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

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

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

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

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

```
In [4]: type(a)
```

```
Out[4]: numpy.ndarray
```

```
In [5]: new=([[45,23,22,2],[24,55,3,22]])
        print(new)
```

```
[[45, 23, 22, 2], [24, 55, 3, 22]]
```

dtype

```
In [6]: np.array([2,4,56,422,32,1],dtype=float)
```

```
Out[6]: array([  2.,   4.,  56., 422.,  32.,   1.])
```

```
In [7]: np.array([2,4,56,422,32,1],dtype=bool)
```

```
Out[7]: array([ True,  True,  True,  True,  True,  True])
```

```
In [8]: np.array([2,4,56,422,32,1],dtype=complex)
```

```
Out[8]: array([  2.+0.j,   4.+0.j,  56.+0.j, 422.+0.j,  32.+0.j,   1.+0.j])
```

arange

In [9]:
```
np.arange(1,25)
```

Out[9]:
```
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 [10]:
```
np.arange(1,25,2)
```

Out[10]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23])
```

reshape

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

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

In [12]:
```
np.arange(1,11).reshape(2,5)
```

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

ones&zeros

In [13]:
```
np.ones((3,4))
```

Out[13]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [14]:
```
np.zeros((3,4))
```

Out[14]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [15]:
```
np.random.rand(4,3)
```

```
Out[15]: array([[0.06416071, 0.57864013, 0.30325031],
                [0.88331716, 0.78160209, 0.42597212],
                [0.59228362, 0.05901777, 0.73569039],
                [0.2904465 , 0.38693834, 0.59067639]])
```

linspace

```
In [16]: np.linspace(-10,10,10,dtype=int)
```

```
Out[16]: array([-10,  -8,  -6,  -4,  -2,   1,   3,   5,   7,  10])
```

```
In [17]: np.linspace(-2,12,6,dtype=int)
```

```
Out[17]: array([-2,  0,  3,  6,  9, 12])
```

identity

diagonally 1's

```
In [18]: np.identity(3)
```

```
Out[18]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

```
In [19]: np.identity(6)
```

```
Out[19]: 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.]])
```

array attribute

```
In [20]: a1=np.arange(10)
         a1
```

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

```
In [21]: a2=np.arange(12, dtype =float).reshape(3,4) #3x4 matrix
         a2
```

```
Out[21]: array([[ 0.,  1.,  2.,  3.],
                [ 4.,  5.,  6.,  7.],
                [ 8.,  9., 10., 11.]])
```

```
In [22]: a3=np.arange(8).reshape(2,2,2)
         a3
```

```
Out[22]: array([[[0, 1],
                 [2, 3]],

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

ndim()

notify the dimension

```
In [23]: a1.ndim
```

```
Out[23]: 1
```

```
In [24]: a2.ndim
```

```
Out[24]: 2
```

```
In [25]: a3.ndim
```

```
Out[25]: 3
```

shape()

```
In [26]: a1.shape

Out[26]: (10,)

In [27]: a2.shape

Out[27]: (3, 4)

In [28]: a3.shape

Out[28]: (2, 2, 2)
```

size()

```
In [29]: a3

Out[29]: array([[[0, 1],
                 [2, 3]],

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

In [30]: a3.size

Out[30]: 8

In [31]: a2

Out[31]: array([[ 0.,  1.,  2.,  3.],
                [ 4.,  5.,  6.,  7.],
                [ 8.,  9., 10., 11.]])

In [32]: a2.size

Out[32]: 12

In [33]: a1

Out[33]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [34]: a1.size
```

```
Out[34]: 10
```

item size

```
In [35]: a1.itemsize
```

```
Out[35]: 4
```

```
In [36]: a2.itemsize
```

```
Out[36]: 8
```

```
In [37]: a3.itemsize
```

```
Out[37]: 4
```

```
In [38]: print(a1.dtype)
         print(a2.dtype)
         print(a3.dtype)
```

```
int32
float64
int32
```

changing data type

astype()

```
In [39]: x=np.array([33,22,2.5])
         x
```

```
Out[39]: array([33. , 22. ,  2.5])
```

```
In [40]: x.astype(int)
```

```
Out[40]: array([33, 22,  2])
```

```
In [41]: z1 = np.arange(12).reshape(3,4)
         z2 = np.arange(12,24).reshape(3,4)
```

```
In [42]: z1
```

```
Out[42]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [43]: z2
```

```
Out[43]: array([[12, 13, 14, 15],
                [16, 17, 18, 19],
                [20, 21, 22, 23]])
```

relational operator

```
In [44]: z1+2
```

```
Out[44]: array([[ 2,  3,  4,  5],
                [ 6,  7,  8,  9],
                [10, 11, 12, 13]])
```

```
In [45]: z2-2
```

```
Out[45]: array([[10, 11, 12, 13],
                [14, 15, 16, 17],
                [18, 19, 20, 21]])
```

```
In [46]: z1*2
```

```
Out[46]: array([[ 0,  2,  4,  6],
                [ 8, 10, 12, 14],
                [16, 18, 20, 22]])
```

```
In [47]: z1**2
```

```
Out[47]: array([[  0,   1,   4,   9],
                [ 16,  25,  36,  49],
                [ 64,  81, 100, 121]])
```

```
In [48]: z1%2
```

```
Out[48]: array([[0, 1, 0, 1],
                [0, 1, 0, 1],
                [0, 1, 0, 1]], dtype=int32)
```

```
In [49]: z2
```

```
Out[49]: array([[12, 13, 14, 15],
                [16, 17, 18, 19],
                [20, 21, 22, 23]])
```

```
In [50]: z2 >2
```

```
Out[50]: array([[ True,  True,  True,  True],
                [ True,  True,  True,  True],
                [ True,  True,  True,  True]])
```

```
In [51]: z2>20
```

```
Out[51]: array([[False, False, False, False],
                [False, False, False, False],
                [False,  True,  True,  True]])
```

vector equation

```
In [52]: z1
```

```
Out[52]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [53]: z2
```

```
Out[53]: array([[12, 13, 14, 15],
                [16, 17, 18, 19],
                [20, 21, 22, 23]])
```

In [54]:
```
z1+z2
```

Out[54]:
```
array([[12, 14, 16, 18],
       [20, 22, 24, 26],
       [28, 30, 32, 34]])
```

In [55]:
```
z1*z2
```

Out[55]:
```
array([[  0,  13,  28,  45],
       [ 64,  85, 108, 133],
       [160, 189, 220, 253]])
```

In [56]:
```
z1-z2
```

Out[56]:
```
array([[-12, -12, -12, -12],
       [-12, -12, -12, -12],
       [-12, -12, -12, -12]])
```

In [57]:
```
z1/z2
```

Out[57]:
```
array([[0.        , 0.07692308, 0.14285714, 0.2       ],
       [0.25      , 0.29411765, 0.33333333, 0.36842105],
       [0.4       , 0.42857143, 0.45454545, 0.47826087]])
```

array function

In [58]:
```
k1=np.random.random((3,3))
k1=np.round(k1*100)
k1
```

Out[58]:
```
array([[59., 77., 52.],
       [ 5., 23., 85.],
       [97., 92., 93.]])
```

In [59]:
```
np.max(k1)
```

Out[59]:
```
97.0
```

In [60]:
```
np.min(k1)
```

```
Out[60]:  5.0

In [61]:  np.sum(k1)

Out[61]:  583.0

In [62]:  np.prod(k1)

Out[62]:  1916484700930800.0

In [63]:  np.max(k1,axis=1)

Out[63]:  array([77., 85., 97.])

In [64]:  np.prod(k1,axis=0)

Out[64]:  array([ 28615., 162932., 411060.])

In [65]:  k1

Out[65]:  array([[59., 77., 52.],
                 [ 5., 23., 85.],
                 [97., 92., 93.]])

In [66]:  np.mean(k1)

Out[66]:  64.77777777777777

In [67]:  k1.mean(axis=0,dtype=int)

Out[67]:  array([53, 64, 76])

In [68]:  np.median(k1)

Out[68]:  77.0

In [69]:  np.median(k1,axis=1)

Out[69]:  array([59., 23., 93.])
```

In [70]:
```
np.std(k1)
```

Out[70]:
```
31.000995603287674
```

In [71]:
```
np.std(k1,axis=0)
```

Out[71]:
```
array([37.74770045, 29.63106478, 17.74510887])
```

In [72]:
```
np.var(k1)
```

Out[72]:
```
961.0617283950617
```

In [73]:
```
np.sin(k1)
```

Out[73]:
```
array([[ 0.63673801,  0.99952016,  0.98662759],
       [-0.95892427, -0.8462204 , -0.17607562],
       [ 0.37960774, -0.77946607, -0.94828214]])
```

In [74]:
```
np.cos(k1)
```

Out[74]:
```
array([[-0.77108022, -0.03097503, -0.16299078],
       [ 0.28366219, -0.53283302, -0.98437664],
       [-0.92514754, -0.62644445,  0.3174287 ]])
```

In [75]:
```
np.tan(k1)
```

Out[75]:
```
array([[ -0.82577401, -32.26857578,  -6.05327238],
       [ -3.38051501,   1.58815308,   0.17887017],
       [ -0.4103213 ,   1.24427006,  -2.98738626]])
```

In [76]:
```
s2 = np.arange(12).reshape(3,4)
s3 = np.arange(12,24).reshape(4,3)
```

In [77]:
```
s2
```

Out[77]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [78]:
```
s3
```

```
Out[78]:  array([[12, 13, 14],
                 [15, 16, 17],
                 [18, 19, 20],
                 [21, 22, 23]])
```

```
In [79]:  np.dot(s2,s3)
```

```
Out[79]:  array([[114, 120, 126],
                 [378, 400, 422],
                 [642, 680, 718]])
```

```
In [80]:  np.exp(s2)
```

```
Out[80]:  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 [81]:  arr = np.array([1.2, 2.7, 3.5, 4.9])
          rounded_arr = np.round(arr)
          print(rounded_arr)

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

```
In [82]:  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 [83]:  np.round(np.random.random((2,3))*100)
```

```
Out[83]:  array([[40.,  2., 73.],
                 [20., 50., 99.]])
```

```
In [84]:  arr = np.array([1.2, 2.7, 3.5, 4.9])
          floored_arr = np.floor(arr)
          print(floored_arr)

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

```
In [85]:  np.floor(np.random.random((2,3))*100)
```

```
Out[85]:  array([[67., 83., 33.],
                 [29.,  1., 33.]])
```

In [86]:
```
arr = np.array([1.2, 2.7, 3.5, 4.9])
ceiled_arr = np.ceil(arr)
print(ceiled_arr)
```

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

In [87]:
```
np.ceil(np.random.random((2,3))*100)
```

Out[87]:
```
array([[22., 42., 69.],
       [83., 63., 48.]])
```

indexing & slicing

In [88]:
```
p1 = np.arange(10)
p2 = np.arange(12).reshape(3,4)
p3 = np.arange(8).reshape(2,2,2)
```

In [89]:
```
p1
```

Out[89]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [90]:
```
p2
```

Out[90]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [91]:
```
p3
```

Out[91]:
```
array([[[0, 1],
        [2, 3]],

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

In [92]:
```
p2
```

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

In [93]:
```
p2[1,2]
```
Out[93]:
```
6
```
In [94]:
```
p2[2,3]
```
Out[94]:
```
11
```
In [95]:
```
p2[1,0]
```
Out[95]:
```
4
```
In [96]:
```
p3
```
Out[96]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```
In [97]:
```
p3[1,0,1]
```
Out[97]:
```
5
```
In [98]:
```
p3 = np.arange(27).reshape(3,3,3)
p3
```
Out[98]:
```
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 [99]:
```
p3[1]
```

```
Out[99]: array([[ 9, 10, 11],
                [12, 13, 14],
                [15, 16, 17]])
```

```
In [100… p3[::2]
```

```
Out[100… array([[[ 0,  1,  2],
                 [ 3,  4,  5],
                 [ 6,  7,  8]],

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

```
In [101… p3[0,1,:]
```

```
Out[101… array([3, 4, 5])
```

```
In [102… p3[2,1:]
```

```
Out[102… array([[21, 22, 23],
                [24, 25, 26]])
```

```
In [103… p3[2,1:,1:]
```

```
Out[103… array([[22, 23],
                [25, 26]])
```

```
In [104… p1
```

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

```
In [105… for i in p1:
              print(i)
```

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

In [106…

```
p2
```

Out[106…

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

In [107…

```
for i in p2:
    print(i)
```

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

In [108…

```
p3
```

Out[108…

```
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 [ ]:

In [109…

```
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 [110…

```
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
```

Transpose

row to column &columns to row

```
In [111… p2
```

```
Out[111… array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [112… np.transpose(p2)
```

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

```
In [113… p2.T
```

```
Out[113… array([[ 0,  4,  8],
                [ 1,  5,  9],
                [ 2,  6, 10],
                [ 3,  7, 11]])
```

```
In [114… p3
```

```
Out[114… 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 [115… p3.T
```

Out[115…
```
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]]])
```

Ravel.

converting into one dimensional array

In [116…
```
p3
```

Out[116…
```
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 [117…
```
p3.ravel()
```

Out[117…
```
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 [118…
```
p2
```

Out[118…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [119…
```
p2.ravel()
```
Out[119…
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

stacking

In [120…
```
w1 = np.arange(12).reshape(3,4)
w2 = np.arange(12,24).reshape(3,4)
```
In [121…
```
w1
```
Out[121…
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```
In [122…
```
w2
```
Out[122…
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```
In [123…
```
np.hstack((w1,w2))
```
Out[123…
```
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 [124…
```
np.vstack((w1,w2))
```
Out[124…
```
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]])
```

splitting

In [125…
```
np.hsplit(w1,2) #horizontal splitting
```

Out[125…
```
[array([[0, 1],
        [4, 5],
        [8, 9]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]])]
```

In [126…
```
np.vsplit(w2,3) #vertical splitting
```

Out[126…
```
[array([[12, 13, 14, 15]]),
 array([[16, 17, 18, 19]]),
 array([[20, 21, 22, 23]])]
```

In [127…
```
P = [i for i in range(10000000)]
import sys
sys.getsizeof(P)
```

Out[127…
```
89095160
```

In [128…
```
R=np.arange(10000000)
sys.getsizeof(R)
```

Out[128…
```
40000112
```

In [129…
```
R = np.arange(10000000, dtype =np.int16)
sys.getsizeof(R)
```

Out[129…
```
20000112
```

In [130…
```
G = np.random.randint(1,100,24).reshape(6,4)
G
```

Out[130…
```
array([[56, 70, 45, 40],
       [33, 94, 54, 78],
       [69, 93, 26, 58],
       [ 2, 68, 98, 16],
       [53, 58, 96, 34],
       [59, 19, 11, 32]])
```

In [131…
```
G>50
```

Out[131…
```
array([[ True,  True, False, False],
       [False,  True,  True,  True],
       [ True,  True, False,  True],
       [False,  True,  True, False],
       [ True,  True,  True, False],
       [ True, False, False, False]])
```

In [132…
```
G[G>50]
```

Out[132…
```
array([56, 70, 94, 54, 78, 69, 93, 58, 68, 98, 53, 58, 96, 59])
```

In [133…
```
G%2==0
```

Out[133…
```
array([[ True,  True, False,  True],
       [False,  True,  True,  True],
       [False, False,  True,  True],
       [ True,  True,  True,  True],
       [False,  True,  True,  True],
       [False, False, False,  True]])
```

In [134…
```
G [ G % 2 == 0]
```

Out[134…
```
array([56, 70, 40, 94, 54, 78, 26, 58,  2, 68, 98, 16, 58, 96, 34, 32])
```

In [135…
```
(G > 50 ) & (G % 2 == 0)
```

Out[135…
```
array([[ True,  True, False, False],
       [False,  True,  True,  True],
       [False, False, False,  True],
       [False,  True,  True, False],
       [False,  True,  True, False],
       [False, False, False, False]])
```

In [136…
```
G [(G > 50 ) & (G % 2 == 0)]
```

Out[136…
```
array([56, 70, 94, 54, 78, 58, 68, 98, 58, 96])
```

In [137…
```
G % 7 == 0
```

Out[137…
```
array([[ True,  True, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False,  True, False],
       [False, False, False, False],
       [False, False, False, False]])
```

In [138…
```
G[~(G % 7 == 0)]
```

Out[138…
```
array([45, 40, 33, 94, 54, 78, 69, 93, 26, 58,  2, 68, 16, 53, 58, 96, 34,
       59, 19, 11, 32])
```

Broadcasting

In [139…
```
# 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 [140…
```
# 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 [141…
```
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 [142…
```
print(b) # 1d array
```

```
[0 1 2]
```

In [143…
```
print(a+b)
```

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

Working with mathematical formula

In [144…
```
k=np.arange(10)
k
```

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

In [145…
```
np.sum(k)
```

Out[145…
```
45
```

In [146…
```
np.sin(k)
```

Out[146…
```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849])
```

sigmoid

In [147…
```
def sigmoid(array):
    return 1/(1+np.exp(-(array)))
    k = np.arange(10)
sigmoid(k)
```

Out[147…

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

In [148…

```
k = np.arange(100)
sigmoid(k)
```

Out[148…

```
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.        ])
```

mean squarred error

In [149…

```
actual = np.random.randint(1,50,25)
predicted = np.random.randint(1,50,25)
```

In [150…

```
actual
```

Out[150…

```
array([34,  1, 38, 32, 11, 31, 22, 44, 39,  4, 43, 21, 46, 18, 23,  8, 39,
       43, 31, 17, 26, 40, 22, 14, 33])
```

In [151…

```
predicted
```

Out[151…
```
array([45,  2,  2, 37, 11, 27, 47, 16, 24, 46, 29, 49, 28, 35, 44, 49, 47,
       18, 46,  6, 41, 21,  9, 25,  3])
```

In [152…
```
def mse(actual,predicted):
    return np.mean((actual-predicted)**2)

mse(actual,predicted)
```

Out[152…
```
455.32
```

In [153…
```
actual-predicted
```

Out[153…
```
array([-11,  -1,  36,  -5,   0,   4, -25,  28,  15, -42,  14, -28,  18,
       -17, -21, -41,  -8,  25, -15,  11, -15,  19,  13, -11,  30])
```

In [154…
```
(actual-predicted)**2
```

Out[154…
```
array([ 121,    1, 1296,   25,    0,   16,  625,  784,  225, 1764,  196,
        784,  324,  289,  441, 1681,   64,  625,  225,  121,  225,  361,
        169,  121,  900])
```

In [155…
```
np.mean((actual-predicted)**2)
```

Out[155…
```
455.32
```

Working with missing value

In [156…
```
S = np.array([1,2,3,4,np.nan,6])
S
```

Out[156…
```
array([ 1.,  2.,  3.,  4., nan,  6.])
```

In [157…
```
np.isnan(S)
```

Out[157…
```
array([False, False, False, False,  True, False])
```

In [158…
```
S[np.isnan(S)]
```

Out[158…
```
array([nan])
```

In [159…
```
S[~np.isnan(S)]
```

Out[159…
```
array([1., 2., 3., 4., 6.])
```

plotting graphs

In [160…
```
x =np.linspace(-10,10,100)
x
```

Out[160…
```
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 [161…
```
y=x
```

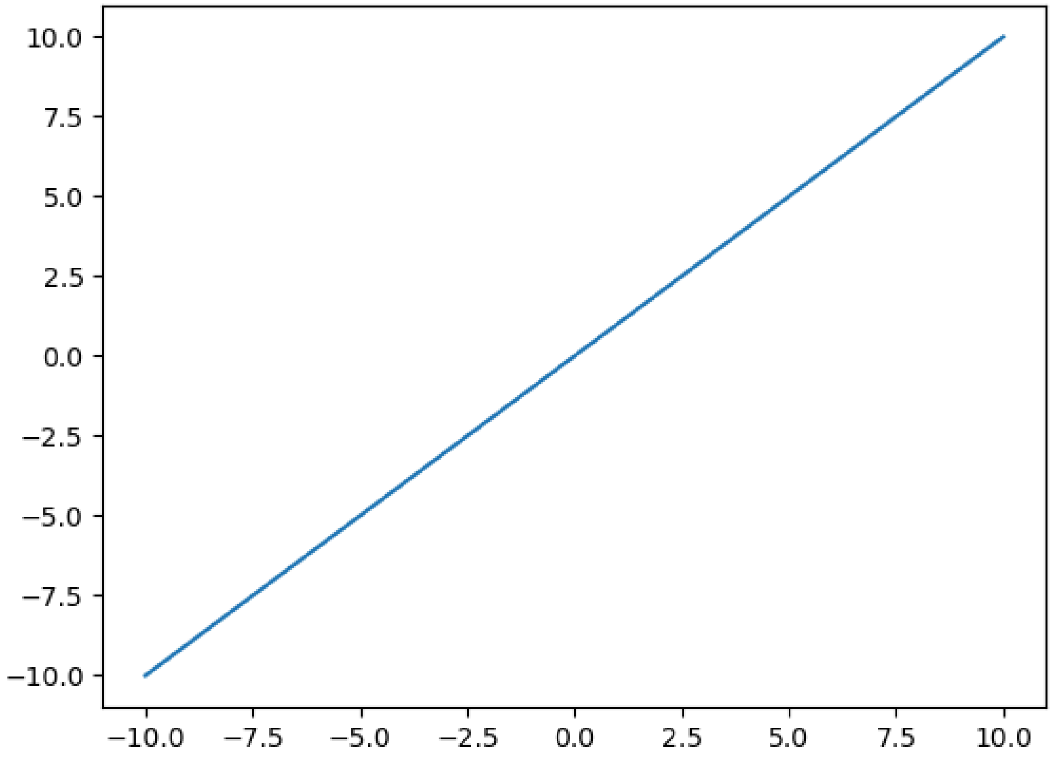
In [162…
```
y
```

```
Out[162… 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 [163…

```python
import matplotlib.pyplot as plt
plt.plot(x ,y)
```

```
Out[163… [<matplotlib.lines.Line2D at 0x2bbb5eb2960>]
```

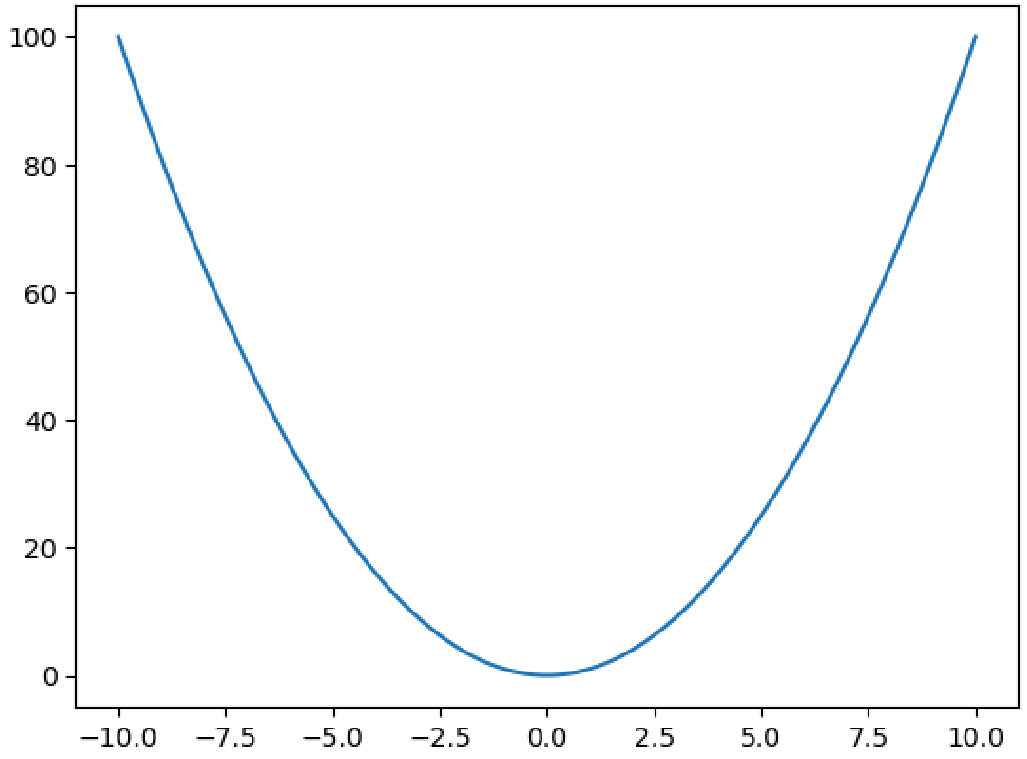


In [164…

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

Out[164…

```
[<matplotlib.lines.Line2D at 0x2bbb76baa50>]
```

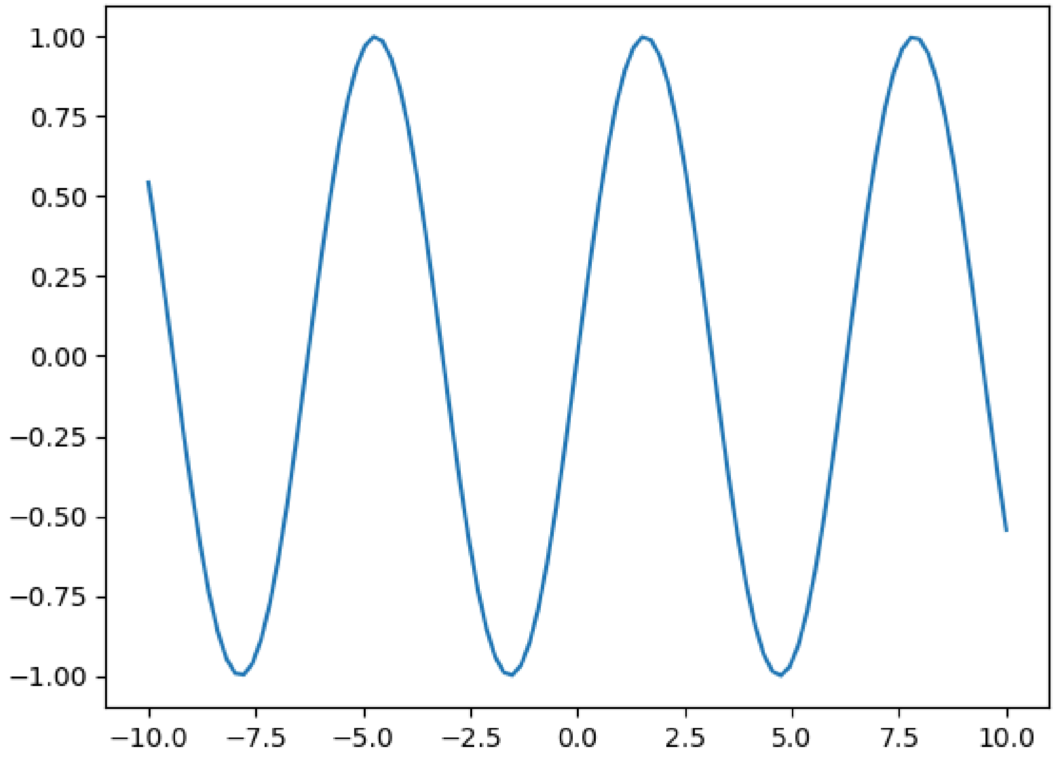


In [165…

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

Out[165…

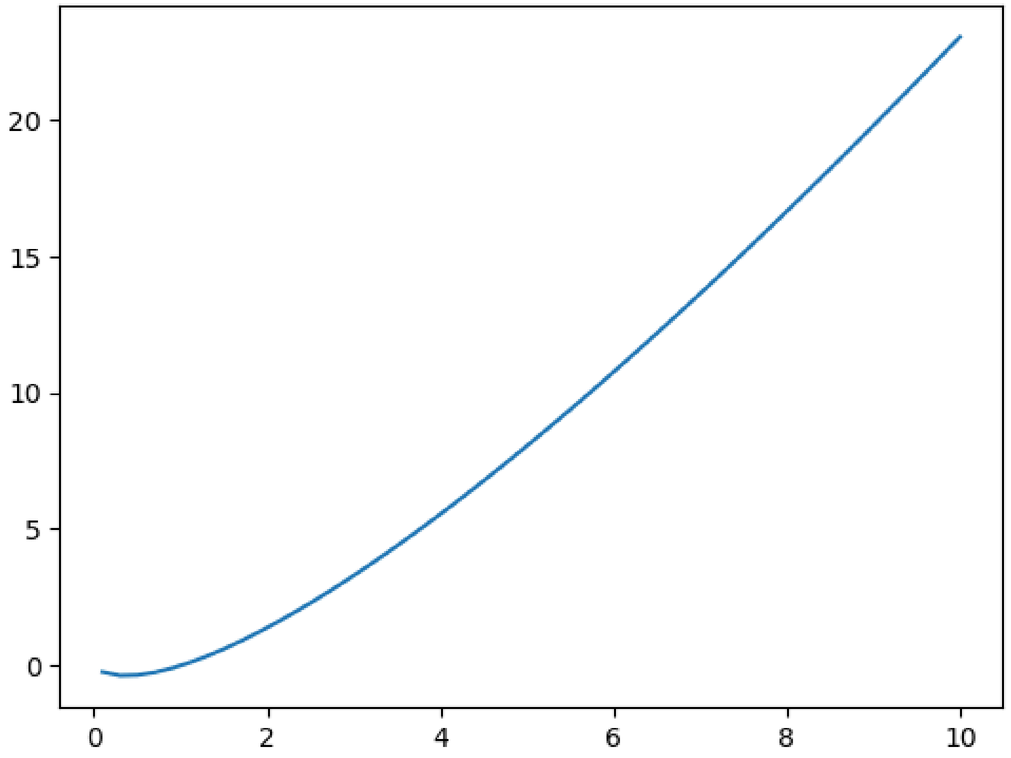
```
[<matplotlib.lines.Line2D at 0x2bbb6ee0da0>]
```



In [166…
```
x = np.linspace(-10,10,100)
y = x * np.log(x)
plt.plot(x,y)
```

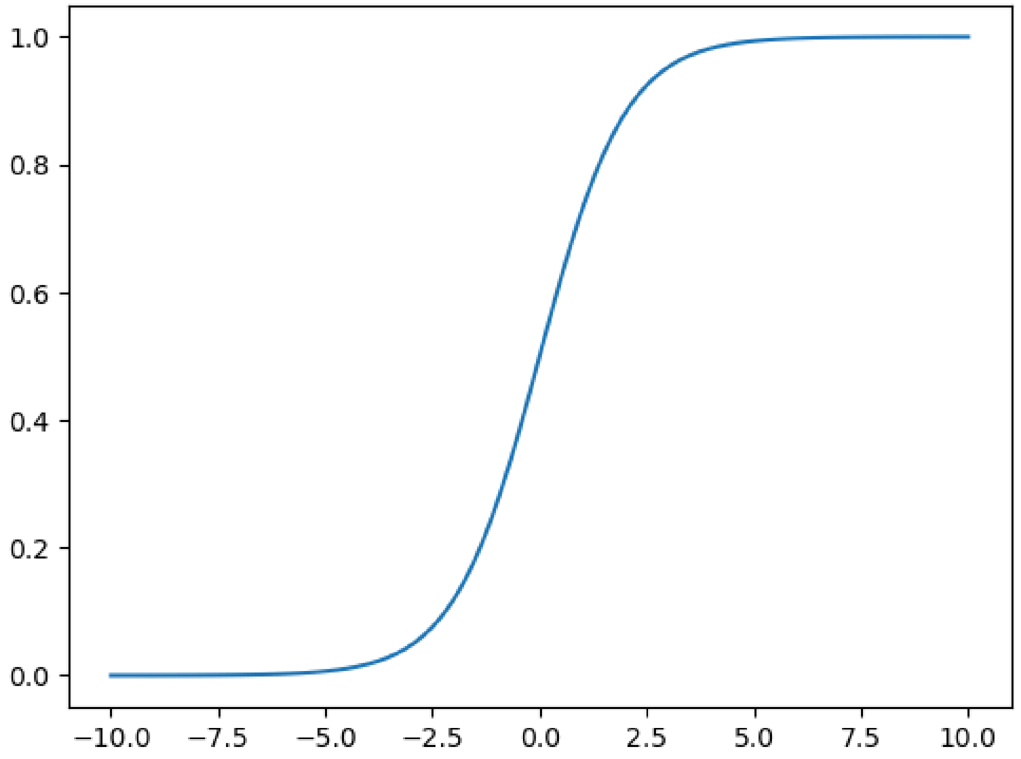
```
C:\Users\YASH\AppData\Local\Temp\ipykernel_7232\3240292629.py:2: RuntimeWarning: invalid value encountered in log
  y = x * np.log(x)
```

Out[166…
```
[<matplotlib.lines.Line2D at 0x2bbb6f690a0>]
```



In [167…
```
x = np.linspace(-10,10,100)
y = 1/(1+np.exp(-x))
plt.plot(x,y)
```

Out[167…
```
[<matplotlib.lines.Line2D at 0x2bbb6fafe60>]
```

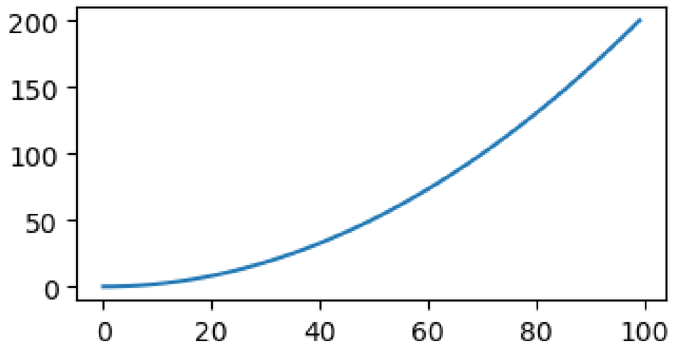


In [168…
```
import numpy as np
import matplotlib.pyplot as plt
```

In [169…
```
x = np.linspace(0,10,100)
y = np.linspace(0,10,100)
```

In [170…
```
f = x**2+y**2
```

In [171…
```
plt.figure(figsize=(4,2))
plt.plot(f)
plt.show()
```



In [172…
```
x = np.arange(3)
y = np.arange(3)
```

In [173…
```
x
```

Out[173…
```
array([0, 1, 2])
```

In [174…
```
y
```

Out[174…
```
array([0, 1, 2])
```

Generating a meshgrid:

In [175…
```
xv ,yv = np.meshgrid(x,y)
```

In [176…
```
xv
```

Out[176…
```
array([[0, 1, 2],
       [0, 1, 2],
       [0, 1, 2]])
```

In [177…
```
yv
```

Out[177…
```
array([[0, 0, 0],
       [1, 1, 1],
       [2, 2, 2]])
```

In [178…
```
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 [179…
```
P_1, V_1 = np.meshgrid(P,V)
```

In [180…
```
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 [181…
```
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.]]
```

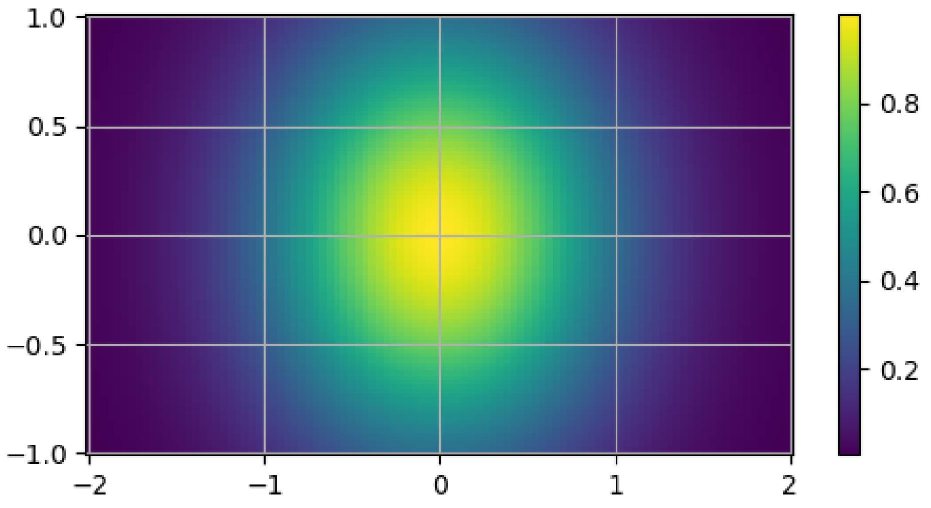
Numpy Meshgrid Creates Coordinates for a Grid System

In [182…
```
xv**2 + yv**2
```

Out[182…
```
array([[0, 1, 4],
       [1, 2, 5],
       [4, 5, 8]])
```

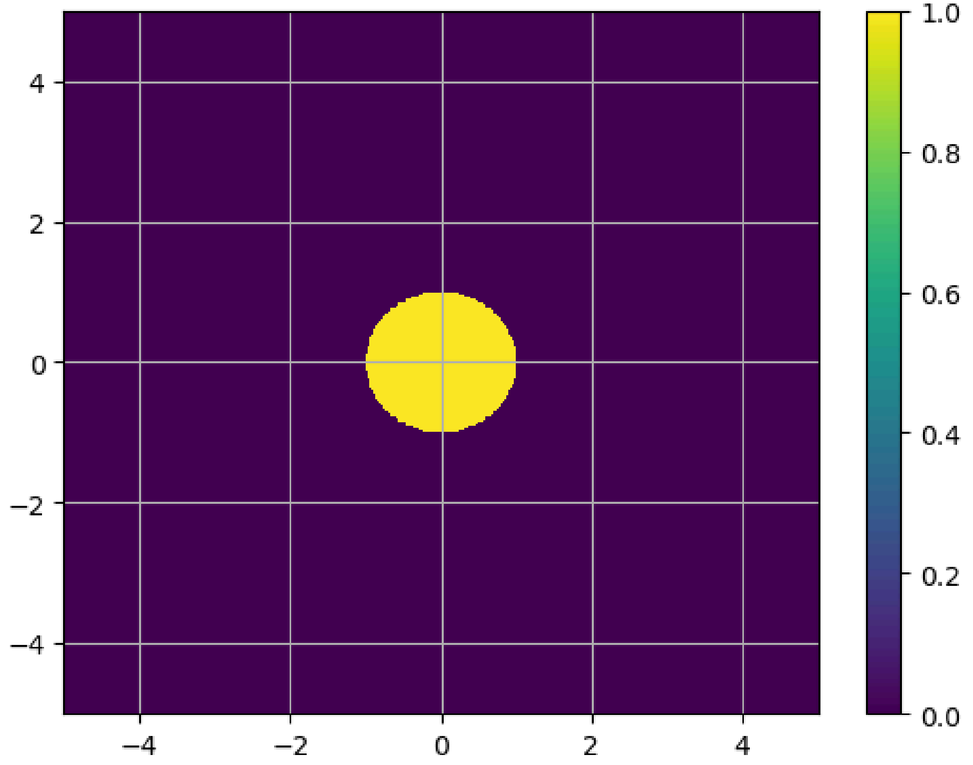
In [183…
```
x = np.linspace(-2,2,100)
y = np.linspace(-1,1,100)
xv, yv = np.meshgrid(x, y)
f = np.exp(-xv**2-yv**2)
```

In [184…
```
plt.figure(figsize=(6, 3))
plt.pcolormesh(xv, yv, f, shading='auto')
plt.colorbar()
plt.grid()
plt.show()
```



In [185…
```
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()
```

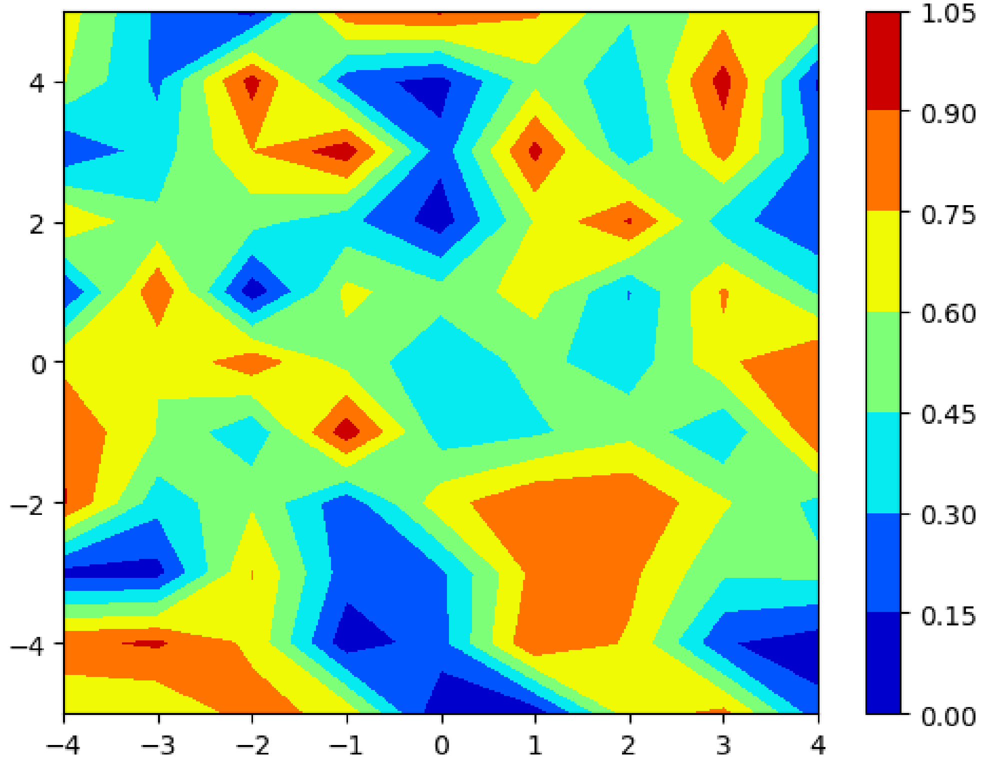


In [186…
```
x = np.linspace(-4, 4, 9)
```

In [187…
```
y = np.linspace(-5, 5, 11)
```

In [188…
```
x_1, y_1 = np.meshgrid(x, y)
```

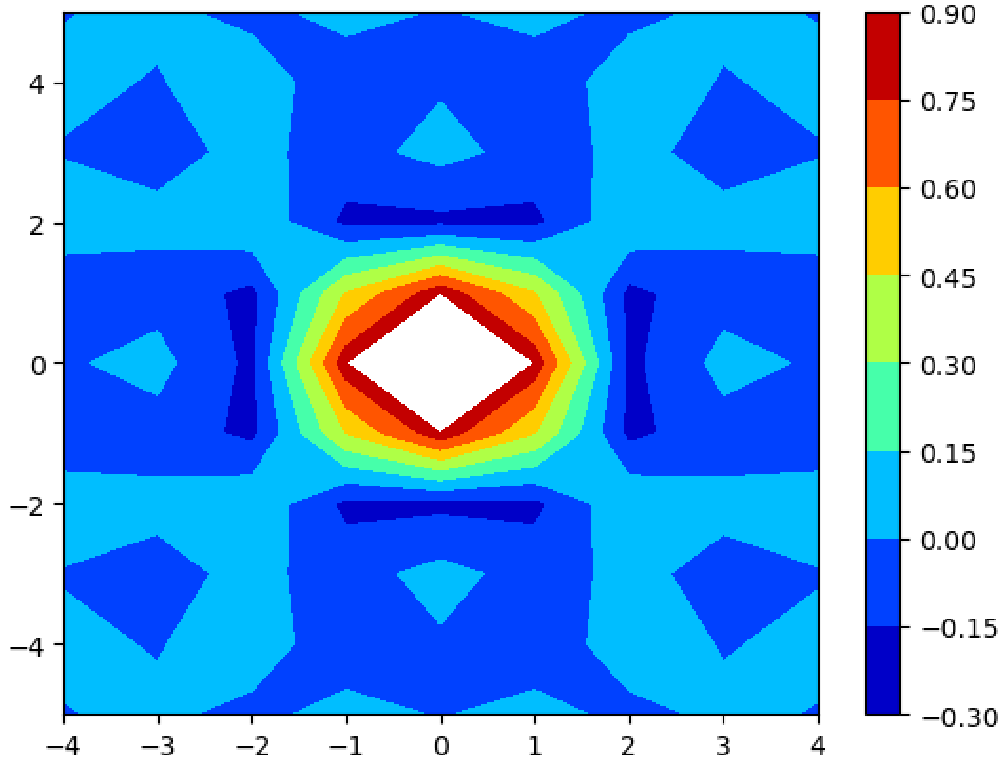
In [189…
```
random_data = np.random.random((11, 9))
plt.contourf(x_1, y_1, random_data, cmap = 'jet')
plt.colorbar()
plt.show()
```



In [190…

```
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\YASH\AppData\Local\Temp\ipykernel_7232\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 [191… x_1, y_1 = np.meshgrid(x, y, sparse = True)
```

```
In [192… x_1
```

```
Out[192… array([[-4., -3., -2., -1.,  0.,  1.,  2.,  3.,  4.]])
```

```
In [193… y_1
```

```
Out[193…  array([[-5.],
                 [-4.],
                 [-3.],
                 [-2.],
                 [-1.],
                 [ 0.],
                 [ 1.],
                 [ 2.],
                 [ 3.],
                 [ 4.],
                 [ 5.]])
```

sorting

```
In [194…  a = np.random.randint(1,100,15) #1D
          a
```

```
Out[194…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

```
In [195…  b = np.random.randint(1,100,24).reshape(6,4) #
          b
```

```
Out[195…  array([[77, 10, 99, 39],
                 [53, 89, 19, 74],
                 [90,  9, 27, 92],
                 [ 6, 34,  6, 37],
                 [45, 36, 30, 87],
                 [ 1, 13, 70,  8]])
```

```
In [196…  np.sort(a)
```

```
Out[196…  array([10, 14, 23, 25, 28, 32, 36, 40, 55, 59, 64, 71, 74, 74, 87])
```

```
In [197…  np.sort(b)
```

```
Out[197…  array([[10, 39, 77, 99],
                 [19, 53, 74, 89],
                 [ 9, 27, 90, 92],
                 [ 6,  6, 34, 37],
                 [30, 36, 45, 87],
                 [ 1,  8, 13, 70]])
```

append

In [198…
```
a
```

Out[198…
```
array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

In [199…
```
np.append(a,200)
```

Out[199…
```
array([ 59,  55,  14,  87,  71,  36,  28,  64,  74,  40,  32,  25,  23,
        74,  10, 200])
```

In [200…
```
b
```

Out[200…
```
array([[77, 10, 99, 39],
       [53, 89, 19, 74],
       [90,  9, 27, 92],
       [ 6, 34,  6, 37],
       [45, 36, 30, 87],
       [ 1, 13, 70,  8]])
```

In [201…
```
np.append(b,np.ones((b.shape[0],1)))
```

Out[201…
```
array([77., 10., 99., 39., 53., 89., 19., 74., 90.,  9., 27., 92.,  6.,
       34.,  6., 37., 45., 36., 30., 87.,  1., 13., 70.,  8.,  1.,  1.,
        1.,  1.,  1.,  1.])
```

In [202…
```
np.append(b,np.ones((b.shape[0],1)),axis=1)
```

Out[202…
```
array([[77., 10., 99., 39.,  1.],
       [53., 89., 19., 74.,  1.],
       [90.,  9., 27., 92.,  1.],
       [ 6., 34.,  6., 37.,  1.],
       [45., 36., 30., 87.,  1.],
       [ 1., 13., 70.,  8.,  1.]])
```

In [203…
```
np.append(b,np.random.random((b.shape[0],1)),axis=1)
```

Out[203…
```
array([[77.        , 10.        , 99.        , 39.        ,  0.83512648],
       [53.        , 89.        , 19.        , 74.        ,  0.66675291],
       [90.        ,  9.        , 27.        , 92.        ,  0.4475757 ],
       [ 6.        , 34.        ,  6.        , 37.        ,  0.75162848],
       [45.        , 36.        , 30.        , 87.        ,  0.862862  ],
       [ 1.        , 13.        , 70.        ,  8.        ,  0.59478929]])
```

np.concetenate

In [204…
```
c = np.arange(6).reshape(2,3)
d = np.arange(6,12).reshape(2,3)
```

In [205…
```
c
```

Out[205…
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [206…
```
d
```

Out[206…
```
array([[ 6,  7,  8],
       [ 9, 10, 11]])
```

In [207…
```
np.concatenate((c,d))
```

Out[207…
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [208…
```
np.concatenate((c,d),axis=1)
```

Out[208…
```
array([[ 0,  1,  2,  6,  7,  8],
       [ 3,  4,  5,  9, 10, 11]])
```

np.unique

In [209…
```
e = np.array([1,1,2,2,3,3,4,4,5,5,6,6])
e
```

```
Out[209… array([1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6])
```

```
In [210… np.unique(e)
```

```
Out[210… array([1, 2, 3, 4, 5, 6])
```

np.expand_dims

```
In [211… a
```

```
Out[211… array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

```
In [212… a.shape
```

```
Out[212… (15,)
```

```
In [213… np.expand_dims(a,axis = 0)
```

```
Out[213… array([[59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10]])
```

```
In [214… np.expand_dims(a,axis = 0).shape
```

```
Out[214… (1, 15)
```

```
In [215… np.expand_dims(a,axis = 1)
```

```
Out[215…  array([[59],
                 [55],
                 [14],
                 [87],
                 [71],
                 [36],
                 [28],
                 [64],
                 [74],
                 [40],
                 [32],
                 [25],
                 [23],
                 [74],
                 [10]])
```

```
In [216…  np.expand_dims(a,axis = 1).shape
```

```
Out[216…  (15, 1)
```

np.where

```
In [217…  a
```

```
Out[217…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

```
In [218…  np.where(a>50)
```

```
Out[218…  (array([ 0,  1,  3,  4,  7,  8, 13], dtype=int64),)
```

```
In [219…  np.where(a>50,0,a)
```

```
Out[219…  array([ 0,  0, 14,  0,  0, 36, 28,  0,  0, 40, 32, 25, 23,  0, 10])
```

```
In [220…  np.where(a%2 == 0,0,a)
```

```
Out[220…  array([59, 55,  0, 87, 71,  0,  0,  0,  0,  0,  0, 25, 23,  0,  0])
```

np.argmax

```
In [221…  a
Out[221…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])

In [222…  np.argmax(a)
Out[222…  3

In [223…  b
Out[223…  array([[77, 10, 99, 39],
                 [53, 89, 19, 74],
                 [90,  9, 27, 92],
                 [ 6, 34,  6, 37],
                 [45, 36, 30, 87],
                 [ 1, 13, 70,  8]])

In [224…  np.argmax(b)
Out[224…  2

In [225…  np.argmax(b,axis=1)
Out[225…  array([2, 1, 3, 3, 3, 2], dtype=int64)

In [226…  a
Out[226…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])

In [227…  np.argmin(a)
Out[227…  14

In [228…  b
```

```
Out[228…  array([[77, 10, 99, 39],
                 [53, 89, 19, 74],
                 [90,  9, 27, 92],
                 [ 6, 34,  6, 37],
                 [45, 36, 30, 87],
                 [ 1, 13, 70,  8]])
```

```
In [229…  np.argmin(b)
```

```
Out[229…  20
```

```
In [230…  np.argmin(b,axis=0)
```

```
Out[230…  array([5, 2, 3, 5], dtype=int64)
```

On statistcs

np.cumsum

```
In [231…  a
```

```
Out[231…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

```
In [232…  np.cumsum(a)
```

```
Out[232…  array([ 59, 114, 128, 215, 286, 322, 350, 414, 488, 528, 560, 585, 608,
                 682, 692])
```

```
In [233…  b
```

```
Out[233…  array([[77, 10, 99, 39],
                 [53, 89, 19, 74],
                 [90,  9, 27, 92],
                 [ 6, 34,  6, 37],
                 [45, 36, 30, 87],
                 [ 1, 13, 70,  8]])
```

```
In [234…  np.cumsum(b)
```

Out[234…
```
array([  77,   87,  186,  225,  278,  367,  386,  460,  550,  559,  586,
        678,  684,  718,  724,  761,  806,  842,  872,  959,  960,  973,
       1043, 1051])
```

In [235…
```
np.cumsum(b,axis=1)
```

Out[235…
```
array([[ 77,  87, 186, 225],
       [ 53, 142, 161, 235],
       [ 90,  99, 126, 218],
       [  6,  40,  46,  83],
       [ 45,  81, 111, 198],
       [  1,  14,  84,  92]])
```

In [236…
```
np.cumsum(b,axis=0)
```

Out[236…
```
array([[ 77,  10,  99,  39],
       [130,  99, 118, 113],
       [220, 108, 145, 205],
       [226, 142, 151, 242],
       [271, 178, 181, 329],
       [272, 191, 251, 337]])
```

cumprod()

In [237…
```
a
```

Out[237…
```
array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

In [238…
```
np.cumprod(a)
```

Out[238…
```
array([         59,        3245,       45430,     3952410,   280621110,
        1512425368,  -601762656,   141895680,  1910345728,  -895582208,
        1406140416,   793772032,  1076887552, -1914699776, -1967128576])
```

np.percentile()

In [239…
```
a
```

Out[239…
```
array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

np.percentile(a,100)

```
In [240…  np.percentile(a,0)

Out[240…  10.0

In [241…  np.percentile(a,50)

Out[241…  40.0

In [242…  np.median(a)

Out[242…  40.0
```

median=percentile

np.histogram

```
In [243…  a

Out[243…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])

In [244…  np.histogram(a , bins= [10,20,30,40,50,60,70,80,90,100])

Out[244…  (array([2, 3, 2, 1, 2, 1, 3, 1, 0], dtype=int64),
           array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100]))

In [245…  np.histogram(a , bins= [0,50,100])

Out[245…  (array([8, 7], dtype=int64), array([  0,  50, 100]))
```

flip

```
In [246…  a

Out[246…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

In [247…
```
np.flip(a)
```

Out[247…
```
array([10, 74, 23, 25, 32, 40, 74, 64, 28, 36, 71, 87, 14, 55, 59])
```

In [248…
```
b
```

Out[248…
```
array([[77, 10, 99, 39],
       [53, 89, 19, 74],
       [90,  9, 27, 92],
       [ 6, 34,  6, 37],
       [45, 36, 30, 87],
       [ 1, 13, 70,  8]])
```

In [249…
```
np.flip(a)
```

Out[249…
```
array([10, 74, 23, 25, 32, 40, 74, 64, 28, 36, 71, 87, 14, 55, 59])
```

In [250…
```
np.flip(b)
```

Out[250…
```
array([[ 8, 70, 13,  1],
       [87, 30, 36, 45],
       [37,  6, 34,  6],
       [92, 27,  9, 90],
       [74, 19, 89, 53],
       [39, 99, 10, 77]])
```

In [251…
```
np.flip(b,axis=1)
```

Out[251…
```
array([[39, 99, 10, 77],
       [74, 19, 89, 53],
       [92, 27,  9, 90],
       [37,  6, 34,  6],
       [87, 30, 36, 45],
       [ 8, 70, 13,  1]])
```

In [252…
```
np.flip(b,axis=0)
```

```
Out[252…  array([[ 1, 13, 70,  8],
                 [45, 36, 30, 87],
                 [ 6, 34,  6, 37],
                 [90,  9, 27, 92],
                 [53, 89, 19, 74],
                 [77, 10, 99, 39]])
```

np.put

```
In [253…  a
```

```
Out[253…  array([59, 55, 14, 87, 71, 36, 28, 64, 74, 40, 32, 25, 23, 74, 10])
```

```
In [254…  np.put(a,[0,1],[110,530])
```

```
In [255…  a
```

```
Out[255…  array([110, 530,  14,  87,  71,  36,  28,  64,  74,  40,  32,  25,  23,
                  74,  10])
```

np.delete

```
In [256…  a
```

```
Out[256…  array([110, 530,  14,  87,  71,  36,  28,  64,  74,  40,  32,  25,  23,
                  74,  10])
```

```
In [257…  np.delete(a,0)
```

```
Out[257…  array([530,  14,  87,  71,  36,  28,  64,  74,  40,  32,  25,  23,  74,
                  10])
```

```
In [258…  np.delete(a,[0,2,4])
```

```
Out[258…  array([530,  87,  36,  28,  64,  74,  40,  32,  25,  23,  74,  10])
```

```
In [259…  m = np.array([1,2,3,4,5])
          n = np.array([3,4,5,6,7])
```

```
In [260…  np.union1d(m,n)

Out[260…  array([1, 2, 3, 4, 5, 6, 7])

In [261…  np.intersect1d(m,n)

Out[261…  array([3, 4, 5])

In [262…  np.setdiff1d(m,n)

Out[262…  array([1, 2])

In [263…  np.setxor1d(m,n)

Out[263…  array([1, 2, 6, 7])

In [264…  np.in1d(m,1)

Out[264…  array([ True, False, False, False, False])

In [265…  m[np.in1d(m,1)]

Out[265…  array([1])

In [266…  np.in1d(m,10)

Out[266…  array([False, False, False, False, False])
```

np.clip

```
In [267…  a

Out[267…  array([110, 530,  14,  87,  71,  36,  28,  64,  74,  40,  32,  25,  23,
                  74,  10])

In [268…  np.clip(a, a_min=15 , a_max =50)

Out[268…  array([50, 50, 15, 50, 50, 36, 28, 50, 50, 40, 32, 25, 23, 50, 15])
```

np.swapaxes

In [269…
```
arr = np.array([[1, 2, 3], [4, 5, 6]])
swapped_arr = np.swapaxes(arr, 0, 1)
```

In [270…
```
arr
```

Out[270…
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [271…
```
swapped_arr
```

Out[271…
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

In [272…
```
print("Original array:")
print(arr)
```

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

In [273…
```
print("Swapped array:")
print(swapped_arr)
```

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

In [297…
```
add.accumulate(array([[1,2,3],[4,5,6]]),axis=0)
```

Out[297…
```
array([[1, 2, 3],
       [5, 7, 9]])
```

In [298…
```
add.accumulate(array([[1,2,3],[4,5,6]]), axis = 1)
```

Out[298…
```
array([[ 1,  3,  6],
       [ 4,  9, 15]])
```

add()

```
In [299…  add(array([-1.2, 1.2]), array([1,3]))

Out[299…  array([-0.2,  4.2])
```

all()

```
In [300…   a = array([True, False, True])

In [301…  a.all()

Out[301…  False

In [302…  all(a)

Out[302…  False

In [303…   a = array([1,2,3])

In [304…  all(a>0)

Out[304…  True

In [ ]:
```