1. Array Creation Functions

In [2]:

```
import numpy as np
```

In [3]:

```
# Create an array from a list
a=np.array([1,2,3])
print("Array a:",a)
```

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

In [4]:

```
# create an array with evenly spaced values
b=np.arange(0,10,2)
print("Array b:",b)
```

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

In [5]:

```
# create an array with linearly spaced values
c=np.linspace(0,1,5)
print("Array c:",c)
```

```
Array c: [0.   0.25 0.5  0.75 1.  ]
```

In [6]:

```
# create an array filled with zeros
d=np.zeros((2,3))
print("Array d:",d)
```

```
Array d: [[0. 0. 0.]
 [0. 0. 0.]]
```

In [7]:

```
#create an array filled with ones
e=np.ones((3,2))
print("Array e:\n",e)
```

```
Array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
```

In [8]:

```
# create an identity matrix
f=np.eye(4)
print("identity matrix f:\n",f)
```

```
identity matrix f:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

2. Array Manipulation Functions

In [10]:

```
# Reshape an array
a1=np.array([1,2,3])
print(a1)
```

```
reshaped=np.reshape(a1,(1,3))
print("Reshaped array:",reshaped)
```

```
[1 2 3]
Reshaped array: [[1 2 3]]
```

In [12]:

```
#Transpose an array
e1 = np.array([[1, 2], [3, 4]])
print(e1)
transposed = np.transpose(e1) # Transpose the array
print("Transposed array:\n", transposed)
```

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

In [13]:

```
# stack arrays vertically
a2=np.array([1,2])
b2=np.array([3,4])
print(a2)
print(b2)
stacked=np.vstack([a2,b2])
print("stacked array:\n",stacked)
```

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

3. Mathematical Functions

In [17]:

```
# add two arrays
g=np.array([1,2,3,4])
print(g)
added=np.add(g,2)
print("Added 2 to g:",added)
```

```
[1 2 3 4]
Added 2 to g: [3 4 5 6]
```

In [21]:

```
# square each element
squared=np.power(g,2)
print(squared)
print("Squared g:",squared)
```

```
[ 1  4  9 16]
Squared g: [ 1  4  9 16]
```

In [22]:

```
# square root of each element
sqrt_val=np.sqrt(g)
print(sqrt_val)
print("square root of g:",sqrt_val)
```

```
[1.         1.41421356 1.73205081 2.        ]
square root of g: [1.         1.41421356 1.73205081 2.        ]
```

In [23]:

```
print(a1)
print(g)
```

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

Dot product of two arrays

a2=np.array([1,2,3]) dot_product=np.dot(a2,g) print("Dot product of a and g:",dot_product)

In [27]:

```
print(a)
print(a1)
```

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

In [28]:

```
a3=np.array([1,2,3])
dot_product=np.dot(a1,a)
print("Dot product of a1 and a:",dot_product)
```

```
Dot product of a1 and a: 14
```

4. Statistical Functions

In [30]:

```
s=np.array([1,2,3,4])
mean=np.mean(s)
print("mean of s:",mean)
```

```
mean of s: 2.5
```

In [31]:

```
# standard devieation of an array
std_dev=np.std(s)
print("standard deviation of s:",std_dev)
```

```
standard deviation of s: 1.118033988749895
```

In [32]:

```
# minimum elements of an array
minimum=np.min(s)
print("min of s:",minimum)
```

```
min of s: 1
```

In [33]:

```
# maximum element of an array
maximum=np.max(s)
print("max of s:",maximum)
```

```
max of s: 4
```

5. Linear Algebra Functions

In [34]:

```
# create a matrix
matrix=np.array ([[1,2],[3,4]])
```

In [35]:

```
# Determinant of a matrix
determinant=np.linalg.det(matrix)
print("Determinant of matrix:",determinant)
```

```
Determinant of matrix: -2.0000000000000004
```

In [37]:

```
# Inverse=np.linalg.inv(matrix)
inverse=np.linalg.inv(matrix)
print("Inverse of matrix:\n",inverse)
```

```
Inverse of matrix:
 [[-2.   1. ]
 [ 1.5 -0.5]]
```

6. Random Sampling Functions

In [39]:

```
# generate random values between 0 and 1
random_vals=np.random.rand(3)
print("random values:",random_vals)
```

```
random values: [0.23238457 0.77635509 0.72762976]
```

In [40]:

```
# set seed for reproducibility
np.random.seed(0)
# generate random values between 0 and 1
random_vals=np.random.rand(3)
print("Random values:",random_vals)
```

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

In [41]:

```
# Generate random inteders
rand_ints=np.random.randint(0,10,size=5)
print("Random integers:",rand_ints)
```

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

In [42]:

```
# set seed for reproducibility
np.random.seed(0)

#Generate random integers
rand_ints=np.random.randint(0,10,size=5)
print("Random integers:",rand_ints)
```

```
Random integers: [5 0 3 3 7]
```

7. Boolean & Logical Functions

In [43]:

```
# check if all elements are True
# all
logical_test=np.array([True,False,True])
```

```
all_true=np.all(logical_test)
print("All elements True:",all_true)
```

```
All elements True: False
```

In [44]:

```
# check if all elements are True
logical_test=np.array([True,False,True])
all_true=np.all(logical_test)
print("All elemeb=nts True:",all_true)
```

```
All elemeb=nts True: False
```

In [45]:

```
# check if all elements are True
logical_test=np.array([False,False,False])
all_true=np.all(logical_test)
print("All elemeb=nts True:",all_true)
```

```
All elemeb=nts True: False
```

In [46]:

```
# check if any elements are true
# any
any_true=np.any(logical_test)
print("Any elements true:",any_true)
```

```
Any elements true: False
```

8. Set Operations

In [53]:

```
# Intersection of two arrays
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 [54]:

```
# Union of two arrays
union=np.union1d(set_a,set_b)
print("union if a and b:",union)
```

```
union if a and b: [1 2 3 4 5 6]
```

9. Array Attribute Functions

In [57]:

```
# Array attributes
a=np.array ([1,2,3])
shape=a.shape
size=a.size
dimensions=a.ndim
dtype=a.dtype
print("shape of a:",shape)
print("size of a:",size)
print("number of dimensions of a:",dimensions)
print("data type of a:",dtype)
```

```
shape of a: (3,)
size of a: 3
number of dimensions of a: 1
data type of a: int64
```

10. Other Functions

In [58]:

```
# create a copy of an array
a=np.array([1,2,3])
copied_array=np.copy(a)
print("copied array :",copied_array)
```

```
copied array : [1 2 3]
```

In [59]:

```
# size in byte of an array
array_size_in_bytes=a.nbytes
print("size of a in bytes:",array_size_in_bytes)
```

```
size of a in bytes: 24
```

In [60]:

```
# check if two arrays share memory
shared=np.shares_memory(a,copied_array)
print("Do a and copied_array share memory?",shared)
```

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