Numpy

1.Array Creation Functions

In [1]:

```
import numpy as np
```

In [2]:

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

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

In [3]:

```
#Create an array with evenly spaced values
b=np.arange(0,10,2) #values from 0 to 10 with step2
print("Array b:",b)
```

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

In [5]:

```
#Create an array filled with zeros
d=np.zeros((2,3))  # 2x3 array of zeros
print("Array d:\n",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) # 4x4 identity matrix
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 [9]:

```
#Reshape an array
a1=np.array([1,2,3])
reshaped=np.reshape(a1,(1,3)) #reshape to 1x3
print("Reshaped array:",reshaped)
```

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

In [10]:

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

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

In [11]:

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

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

In [14]:

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

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

3.Mathematical Functions

In [15]:

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

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

In [17]:

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

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

In [18]:

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

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

In [19]:

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

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

In [20]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 3
      1 #dot product of two arrays
      2 a2=np.array([1,2,3])
----> 3 dot_product=np.dot(a2,g) #dot product of a and g
      4 print("Dot product of a and g:",dot_product)

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

In [21]:

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

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

In [22]:

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

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

4.Statistical Functions

In [23]:

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

```
Mean of s: 2.5
```

In [24]:

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

```
Standard deviation of s: 1.118033988749895
```

In [25]:

```
#Minimum element of an array
minimum=np.min(s)
print('Min of s:',minimum)
```

```
Min of s: 1
```

In [26]:

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

```
Max of s: 4
```

5.Linear Algebra Functions

In [28]:

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

6.Random Sampling Functions

In [30]:

```
#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: [0.413163   0.34367191 0.64080944]
```

In [31]:

```
#Set seed for reproducibility
np.random.seed(0)
#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: [0.5488135  0.71518937 0.60276338]
```

In [32]:

```
#Generate random integers
rand_ints=np.random.randint(0,10,size=5) #random integers between 0 and 1
print("Random integers:",rand_ints)
```

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

In [33]:

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

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

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

7.Boolean & Logical Functions

In [34]:

```
#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 [35]:

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

```
All elements True: False
```

In [36]:

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

```
Any elements True: False
```

8.Set Operations

In [39]:

```
#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 [40]:

```
#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 [41]:

```
# Array attributes
a = np.array([1, 2, 3])
shape = a.shape  # Shape of the array
size = a.size    # Number of elements
dimensions = 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 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 [42]:

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

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

In [43]:

```
#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: 24
```

In [44]:

```
#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
```

In [ ]: