0)
```

Out[879…
```
array([[82, 62, 47, 24],
       [ 4, 53, 94, 48],
       [90, 34, 58, 94],
       [95, 94, 48,  6],
       [74, 28, 15, 93],
       [ 4, 73, 75, 13]])
```

In [881…
```
#np.put
a
```

Out[881…
```
array([46,  3, 84, 28, 49, 69,  5, 31, 77, 31, 69, 99, 98,  4, 53])
```

In [883…
```
np.put(a,[0,1],[110,530])
```

In [885…
```
a
```

Out[885…
```
array([110, 530,  84,  28,  49,  69,   5,  31,  77,  31,  69,  99,  98,
         4,  53])
```

In [887…
```
#np.delete
a
```

Out[887…
```
array([110, 530,  84,  28,  49,  69,   5,  31,  77,  31,  69,  99,  98,
         4,  53])
```

In [889…
```
np.delete(a,0)
```

Out[889…
```
array([530,  84,  28,  49,  69,   5,  31,  77,  31,  69,  99,  98,   4,
        53])
```

In [891…
```
np.delete(a,[0,2,4])
```

Out[891… array([530, 28, 69, 5, 31, 77, 31, 69, 99, 98, 4, 53])

In [893…
```
#set function
m = np.array([1,2,3,4,5])
n = np.array([3,4,5,6,7])
```

In [901…
```
#union
np.union1d(m,n)
```

Out[901… array([1, 2, 3, 4, 5, 6, 7])

In [907…
```
#intersection
np.intersect1d(m,n)
```

Out[907… array([3, 4, 5])

In [909…
```
#set difference
np.setdiff1d(m,n)
```

Out[909… array([1, 2])

In [911…
```
np.setdiff1d(n,m)
```

Out[911… array([6, 7])

In [929…
```
#set xor
np.setxor1d(m,n)
```

Out[929… array([1, 2, 6, 7])

In [927…
```
#in 1d
np.in1d(m,1)
```

Out[927… array([True, False, False, False, False])

In [931…
```
m[np.in1d(m,1)]
```

Out[931… array([1])

In [933…
```
np.in1d(m,10)
```

Out[933… array([False, False, False, False, False])

In [935…
```
#np.clip
a
```

Out[935… array([110, 530, 84, 28, 49, 69, 5, 31, 77, 31, 69, 99, 98,
4, 53])

In [939…
```
np.clip(a,a_min=15,a_max=50)
```

Out[939… array([50, 50, 50, 28, 49, 50, 15, 31, 50, 31, 50, 50, 50, 15, 50])

In [941…
```
#np.swapaxes
arr = np.array([[1,2,3], [4,5,6]])
```

```
swapped_arr = np.swapaxes(arr,0,1)
```

In [943…
```
arr
```

Out[943…
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [945…
```
swapped_arr
```

Out[945…
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

In [947…
```
print("original array:")
print(arr)
```

```
original array:
[[1 2 3]
 [4 5 6]]
```

In [949…
```
print("swapped array:")
print(swapped_arr)
```

```
swapped array:
[[1 4]
 [2 5]
 [3 6]]
```

In [ ]:
```

```