1. Array Creation Function

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

In [2]:
```
a = np.array([1,2,3,4])
print("Array of a :",a) # creating array from list
```

```
Array of a : [1 2 3 4]
```

In [3]:
```
b=np.arange(0,10,2)
print("Array of b is :",b)
```

```
Array of b is : [0 2 4 6 8]
```

In [4]:
```
c = np.zeros((3,3))
print("Array of c : \n",c)
```

```
Array of c :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [5]:
```
d = np.ones((3,4), dtype=int)
print("Array of d : \n",d)
```

```
Array of d :
 [[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

In [6]:
```
# create a indetity matrix
f = np.eye(4)
print("Identity matrix of f : \n",f)
```

```
Identity matrix of f :
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

In [7]:
```
g=np.eye(5,dtype=int)
print("Identity matrix of g : \n",g)
```

```
Identity matrix of g :
 [[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]]
```

2. Array Manipulation Function

In [8]:
```
a1=np.array([1,2,3,4]) # Reshape array
re=np.reshape(a1,(2,2))
print("Reshape array :\n",re)
```

```
Reshape array :
 [[1 2]
 [3 4]]
```

In [9]:
```
b1=np.array([1,2,3,4,5,6,7,8])
reshape=np.reshape(b1,(4,2))
print("Reshape array :\n",reshape)
```

```
Reshape array :
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]
```

In [10]:
```
 # Flatten an Array
f1 = np.array([[1, 2, 3], [5, 4, 6]])
flattened = np.ravel(f1)  # Flatten to 1D array
print("Flattened array:", flattened)
```

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

In [11]:
```
arr = np.array([           #Flatten array print the arran in 1d
    [10,20,30],
    [40,50,60],
    [70,80,90]
```

```
            ])
flat = np.ravel(arr)
print("Original array:\n", arr)
print('         ')
print("Flattened array:", flat)
```

```
Original array:
 [[10 20 30]
 [40 50 60]
 [70 80 90]]

Flattened array: [10 20 30 40 50 60 70 80 90]
```

In [12]:
```
arr1 = np.array ([
    [[1, 2], [3, 4]],
    [[5, 6], [7, 8]]
                ])
flatten = np.ravel(arr1)
print("Flatten array :", flatten)
```

```
Flatten array : [1 2 3 4 5 6 7 8]
```

In [13]:
```
# Transpose Array
e = np.array([[1,2], [3,4]])
transpose = np.transpose(e)
print("Transpose of Array \n",transpose)
```

```
Transpose of Array
 [[1 3]
 [2 4]]
```

In [14]:
```
a2 = np.array([1, 2])       # Stack vertically array
b2 = np.array([3, 4])
stack = np.vstack([a2,b2])
print("Stacked Array :\n", stack)
```

```
Stacked Array :
 [[1 2]
 [3 4]]
```

In [15]:
```
# Stack arrays horizontally

a2 = np.array([1, 2, 6, 8])
b2 = np.array([3, 4, 5, 9])
```

```
stacked = np.hstack([a2, b2])
print("Stacked arrays:\n", stacked)
```

```
Stacked arrays:
 [1 2 6 8 3 4 5 9]
```

Mathematical Function

In [16]:
```
p = np.array([2,4,6,8]) # >> value 2 se add hua hai
added = np.add(p,2)
print("Added 2 of p:",added)
```

```
Added 2 of p: [ 4  6  8 10]
```

In [17]:
```
squ = np.power(p,2)
print("Square of array is ",squ)
```

```
Square of array is  [ 4 16 36 64]
```

In [18]:
```
sqrt_val = np.sqrt(p)  # Square root of each element
print("Square root of p:", sqrt_val)
```

```
Square root of p: [1.41421356 2.         2.44948974 2.82842712]
```

In [19]:
```
print(a1)
```

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

In [20]:
```
# dot product
a3 = np.array([4,3,2,1])
dot = np.dot(a1,a3)
print("Dot product is ",dot)
```

```
Dot product is  20
```

4. Statistical Function

In [21]:
```
s = np.array([1,2,3,4,5,6,7,8])       # Mean = total summ / total count
mean = np.mean(s)
print("Mean of this array is :",mean)
```

```
Mean of this array is : 4.5
```

In [22]:
```
s = np.array([1, 2, 3, 4,5,6,7,8])  # variance
mean = np.var(s)
print("Mean of s:", mean)
```

```
Mean of s: 5.25
```

In [23]:
```
# Minimun value in s :
minimum = np.min(s)
print("Min of s:", minimum)
```

```
Min of s: 1
```

In [24]:
```
# Maximum value in s
maximum = np.max(s)
print("max of s :",maximum)
```

```
max of s : 8
```

5. Linear Algebra Functions

In [25]:
```
matrix = np.array([[1, 2], [3, 4]])
matrix
```

Out[25]:
```
array([[1, 2],
       [3, 4]])
```

6. Random Sampling Function

In [26]:
```
r = np.random.rand(4)
print("Random value is ",r)
```

```
Random value is  [0.82356218 0.94600443 0.23164795 0.91555533]
```

In [27]:
```
np.random.seed(0)    # seed -> same ouput
random_vals = np.random.rand(3)
print("Random values:", random_vals)
```

```
Random values: [0.5488135  0.71518937 0.60276338]
```

In [28]:
```
rand_ints = np.random.randint(0, 10, size=5)
print("Random integers:", rand_ints)
```

```
Random integers: [3 7 9 3 5]
```

In [29]:
```
np.random.seed(0)
rand_val = np.random.randint(0,10, size=5)
print("random integers", rand_val)
```

```
random integers [5 0 3 3 7]
```

7. Boolean and Logical Function

In [30]:
```
logic = np.array([True,False,True])
all_true = np.all(logic)
print("All elements True :",all_true)
```

```
All elements True : False
```

In [31]:
```
logical_test = np.array([True, False, True])
all_true = np.all(logical_test)
print("All elements True:", all_true)
```

```
All elements True: False
```

In [32]:
```
any_true = np.any(logical_test)  # any
print("Any elements True:", any_true)
```

```
Any elements True: True
```

In [33]:
```
any_true = np.any(logic)
print("any elements is true :",any_true)
```

```
any elements is true : True
```

8. Set Operation

In [34]:
```
set_a = np.array([1, 2, 3, 4])
set_b = np.array([3, 4, 5, 6])
```

```
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)
```

```
Intersection of a and b: [3 4]
```

In [35]:
```
union = np.union1d(set_a, set_b)
print("Union of a and b:", union)
```

```
Union of a and b: [1 2 3 4 5 6]
```

9. Array Attribute Function

In [36]:
```
a = np.array([1,2,3,4,5,6])

size = a.size
shape = a.shape
dimension = a.ndim
dtype = a.dtype

print("size of array is :",size)

print("shape of array is :",shape)

print("dimension of array is :",dimension)

print("data type of array is :",dtype)
```

```
size of array is : 6
shape of array is : (6,)
dimension of array is : 1
data type of array is : int64
```

10. Other Function

In [37]:
```
a = np.array([1,2,3,4]) # Copied array
copy_arr = np.copy(a)
print("Copy array :",copy_arr)
```

```
Copy array : [1 2 3 4]
```

In [38]:
```
a = [1,2,3] # b copied a but when we add some value in a
b = [4,5,6]

b = np.copy(a)
a[0] = 100

print(a)
print(b)
```

```
[100, 2, 3]
[1 2 3]
```

In [39]:
```
a1 = [1,2,3,4,5]
b1 = np.copy(a1)

a1[0]= 200

print(a1)
print(b1)
```

```
[200, 2, 3, 4, 5]
[1 2 3 4 5]
```

In [40]:
```
a2 = np.array([1,2,3,4])
array_size = a2.nbytes
print("size in bytes ", array_size)
```

```
size in bytes  32
```

In [41]:
```
# Check if two arrays share memory
shared = np.shares_memory(a, copy_arr)  # Check if arrays share memory
print("Do a and copied_array share memory?", shared)
```

```
Do a and copied_array share memory? False
```

In [ ]: