In [1]:
```
# 1. Create an array using Numpy.
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr)
```

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

In [2]:
```
# 2. Create more than one dimensions array using Numpy.
arr2 = np.array([[1,2,3],[4,5,6]])
print(arr2)
print(arr2.ndim)
```

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

In [3]:
```
# 3. Create minimum dimensions array using Numpy.
arr3 = np.array([1,2,3])
print(arr3)
print(arr3.ndim)
```

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

In [4]:
```
# 4. Check the data type of following array using Numpy
type1 = np.array([1, 2, 3, 4, 5, 6])
type2 = np.array([1.5, 2.5, 0.5, 6])
type3 = np.array(['a', 'b', 'c'])
type4 = np.array(["Canada", "Australia"], dtype='U5')
type5 = np.array([555, 666], dtype=float)
print("Data type of Array 1 :-",type1.dtype)
print("Data type of Array 2 :-",type2.dtype)
print("Data type of Array 3 :-",type3.dtype)
print("Data type of Array 4 :-",type4.dtype)
print("Data type of Array 5 :-",type5.dtype)
```

```
Data type of Array 1 :- int64
Data type of Array 2 :- float64
Data type of Array 3 :- <U1
Data type of Array 4 :- <U5
Data type of Array 5 :- float64
```

In [5]:
```
# 5. Check the following array shape using Numpy
array1d = np.array([1, 2, 3, 4, 5, 6])
array2d = np.array([[1, 2, 3], [4, 5, 6]])
array3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print("Shape of Array 1 is -",array1d.shape)
print("Shape of Array 2 is -",array2d.shape)
print("Shape of Array 3 is -",array3d.shape)
```

```
Shape of Array 1 is - (6,)
Shape of Array 2 is - (2, 3)
Shape of Array 3 is - (2, 2, 3)
```

In [6]:
```
# 6. Use the ndim method to determine the dimension of NumPy array.
arr4 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(arr4)
print("Dimension of the array is -",arr4.ndim)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Dimension of the array is - 2
```

In [7]:
```
# 7. Use the resize and reshape method on Numpy array.
arr5 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
print("Original Array is -",arr5)
print("Using Reshape -\n",arr5.reshape(3,4))
print("Using Resize -\n",np.resize(arr5,(3,3)))
```

```
Original Array is - [ 1  2  3  4  5  6  7  8  9 10 11 12]
Using Reshape -
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Using Resize -
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [8]:
```
# 8. Create the Program to Transform List or Tuple into NumPy array.
list1 = [1,2,3,4,5]
print(type(list1))
print(list1)
```

```
arr6 = np.asarray(list1)
print(type(arr6))
print(arr6)

tuple1 = ([1,2,3],[4,5,6])
print(type(tuple1))
print(tuple1)

arr7 = np.asarray(tuple1)
print(type(arr7))
print(arr7)
```

```
<class 'list'>
[1, 2, 3, 4, 5]
<class 'numpy.ndarray'>
[1 2 3 4 5]
<class 'tuple'>
([1, 2, 3], [4, 5, 6])
<class 'numpy.ndarray'>
[[1 2 3]
 [4 5 6]]
```

In [9]:

```
# 9. Perform the following Indexing Operations using Numpy array.
array1d = np.array([1, 2, 3, 4, 5, 6])
# 1. Get first value
# 2. Get last value
# 3. Get 4th value from first
# 4. Get 5th value from last
# 5. Get multiple values.
print("Original Array is -",array1d)
print("First value is -",array1d[0])
print("Last value is -",array1d[-1])
print("4th Value from start -",array1d[3])
print("5th value from last -",array1d[-5])
print("Multiple Values -",array1d[1::2])
```

```
Original Array is - [1 2 3 4 5 6]
First value is - 1
Last value is - 6
4th Value from start - 4
5th value from last - 2
Multiple Values - [2 4 6]
```

In [10]:
```
# 10. Perform the following Indexing Operations using Numpy array.
array2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# 1. Get first row first col
# 2. Get first row second col
# 3. Get third row second col
# 4. Get second row second col
print("Original Array is -\n",array2d)
print("First Row first column -",array2d[0, 0])
print("First Row second column -",array2d[0, 1])
print("Third Row second column -",array2d[2, 1])
print("Second Row second column -",array2d[1,1])
```

```
Original Array is -
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
First Row first column - 1
First Row second column - 2
Third Row second column - 8
Second Row second column - 5
```

In [11]:
```
# 11. Perform the following Indexing Operations using Numpy array.
array3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print("Original Array is -\n",array3d)
print("\nSecond Row third column -",array3d[0,1,2])
print("Third Row first column -",array3d[1,0,0])
print("Fourth Row second column -",array3d[1,1,1])
```

```
Original Array is -
 [[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]

Second Row third column - 6
Third Row first column - 7
Fourth Row second column - 11
```

In [12]:
```
# 12. Perform the following Single Dimensional Slicing Operations using Numpy array.
# 1. from index 4 to last index
# 2. From index 0 to 4 index
# 3. From index 4(included) up to index 7(excluded)
# 4. Excluded last element
```

```
# 5. Up to second last index(negative index)
# 6. From last to first in reverse order(negative step)
# 7. All odd numbers in reversed order
# 8. All even numbers in reversed order
# 9. All elements
arr1d = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print("Orignial Array is -",arr1d)
print("from index 4 to last index -",arr1d[4:])
print("From index 0 to 4 index -",arr1d[0:5])
print("From index 4(included) up to index 7(excluded) -",arr1d[4:7])
print("Excluded last element -",arr1d[0:9])
print("Up to second last index(negative index) -",arr1d[-10:-1])
print("From last to first in reverse order(negative step) -",arr1d[-1::-1])
print("All odd numbers in reversed order -",arr1d[-1::-2])
print("All even numbers in reversed order -",arr1d[-2::-2])
print("All elements -",arr1d)
```

```
Orignial Array is - [0 1 2 3 4 5 6 7 8 9]
from index 4 to last index - [4 5 6 7 8 9]
From index 0 to 4 index - [0 1 2 3 4]
From index 4(included) up to index 7(excluded) - [4 5 6]
Excluded last element - [0 1 2 3 4 5 6 7 8]
Up to second last index(negative index) - [0 1 2 3 4 5 6 7 8]
From last to first in reverse order(negative step) - [9 8 7 6 5 4 3 2 1 0]
All odd numbers in reversed order - [9 7 5 3 1]
All even numbers in reversed order - [8 6 4 2 0]
All elements - [0 1 2 3 4 5 6 7 8 9]
```

In [13]:

```
# 13. Perform the following Multidimensional Slicing Operations using Numpy array.
# 1. 2nd and 3rd col
# 2. 2nd and 3rd row
# 3. Reverse an array
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("orignial Array -\n",arr2d)
print("2nd and 3rd col -\n",arr2d[:,[1,2]])
print("2nd and 3rd row -\n",arr2d[1:])
print("Reverse an array -\n",arr2d[-1::-1])
```

```
orignial Array -
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
2nd and 3rd col -
 [[2 3]
 [5 6]
 [8 9]]
2nd and 3rd row -
 [[4 5 6]
 [7 8 9]]
Reverse an array -
 [[7 8 9]
 [4 5 6]
 [1 2 3]]
```

In [14]:

```
# 14. Perform the following operations to Manipulating the Dimensions and the Shape of
# Arrays(Flips the order of the Axes)
# 1. Permute the dimensions of an array
# 2. Flip array in the left/right direction
# 3. Flip array in the up/down direction
# 4. Rotate an array by 90 degrees in the plane specified by axes
array2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("Orignial Array is -\n",array2d)
print("Permute the dimensions of an array -\n",np.transpose(array2d))
print("Flip array in the left/right direction -\n",np.fliplr(array2d))
print("Flip array in the up/down direction -\n",np.flipud(array2d))
print("Rotate an array by 90 degrees in the plane specified by axes -\n",np.rot90(array2d))
```

```
Orignial Array is -
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Permute the dimensions of an array -
 [[1 4 7]
 [2 5 8]
 [3 6 9]]
Flip array in the left/right direction -
 [[3 2 1]
 [6 5 4]
 [9 8 7]]
Flip array in the up/down direction -
 [[7 8 9]
 [4 5 6]
 [1 2 3]]
Rotate an array by 90 degrees in the plane specified by axes -
 [[3 6 9]
 [2 5 8]
 [1 4 7]]
```

In [15]:

```
# 15.Perform the following operations to Manipulating the Dimensions and the Shape
# of Arrays(Joining and Stacking)
# 1. Stack arrays in sequence horizontally (column wise).
# 2. Stack arrays in sequence vertically (row wise)
# 3. Stack arrays in sequence depth wise (along third axis)
# 4. Appending arrays after each other, along a given axis
# 5. Append values to the end of an array
array1 = np.array([[1, 2, 3], [4, 5, 6]])
array2 = np.array([[7, 8, 9], [10, 11, 12]])
print("Original Array 1 is -\n",array1)
print("Original Array 2 is -\n",array2)
print("Stack arrays in sequence horizontally (Column wise) -\n",np.hstack((array1, array2)))
print("Stack arrays in sequence vertically (row wise) -\n",np.vstack((array1,array2)))
print("Stack arrays in sequence depth wise (along third axis) -\n",np.dstack((array1,array2)))
print("Appending arrays after each other, along a given axis -\n",np.concatenate((array1,array2)))
print("Append values to the end of an array -\n",np.append(array1,array2,axis=0))
```

```
Original Array 1 is -
 [[1 2 3]
 [4 5 6]]
Original Array 2 is -
 [[ 7  8  9]
 [10 11 12]]
Stack arrays in sequence horizontally (Column wise) -
 [[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
Stack arrays in sequence vertically (row wise) -
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
Stack arrays in sequence depth wise (along third axis) -
 [[[ 1  7]
  [ 2  8]
  [ 3  9]]

 [[ 4 10]
  [ 5 11]
  [ 6 12]]]
Appending arrays after each other, along a given axis -
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
Append values to the end of an array -
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

In [16]:

```
# 16.Perform the following Arithmetic Operations using Numpy Array.
# 1. array1 + array2
# 2. array1 - array2
# 3. array1 * array2
# 4. array2 / array1
# 5. array1 ** array2
array1 = np.array([[1, 2, 3], [4, 5, 6]])
array2 = np.array([[7, 8, 9], [10, 11, 12]])
print("Original Array 1 -\n",array1)
print("Original Array 2 -\n",array2)
print("Array 1 + Array 2 -\n",array1+array2)
print("Array 1 - Array 2 -\n",array1-array2)
```

```
print("Array 1 * Array 2 -\n",array1*array2)
print("Array 1 / Array 2 -\n",array1/array2)
print("Array 1 ** Array 2 -\n",array1**array2)
```

```
Original Array 1 -
 [[1 2 3]
 [4 5 6]]
Original Array 2 -
 [[ 7  8  9]
 [10 11 12]]
Array 1 + Array 2 -
 [[ 8 10 12]
 [14 16 18]]
Array 1 - Array 2 -
 [[-6 -6 -6]
 [-6 -6 -6]]
Array 1 * Array 2 -
 [[ 7 16 27]
 [40 55 72]]
Array 1 / Array 2 -
 [[0.14285714 0.25       0.33333333]
 [0.4        0.45454545 0.5       ]]
Array 1 ** Array 2 -
 [[         1        256      19683]
 [   1048576   48828125 2176782336]]
```

In [17]:

```
# 17.Perform the following Scalar Arithmetic Operations using Numpy Array.
# 1. array1 + 2
# 2. array1 – 5
# 3. array1 * 2
# 4. array1 / 5
# 5. array1 ** 2
array1 = np.array([[10, 20, 30], [40, 50, 60]])
print("Original Array is -\n",array1)
print("Array 1 + (2) =\n",array1+2)
print("Array 1 - (5) =\n",array1-5)
print("Array 1 * (2) =\n",array1*2)
print("Array 1 / (5) =\n",array1/5)
print("Array 1 ** (2) =\n",array1**2)
```

```
Original Array is -
 [[10 20 30]
 [40 50 60]]
Array 1 + (2) =
 [[12 22 32]
 [42 52 62]]
Array 1 - (5) =
 [[ 5 15 25]
 [35 45 55]]
Array 1 * (2) =
 [[ 20  40  60]
 [ 80 100 120]]
Array 1 / (5) =
 [[ 2.  4.  6.]
 [ 8. 10. 12.]]
Array 1 ** (2) =
 [[ 100  400  900]
 [1600 2500 3600]]
```

In [18]:

```
# 18.Perform the following Elementary Mathematical Functions using Numpy Array.
# 1. sin(array1)
# 2. cos(array1)
# 3.tan(array1)
# 4. sqrt(array1)
# 5. exp(array1)
# 6. log10(array1)
array1 = np.array([[10, 20, 30], [40, 50, 60]])
print("Original Array is -\n",array1)
print("sin(array1) -\n",np.sin(array1))
print("cos(array1) -\n",np.cos(array1))
print("tan(array1) -\n",np.tan(array1))
print("sqrt(array1) -\n",np.sqrt(array1))
print("exp(array1) -\n",np.exp(array1))
print("log10(array1) -\n",np.log10(array1))
```

```
Original Array is -
 [[10 20 30]
 [40 50 60]]
sin(array1) -
 [[-0.54402111  0.91294525 -0.98803162]
 [ 0.74511316 -0.26237485 -0.30481062]]
cos(array1) -
 [[-0.83907153  0.40808206  0.15425145]
 [-0.66693806  0.96496603 -0.95241298]]
tan(array1) -
 [[ 0.64836083  2.23716094 -6.4053312 ]
 [-1.11721493 -0.27190061  0.32004039]]
sqrt(array1) -
 [[3.16227766 4.47213595 5.47722558]
 [6.32455532 7.07106781 7.74596669]]
exp(array1) -
 [[2.20264658e+04 4.85165195e+08 1.06864746e+13]
 [2.35385267e+17 5.18470553e+21 1.14200739e+26]]
log10(array1) -
 [[1.         1.30103    1.47712125]
 [1.60205999 1.69897    1.77815125]]
```

In [19]:

```
# 19.Perform the following Element-wise Mathematical Operations using Numpy Array.
# 1. Addition of array1 and array2
# 2. Multiplication of array1 and array2
# 3. Power of array1 and array2
array1 = np.array([[10, 20, 30], [40, 50, 60]])
array2 = np.array([[2, 3, 4], [4, 6, 8]])
array3 = np.array([[-2, 3.5, -4], [4.05, -6, 8]])
print("Addition of Array 1 and Array 2 -\n",np.add(array1,array2))
print("Multiplication of Array 1 and Array 2 is -\n",np.multiply(array1, array2))
print("Power of Array1 and Array2 is -\n",np.power(array1,array2))
```

```
Addition of Array 1 and Array 2 -
 [[12 23 34]
 [44 56 68]]
Multiplication of Array 1 and Array 2 is -
 [[ 20  60 120]
 [160 300 480]]
Power of Array1 and Array2 is -
 [[           100           8000         810000]
 [       2560000     15625000000 1679616000000000]]
```

In [20]:

```
# 20.Perform the following Aggregate and Statistical Functions using Numpy Array.
```

```
# 1. Mean
# 2. Standard deviation
# 3. Variance
# 4. Sum of array elements
array1 = np.array([[10, 20, 30], [40, 50, 60]])
print("Mean of the Array 1 is -",np.mean(array1))
print("Standard Variance of the Array 1 is -",np.std(array1))
print("Variance of the Array 1 is -",np.var(array1))
print("Sum of the Array 1 is -",np.sum(array1))
```

```
Mean of the Array 1 is - 35.0
Standard Variance of the Array 1 is - 17.07825127659933
Variance of the Array 1 is - 291.6666666666667
Sum of the Array 1 is - 210
```

In [21]:

```
# Use the Where(), Select() and Choose() function to identify the element is less than 4, mul by 2 else by 3
arr1 = np.array([[1, 2, 3], [4, 5, 6]])
print("Original Array is -\n",arr1)
print("Using Where() method -\n",np.where(arr1 < 4, arr1 * 2, arr1 * 3))
# print("Using Select() method -\n",np.select([arr1 < 4, arr1], [arr1 * 2, arr1 * 3]))
# print("Using Choose() method -\n",np.choose())
```

```
Original Array is -
 [[1 2 3]
 [4 5 6]]
Using Where() method -
 [[ 2  4  6]
 [12 15 18]]
```

In [22]:

```
# 22.Perform the following Logical Operations using Numpy Array.
# 1. logical_or(Condition array<10, array>15)
# 2. logical_and(Condition array<10, array>15)
# 3. logical_not(Condition array<20)
thearray = np.array([[10, 20, 30], [14, 24, 36]])
print("Original Array is -\n",thearray)
print("Array Elements <10 and >15 using logical or opr are -\n",np.logical_or(thearray < 10, thearray > 15))
print("Array Elements <10 and >15 using logical and opr are -\n",np.logical_and(thearray < 10, thearray > 15
print("Array Elements <20 using logical not are -\n",np.logical_not(thearray < 20))
```

```
Original Array is -
 [[10 20 30]
 [14 24 36]]
Array Elements <10 and >15 using logical or opr are -
 [[False  True  True]
 [False  True  True]]
Array Elements <10 and >15 using logical and opr are -
 [[False False False]
 [False False False]]
Array Elements <20 using logical not are -
 [[False  True  True]
 [False  True  True]]
```

In [23]:

```
# 23.Perform the following Standard Set Operations using Numpy Array.
array1 = np.array([[10, 20, 30], [14, 24, 36]])
array2 = np.array([[20, 40, 50], [24, 34, 46]])
# 1. Find the union of two arrays
# 2. Find the intersection of two arrays
# 3. Find the set difference of two arrays
print("Original Array 1 is -\n",array1)
print("Original Array 2 is -\n",array2)
print("Union of the two arrays are -\n",np.union1d(array1,array2))
print("Intersection of two arrays -\n",np.intersect1d(array1,array2))
print("Set Difference of two  arrays are -\n",np.setdiff1d(array1,array2))
```

```
Original Array 1 is -
 [[10 20 30]
 [14 24 36]]
Original Array 2 is -
 [[20 40 50]
 [24 34 46]]
Union of the two arrays are -
 [10 14 20 24 30 34 36 40 46 50]
Intersection of two arrays -
 [20 24]
Set Difference of two  arrays are -
 [10 14 30 36]
```