1.Array creation functions

In [51]:

```
import numpy as np
```

In [52]:

```
a=np.array([1,2,3])
print("array a",a)
```

```
array a [1 2 3]
```

In [53]:

```
b=np.arange(0,10,2)
print("array b:",b)
```

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

In [54]:

```
c=np.linspace(0,1,5)
print("array c:",c)
```

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

In [55]:

```
d=np.zeros((2,3))
print("array d:\n",d)
```

```
array d:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

In [56]:

```
e=np.ones((3,2))
print("array e:\n",e)
```

```
array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
```

In [57]:

```
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 [58]:

```
a1=np.array([1,2,3])
reshaped=np.reshape(a1,(1,3))
print("Reshaped array:",reshaped)
```

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

In [59]:

```
f1=np.array([[1,2],[3,4]])
flattened=np.ravel(f1)
```

```
print("Flattened array:",flattened)
```

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

In [60]:
```
e1=np.array([[1,2],[3,4]])
transposed=np.transpose(e1)
print("Transport array:\n",transposed)
```

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

3.Mathyematical functions

In [61]:
```
g=np.array([1,2,3,4])
added=np.add(g,2)
print("Added 2 to g:",added)
```

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

In [62]:
```
squared=np.power(g,2)
print("squared g:",squared)
```

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

In [63]:
```
sqrt_val=np.sqrt(g)
print("Square root of g:",sqrt_val)
```

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

In [64]:
```
print(a1)
print(g)
```

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

In [65]:
```
a2=np.array([1,2,3])
dot_product=np.dot(a2,g)
print("DOt product of a and g:",dot_product)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[65], line 2
      1 a2=np.array([1,2,3])
----> 2 dot_product=np.dot(a2,g)
      3 print("DOt product of a and g:",dot_product)

ValueError: shapes (3,) and (4,) not aligned: 3 (dim 0) != 4 (dim 0)
```

In [66]:
```
print(a)
print(a1)
```

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

In [67]:
```
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.Statisticccal functions

In [68]:

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

```
mean of s: 2.5
```

In [69]:

```
std_dev=np.std(s)
print("standard deviaton of s:",std_dev)
```

```
standard deviaton of s: 1.118033988749895
```

In [70]:

```
minimum=np.min(s)
print("Min of s:",minimum)
```

```
Min of s: 1
```

In [71]:

```
maximum=np.max(s)
print("Max of s:",maximum)
```

```
Max of s: 4
```

5.Linear Algebra Functions

In [72]:

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

In [73]:

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

```
Determination of matrix: -2.0000000000000004
```

In [74]:

```
# Inverse of a 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 [75]:

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

```
Random values: [0.4236548  0.64589411 0.43758721]
```

In [76]:

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

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

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

In [77]:

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

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

In [78]:

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

# Genarate 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 [79]:

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

```
All elements True: False
```

In [80]:

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

```
 elements True: True
```

In [81]:

```
# check if all elements are true
logical_test=np.array([logical_test]) # check if all are True
print("Any elements True:",any_true)
```

```
Any elements True: True
```

In [82]:

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

```
Any elements True: False
```

8.Set Oparations

In [83]:

```python
# 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 [84]:

```python
# union of two arrays
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 functions

In [85]:

```python
# Array attributes
a=np.array([1,2,3])
shape=a.shape # shape of the array
size=a.size   # number of elements
dimension=a.ndim  #number of dimensions
dtype=a.dtype   # Data type of the array
print("Shape of a :",shape)
print("Size of a:",size)
print("Number of dimension of a:",dimension)
print("Data type of a :",dtype)
```

```
Shape of a : (3,)
Size of a: 3
Number of dimension of a: 1
Data type of a : int32
```

10.Other Functions

In [86]:

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

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

In [87]:

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

```
Size of a in bytes: 12
```

In [90]:

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

```
Do a and copied_array shere memory False
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: