Array Creation Functions::

In [3]:

```
import numpy as np

np.__version__
```

Out[3]:

```
'1.26.4'
```

In [4]:

```
## create an array from a list

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

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

In [6]:

```
# create an array with evenly spaced values

b= np.arange(0,10,2)  #Values from 0 to 10 with step 2
print("Array b:",b)
```

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

In [7]:

```
# 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 [8]:

```
## create an array filled with ones

e = np.ones((3,2))  ## 3x2 array of ones
print("Array e:\n",e)
```

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

In [9]:

```
## creae 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.]]
```

Array Manipulation Functions::

In [12]:

```
# 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 [13]:

```
## 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 [14]:

```
# 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 [15]:

```
# stack array 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]]
```

Mathematical Functions

In [16]:

```
## 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]:

```
# Square each element

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

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

In [18]:

```
## Square root of each element

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 [20]:

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

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

In [21]:

```
## 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[21], line 4
      1 ## Dot product of two arrays
      3 a2 = np.array([1,2,3])
----> 4 dot_product = np.dot(a2,g)  # Dot product of a and g
      5 print("Dot product of a and g:", dot_product)

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

In [22]:

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

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

In [23]:

```
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
```

Statistical Functions::

In [24]:

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

```
Mean of s: 2.5
```

In [27]:

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

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

In [28]:

```
# Minimum element of an array

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

```
Min of s: 1
```

In [29]:

```
# Maximum element of an array

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

```
Max of s: 4
```

Linear Algebra Functions

In [36]:

```
# create a matrix

matrix = np.array([[1,2],[3,4]])
print(matrix)
```

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

Random Sampling Functions

In [35]:

```
## Generate random values btween 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: [0.52316162 0.60828863 0.54685059]
```

In [37]:

```
# set seed for reproducibility

np.random.seed(0)

#Generate random values b/w 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 [40]:

```
## Generate random integers

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

```
Random integers: [8 1 6 7 7]
```

In [41]:

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

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

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

Boolean & Logical Functions::

In [42]:

```
## check if all elements are True
# all

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

```
All elements True: False
```

In [43]:

```
## 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("All elements True:", all_true)
```

```
All elements True: False
```

In [44]:

```
## check if all elements are True
# any

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

```
Any elements True: True
```

Set Operations::

In [49]:

```
# Intersections 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 [50]:

```
## 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]
```

Array Attribute Functions::

In [52]:

```
## 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: int32
```

Other Functions

In [53]:

```
# 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 [55]:

```
## 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 [56]:

```
## Check if two arrays share memory

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

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

In [ ]: