In [1]:
```
!pip install numpy
```
```
Requirement already satisfied: numpy in c:\users\sachin\anaconda3\lib\site-
packages (1.24.3)
```

In [3]:
```
#importing
import numpy as np
```

In [58]:
```
a=np.array((1,2,3))
print(a)
```
```
[1 2 3]
```

In [62]:
```
a=np.array(((1,2,3),(11,22,44)))
print(a)
```
```
[[ 1  2  3]
 [11 22 44]]
```

In [24]:
```
a=np.array([[[1,2,3],[11,22,44], [1,4,6]]])
print(a)
```
```
[[[ 1  2  3]
  [11 22 44]
  [ 1  4  6]]]
```

In [37]:
```
#1 dimensional array
a=np.array([[[1, 2, 3,3], [11, 22, 44,4], [1, 4, 6,7], [1, 3, 6,8]]])
print(a)
```
```
[[[ 1  2  3  3]
  [11 22 44  4]
  [ 1  4  6  7]
  [ 1  3  6  8]]]
```

In [44]:
```
import numpy as np

#2dimensional array
a = np.array([[[[1, 2, 3], [11, 22, 44]], [[1, 4, 6], [1, 3, 6]], [[1,3,5],
print(a)
```
```
[[[[ 1  2  3]
   [11 22 44]]

  [[ 1  4  6]
   [ 1  3  6]]

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

In [48]:
```
#3dimensional array
a = np.array([[[[1, 2, 3], [11, 22, 44], [1, 4, 6]], [[1, 3, 6], [1,3,5], [2
print(a)
```

```
[[[[ 1  2  3]
   [11 22 44]
   [ 1  4  6]]

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

In [54]:
```
a=np.zeros((2,4))
print(a)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

In [52]:
```
import numpy as np

a = np.zeros((2, 4))
print(a)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

In [56]:
```
a=np.ones((3,6))
print(a)
```

```
[[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

In [12]:
```
#Create an array of evenly spaced values (step value)
# here 10 is the start and 50 is the end value, and 3 is the step value elem
d = np.arange(5,100,3)
print(d)
```

```
[ 5  8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74
 77 80 83 86 89 92 95 98]
```

In [13]:
```
#Create an array of evenlyspaced values (number of samples)
print(np.linspace(0,8,8))
```

```
[0.         1.14285714 2.28571429 3.42857143 4.57142857 5.71428571
 6.85714286 8.        ]
```

In [14]:
```
#Create a constant array
#Create a 1D array
e = np.full((2),9)
print(e)
print(e.ndim) #to find no of dimension
```

```
[9 9]
1
```

In [16]:
```
e = np.full((3),7)
print(e)
print(e.ndim)
```

```
[7 7 7]
1
```

In [19]:
```
#Create a 2D array
e = np.full((4,3),9)
print(e)
print(e.ndim)
```

```
[[9 9 9]
 [9 9 9]
 [9 9 9]
 [9 9 9]]
2
```

In [21]:
```
#Create a 3D array
e = np.full((4,3,5),9)
print(e)
print(e.ndim)
```

```
[[[9 9 9 9 9]
  [9 9 9 9 9]
  [9 9 9 9 9]]

 [[9 9 9 9 9]
  [9 9 9 9 9]
  [9 9 9 9 9]]

 [[9 9 9 9 9]
  [9 9 9 9 9]
  [9 9 9 9 9]]

 [[9 9 9 9 9]
  [9 9 9 9 9]
  [9 9 9 9 9]]]
3
```

In [23]:
```
#Create a 4D array
e = np.full((4,3,2,4),9)
print(e)
print(e.ndim)
```

```
[[[[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]]


 [[[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]]


 [[[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]]


 [[[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]

  [[9 9 9 9]
   [9 9 9 9]]]]
4
```

In [25]:
```
#Create a 2X2 identity matrix it prints only 0's and 1's
f = np.eye(5)
print(f)
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

In [30]:
```
#Create an array with random values
a=np.random.random((3,2))
print(a)
```

```
[[0.11641737 0.6221033 ]
 [0.6485375  0.89456566]
 [0.69709806 0.9421757 ]]
```

In [38]:
```
#to find shape of an Array
w = np.array([[1,2,3,4,5],
              [5,6,7,8,6],
              [5,0,7,8,7],
              [2,3,5,6,9]])
print(w.shape) #output is (4, 5)

#Output explaination
# 4 denotes no of ROWS
# 5 Denotes no of Columns

# Size: it prints the number of elements in a array
print(w.size)   # out put is 20

#To Find len of array
print(len(w))
#output explaintion
# It Will count only the rows
#output: 4
```

```
(4, 5)
20
4
```

In [47]:
```
# Access Array Elements

b = np.array([1,2,3,4,5,4,6])

print(b[4])

#You can access an array element by referring to its index number.
#The indexes in NumPy arrays start with 0,
#meaning that the first element has index 0
```

```
5
```

In [49]:
```
# Adding two Index position
b = np.array([8,4,3,4,9,8,55])

print(b[2]+b[5]+b[4]) # addition of index positions
print(b[0]*b[3]) # multipliction of index positions
print(b[0]**b[3]) # exponents of the index positions
```

```
20
32
4096
```

In [53]:
```
#Access 2-D Arrays
#Access the element on the first row, second column:
p = np.array([[1,2,3,4,5],
              [6,7,8,9,10],
              [1,2,3,4,5]]
              )

print('2nd element on 1st row: ', p[0, 1])
print(p[2,3])
```

```
2nd element on 1st row:  2
4
```

In [55]:
```
#Access 3-D Arrays

arr = np.array([[[1,2,3], [4,5,6]],
                [[7,8,9], [10,11,12]],
                [[3,4,5], [2,4,6,]]])

#Access by index
print(arr[2, 1, 1])
```

```
4
```

In [6]:
```
# Addition of two Arrays

x=[3,5,7]
y=[3,4,5]
print(x+y)
print(x*y)
```

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

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 6
      4 y=[3,4,5]
      5 print(x+y)
----> 6 print(x*y)

TypeError: can't multiply sequence by non-int of type 'list'
```

Data types

In [13]:
```
#string
a = np.array([1, 2, 3, 4])
print(a)
print(a.dtype)
```

```
[1 2 3 4]
int32
```

In [15]:
```
a = np.array([1.0, 2.2, 3.5, 4.7])
print(a)
print(a.dtype)
```
```
[1.  2.2 3.5 4.7]
float64
```

In [16]:
```
a = np.array([1.0, 2.2, 3.5, 4.7], dtype="S")
print(a)
print(a.dtype) # elements will be stored as a bit wise
```
```
[b'1.0' b'2.2' b'3.5' b'4.7']
|S3
```

In [20]:
```
a = np.array([1.0, 2.2, 3.5, 4.7], dtype='i')
print(a)
print(a.dtype)
```
```
[1 2 3 4]
int32
```

In [26]:
```
a = np.array([1.0, 2.2, 3.5, 4.7])
b=arr.astype('i')
print(b)
print(b.dtype)
```
```
[1 2 3 4]
int32
```

In [30]:
```
#Change data type from integer to boolean:
b = np.array([1, 4.8, 0.2, 0.0,3])
x= b.astype(bool)
print(x)
print(x.dtype)
```
```
[ True  True  True False  True]
bool
```

In [31]:
```
#copy method
a=np.array([1,2,3,4,5,9,8,7])
b=a.copy()
print(b)
```
```
[1 2 3 4 5 9 8 7]
```

In [33]:
```

a=np.array([1,2,3,4,5,9,8,7])
b=a.copy()
a[2]=143  # it replaces the element at the index 2 and give output as [1,2,1
print(b)
print(a)
```
```
[1 2 3 4 5 9 8 7]
[  1   2 143   4   5   9   8   7]
```

In [34]:
```
#view method
a=np.array([1,2,3,4,5,9,8,7])
b=a.view()
a[2]=143  # it replaces the element at the index 2 and give output as [1,2,1
print(b)
print(a)
```

```
[  1   2 143   4   5   9   8   7]
[  1   2 143   4   5   9   8   7]
```

In [40]:
```
# array shape
a=[2, 2, 3, 4, 5]
b=[2, 2, 3, 4, 5]
c = np.array([a, b])
print(c.shape)
```

```
(2, 5)
```

In [43]:
```
#Reshaping arrays
#Converting the following 1-D array to 2-D array.

b = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
c = b.reshape(4, 5)
d= b.reshape(5,4)
print(d)
print(c)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]
 [17 18 19 20]]
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

In [45]:
```
# converting 1d array to 3d array
b = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
c=b.reshape(5,2,2)
print(c)
```

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

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]

 [[13 14]
  [15 16]]

 [[17 18]
  [19 20]]]
```

In [54]:
```
# converting 2d array to 3d array
b = np.array([[1,2,3,4],
             [5,6,7,8],
             [9,10,11,12],
             [13,14,15,16],
             [17,18,19,20]])
c=b.reshape(5,2,2)
print(c)
```

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

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]

 [[13 14]
  [15 16]]

 [[17 18]
  [19 20]]]
```

In [50]:
```
import numpy as np

# Example 2D array
array_2d = np.array([[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]])

# Reshape the 2D array to a 3D array
array_3d = array_2d.reshape(1, array_2d.shape[0], array_2d.shape[1])

print("2D Array:")
print(array_2d)
print("Shape:", array_2d.shape)
print("\n3D Array:")
print(array_3d)
print("Shape:", array_3d.shape)
```

```
2D Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)

3D Array:
[[[1 2 3]
  [4 5 6]
  [7 8 9]]]
Shape: (1, 3, 3)
```

In [64]:
```
#Slice elements from index 1 to index 5 from the following array:

a = np.array([1, 2, 3, 4, 5, 6, 7,9,8])
print(a[2:8])  # gives elems from index 2 to n-1
print(a[1:]) #gives elements from index 1 to the end of the array
print(a[:3]) #Slice elements from the beginning to index 3(n-1) (not include
```

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

In [71]:
```
# Negative Slicing
a = np.array([1, 2, 3, 4, 5, 6, 7,9,8])
print(a[-3:-1]) #gives output as index of -3 and -1
```

```
[7 9]
```

In [73]:
```
#slicing by using step
b = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 4, 1])
print(b[1:6:2]) #step is apply like step-1
```

```
[2 4 6]
```

In [77]:
```
#Return entire element from the entire array:
b = np.array([11, 22, 33, 44, 55, 66, 77])
print(b[::2])
```

```
[11 33 55 77]
```

In [86]:
```
#Slicing 2-D Arrays
#From the second element, slice elements from index 1 to index 4 (not includ
a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(a[1, 1:4])
```

```
[7 8 9]
```

In [96]:
```
#return both element index 3
arr = np.array ([[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10],[11,12,13,14,15]]])
print(arr[0:3,2])
```

```
[[11 12 13 14 15]]
```

In [ ]: