In [2]:
```
import numpy as np
```

In [2]:
```
np.__version__
```

Out[2]:
```
'1.26.4'
```

In [4]:
```
np.arange(6)
```

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

In [8]:
```
my_arr = np.arange(1,10)      # prints o/p from 1 to n-1 values
my_arr
```

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

In [10]:
```
np.arange(6,7)      # in this starting is 6 and end value is 7 but array prints
```

Out[10]:
```
array([6])
```

In [11]:
```
np.arange(20,30,2)       # prints between 20,29 values with step count 2
```

Out[11]:
```
array([20, 22, 24, 26, 28])
```

Matrix printing using Arrays

In [15]:
```
my_arr = np.zeros((10,10),dtype = int)
my_arr
```

Out[15]:
```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

In [16]:
```
my_arr = np.ones(10)
my_arr
```

Out[16]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [18]:
```
my_arr = np.ones((1,100), dtype=int)
my_arr
```

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

linespace(arg1,arg2,arg3) is used to create an array with values that are spaced linearly with a specific Interval

where arg3 is says how many arg sholud be printed

In [3]:
```
my_arr = np.ones(10)
my_arr
```

Out[3]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [8]:
```
my_arr1= np.linspace(0,10, 4)
my_arr1
```

Out[8]:
```
array([ 0.        ,  3.33333333,  6.66666667, 10.        ])
```

In [13]:
```
my_arr1= np.linspace(10,20,8)
my_arr1
```

Out[13]:
```
array([10.        , 11.42857143, 12.85714286, 14.28571429, 15.71428571,
       17.14285714, 18.57142857, 20.        ])
```

Adding, removing, and sorting elements

sort() used to order the array

concate() is used to add both the arrays

size() gives the size of the array

In [15]:
```
my_arr2 = [3,4,2,6,5,7,1]
my_arr2
```

Out[15]:
```
[3, 4, 2, 6, 5, 7, 1]
```

In [16]:
```
np.sort(my_arr2)
```

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

In [34]:
```
my_arr1 = np.array([[3,2,5,3],[3,4,5,6]])
my_arr2 = np.array([ [6,7,8,9],[2,3,5,6]])    #for concatenation both the array
my_arr3 = np.concatenate((my_arr1,my_arr2))
my_arr3
```

Out[34]:
```
array([[3, 2, 5, 3],
       [3, 4, 5, 6],
       [6, 7, 8, 9],
       [2, 3, 5, 6]])
```

In [23]:
```
a = [1,2]
b = [3,4]
c = [5,6]
np.concatenate((a,b),axis = 0)
```

Out[23]:
```
array([1, 2, 3, 4])
```

In [31]:
```
my_arr1.ndim
```

Out[31]:
```
2
```

In [36]:
```
my_arr3.ndim
```

Out[36]:
```
2
```

In [37]:
```
my_arr3.size
```

Out[37]:
```
16
```

In [38]:
```
my_arr3.shape    # gives the dimensions of the array
```

Out[38]:
```
(4, 4)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

random() functions

every time this function will give different o/p

this function is used to get random OTP etc

In [20]:
```
np.random.rand(2)       #this will gives float values
```

Out[20]:
```
array([0.46923384, 0.22947438])
```

In [24]:
```
np.random.rand(8)
```

Out[24]:
```
array([0.31916523, 0.62296078, 0.06273441, 0.79139729, 0.2647462 ,
       0.34180704, 0.11328965, 0.32279117])
```

In [45]:
```
np.random.randint(8)        # this randint() will give the integer value till n-
```

Out[45]:
```
0
```

In [43]:
```
np.random.randint(2,10)     # o/p will gives one value starts from 2
```

Out[43]:
```
2
```

In [47]:
```
np.random.randint(2,10,4)    # it generates numbers in the condition of inclusiv
```

Out[47]:
```
array([8, 4, 2, 7])
```

In [64]:
```
np.random.randint(-30,10,4)     #prints both +ve and -ve values till 10 randomly
```

Out[64]:
```
array([-20,   3, -16, -13])
```

In [65]:
```
np.random.randint(-20,-10,3)    #prints only -ve values randomly
```

Out[65]:
```
array([-13, -18, -13])
```

reshape() function is used to change the array into matrix form based on the size

it takes 2 arg, 1st arg is represents rows and 2nd arg represents col

In [8]:
```
arr = np.arange(6)
arr
```

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

In [9]:
```
arr.reshape(2,3)
```

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

In [11]:
```
arr.reshape(3,2)
```

Out[11]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [4]:
```
arr.reshape(2,5)     # array is having 6 elements
                     # in the above code we can adjust 6 elements in 3 rows and
                     # but here we cannot adjust 6 elements in 2 rows and 5 colm
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[4], line 1
----> 1 arr.reshape(2,5)

ValueError: cannot reshape array of size 6 into shape (2,5)
```

In [12]:
```
arr.reshape(6,1)
```

Out[12]:
```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
```

In [13]:
```
arr.reshape(1,6)
```

Out[13]:
```
array([[0, 1, 2, 3, 4, 5]])
```

slicing

In [18]:
```
b= np.random.randint(2,10,(5,4))
b                                #prints values from 2 to 9 in 5/4 matrix form
                                   # when we prents enter for output for next ti
```

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

In [19]:
```
b[:]                # it will get the o/p same as random.randint gets
```

Out[19]:
```
array([[8, 9, 6, 5],
       [4, 8, 5, 2],
       [4, 6, 8, 6],
       [7, 2, 3, 3],
       [5, 9, 3, 2]])
```

In [20]:
```
b[2:]
```

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

In [23]:
```
b[0:3]
```

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

In [22]:
```
b[1:4]
```

Out[22]:
```
array([[4, 8, 5, 2],
       [4, 6, 8, 6],
       [7, 2, 3, 3]])
```

In [24]:
```
b[-1:]
```

Out[24]:
```
array([[5, 9, 3, 2]])
```

In [25]:
```
b[:-1]
```

Out[25]:
```
array([[8, 9, 6, 5],
       [4, 8, 5, 2],
       [4, 6, 8, 6],
       [7, 2, 3, 3]])
```

In [27]:
```
b
```

Out[27]:
```
array([[8, 9, 6, 5],
       [4, 8, 5, 2],
       [4, 6, 8, 6],
       [7, 2, 3, 3],
       [5, 9, 3, 2]])
```

In [29]:
```
b[1,3]      #prints 1st row 3rd col value
```

Out[29]:
```
2
```

In [33]:
```
b[3:]
```

Out[33]:
```
array([[7, 2, 3, 3],
       [5, 9, 3, 2]])
```

In [34]:
```
b[2,3]
```

Out[34]:
```
6
```

In [35]:
```
b[3,1]
```

Out[35]:
```
2
```

In [41]:
```
b1 = np.random.randint(1,100,(10,10))
b1
```

Out[41]:
```
array([[28, 13, 45, 59, 59, 94, 57,  6, 84, 57],
       [91,  9, 59,  4, 45, 71,  1, 95, 62, 20],
       [71, 48, 24, 23, 34, 60, 74, 63,  8, 14],
       [98, 92, 48, 54, 96, 64, 53, 90, 74,  7],
       [86, 52, 15, 64, 72, 78, 42, 18, 19, 59],
       [28, 83, 29, 88, 81, 23, 81, 75, 69, 64],
       [69, 74, 23, 74, 59, 86, 97, 59, 92, 25],
       [89, 98, 77, 51, 25,  8, 24,  9, 36, 80],
       [17, 61, 84, 56, 52, 32, 48, 24, 56, 59],
       [93, 20, 84, 87, 28, 51, 30, 27,  9, 14]])
```

In [43]:
```
b1[:]
```

Out[43]:
```
array([[ 3,  2, 37, 10, 63, 55, 95, 81,  7,  9],
       [18, 32, 18, 77, 90, 97, 35, 99, 69, 63],
       [11, 52, 64, 21, 70, 25, 82, 16, 45, 28],
       [ 5, 53, 99, 24, 65, 67, 64, 73, 63, 70],
       [26, 93, 29, 77, 85,  9, 90,  5, 55, 43],
       [23, 91, 66, 35, 14, 40, 28, 21, 79, 69],
       [50, 71,  6, 84, 90, 90, 32, 96, 95, 47],
       [81, 62, 64, 55, 77, 85, 30, 47, 68, 96],
       [ 3, 70, 99, 77, 75, 67, 29, 59, 51, 10],
       [45, 59,  3, 19, 16, 86, 92,  1, 77, 64]])
```

In [45]:
```
b1[5,6]
```

Out[45]:
```
28
```

In [50]:
```
b1[-8:]
```

Out[50]:
```
array([[11, 52, 64, 21, 70, 25, 82, 16, 45, 28],
       [ 5, 53, 99, 24, 65, 67, 64, 73, 63, 70],
       [26, 93, 29, 77, 85,  9, 90,  5, 55, 43],
       [23, 91, 66, 35, 14, 40, 28, 21, 79, 69],
       [50, 71,  6, 84, 90, 90, 32, 96, 95, 47],
       [81, 62, 64, 55, 77, 85, 30, 47, 68, 96],
       [ 3, 70, 99, 77, 75, 67, 29, 59, 51, 10],
       [45, 59,  3, 19, 16, 86, 92,  1, 77, 64]])
```

In [42]:
```
print(b1[b1<50])
```

```
[28 13 45  6  9  4 45  1 20 48 24 23 34  8 14 48  7 15 42 18 19 28 29 23
 23 25 25  8 24  9 36 17 32 48 24 20 28 30 27  9 14]
```

In [46]:
```
c = print(b1>=60)
```

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

In [49]:
```
my_arr1
```

Out[49]:
```
array([[3, 2, 5, 3],
       [3, 4, 5, 6]])
```

In [48]:
```
div_by_2 = my_arr1[my_arr1%2 ==0]
print(div_by_2)
```

```
[2 4 6]
```

In [57]:
```
a = my_arr1[(my_arr1 > 5) | (my_arr1 == 3)]
print(a)
```

```
[3 3 3 6]
```

In [59]:
```
a = my_arr1[(my_arr1 > 5) &(my_arr1 == 3)]
print(a)
```

```
[]
```

In [60]:
```
a = np.nonzero( my_arr1 <3)
print(a)
```

```
(array([0], dtype=int64), array([1], dtype=int64))
```

In [61]:
```
 x = np.arange(1, 25).reshape(2, 12)
print(x)
```

```
[[ 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 [65]:
```
 a1 = np.array([[1, 1], [2, 2]])
 a2 = np.array([[3, 3], [4, 4]])
print(a1,' \n' ,a2)
```

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

In [66]:
```
np.vstack([a1,a2])     #prints arrays in vertically
```

Out[66]:
```
array([[1, 1],
       [2, 2],
       [3, 3],
       [4, 4]])
```

In [67]:
```
np.hstack([a1,a2])    # prints arrays in  horizotally
```

Out[67]:
```
array([[1, 1, 3, 3],
       [2, 2, 4, 4]])
```

In [68]:
```
x = np.arange(1, 25).reshape(2, 12)
print(x)
```

```
[[ 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 [70]:
```
arr_split = np.hsplit(x , 4)       # hsplit is used to split the array into 4 eq
print(arr_split)                   # 2nd argument is shows how many splits can b
```

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

In [72]:
```
arr_split = np.hsplit(x,2)
print(arr_split)
```

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

In [73]:
```
split = np.hsplit(x, (3,4))  # (3,4) shows split the array after 3rd and 4th col
split
```

Out[73]:
```
[array([[ 1,  2,  3],
        [13, 14, 15]]),
 array([[ 4],
        [16]]),
 array([[ 5,  6,  7,  8,  9, 10, 11, 12],
        [17, 18, 19, 20, 21, 22, 23, 24]])]
```

indexing

In [74]:
```
index_arr = np.arange(0,100).reshape(10,10)
print(index_arr)
```

```
[[ 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 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

In [76]:
```
index_arr[7,5]      # prints 7th row and 5th col value
```

Out[76]:
```
75
```

In [77]:
```
index_arr[5,8]
```

Out[77]:
```
58
```

In [78]:
```
index_arr[::,5]       #only 5th col prints
```

Out[78]:
```
array([ 5, 15, 25, 35, 45, 55, 65, 75, 85, 95])
```

In [79]:
```
index_arr[1]
```

Out[79]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [81]:
```
index_arr[2:9:3]
```

Out[81]:
```
array([[20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89]])
```

In [82]:
```
index_arr
```

Out[82]:
```
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, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [80]:
```
index_arr[::-1]         #prints reverse matrix
```

Out[80]:
```
array([[90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9]])
```

In [83]:
```
index_arr[::-2]      #in revese matrix prints step count 2
```

Out[83]:
```
array([[90, 91, 92, 93, 94, 95, 96, 97, 98, 99],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

In [84]:
```
index_arr
```

Out[84]:
```
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, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [85]:
```
index_arr[2:4,4:7]     #2:4 tells about print values  from 2nd index to (4-1) in
                        #[4:7] tells about 4th index col to (7-1) index col valu
```

Out[85]:
```
array([[24, 25, 26],
       [34, 35, 36]])
```

In [86]:
```
index_arr[5:8,7:9]
```

Out[86]:
```
array([[57, 58],
       [67, 68],
       [77, 78]])
```

In [87]:
```
index_arr[6:9,3:9]
```

Out[87]:
```
array([[63, 64, 65, 66, 67, 68],
       [73, 74, 75, 76, 77, 78],
       [83, 84, 85, 86, 87, 88]])
```

Masking/Filtering

In [88]:
```
index_arr > 50         # prints true if the conditon is true from where the cond
                       # prints the false if the condition is not satisfied
```

Out[88]:
```
array([[False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True]])
```

In [89]:
```
index_arr <50
```

Out[89]:
```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False]])
```

In [90]:
```
index_arr !=50
```

Out[90]:
```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [False,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True]])
```

In [91]:
```
index_arr == 50
```

Out[91]:
```
array([[False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [ True, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False],
       [False, False, False, False, False, False, False, False, False,
        False]])
```

Binary operations

In [97]:
```
a = np.array([1,2,3])
b = np.array([4,5,6])
type(a)
```

Out[97]:
```
numpy.ndarray
```

In [99]:
```
add1 = np.array( a + b)
print(add1)
```

```
[5 7 9]
```

In [101…
```
sub = np.array( a - b)
print(sub)
```

```
[-3 -3 -3]
```

In [102…
```
mul = np.array( a * b)
print(mul)
```

```
[ 4 10 18]
```

in 2d array to add data use sum()

In [103…
```
d2_arr = np.array([1,2,3,4])
print(d2_arr)
```

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

In [104…
```
d2_arr.sum()
```

Out[104… 10

In [106…
```
d2_arr = np.array([[1,2],[3,4]])
print(d2_arr)
```

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

In [107…
```
d2_arr.sum(axis = 1)   # addition done with 2 columns
```

Out[107… array([3, 7])

In [110…
```
d2_arr.sum(axis = 0)  # addition done with 2 rows
```

Out[110… array([4, 6])

In [ ]: