Import Library

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

Creating Arrays

In [2]:
```
# 1D Array
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)

# 2D Array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
```

```
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
```

Using Built-in Functions

In [3]:
```
# Array of zeros
zeros = np.zeros((2, 3))
print(zeros)

# Array of ones
ones = np.ones((3, 3))
print(ones)

# Array with a range of numbers
range_arr = np.arange(0, 10, 2)
print(range_arr)

# Linearly spaced array
lin_space = np.linspace(0, 1, 5)
print(lin_space)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[0 2 4 6 8]
[0.   0.25 0.5  0.75 1.  ]
```

Basic Operations

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

# Addition
print(arr + 5)

# Multiplication
print(arr * 2)

# Exponentiation
print(arr ** 2)
```

```
[ 6  7  8  9 10]
[ 2  4  6  8 10]
[ 1  4  9 16 25]
```

Mathematical Functions

In [5]:
```
# Sine and Cosine

angles = np.array([0, np.pi/2, np.pi])
print(np.sin(angles))
print(np.cos(angles))

#  sin(π)≈0, but due to floating-point precision, it shows a very small value close to zero (1.224647e-16)
```

```
[0.0000000e+00 1.0000000e+00 1.2246468e-16]
[ 1.000000e+00  6.123234e-17 -1.000000e+00]
```

In [6]:
```
# Square root and Exponential

nums = np.array([1, 4, 9, 16])
print(np.sqrt(nums))
print(np.exp(nums))
```

```
[1. 2. 3. 4.]
[2.71828183e+00 5.45981500e+01 8.10308393e+03 8.88611052e+06]
```

Array Indexing and Slicing

In [7]:
```
#Indexing

arr = np.array([10, 20, 30, 40, 50])

# Accessing elements
print(arr[0])  # First element
print(arr[-1]) # Last element
```

```
10
50
```

In [8]:
```
# Slicing

print(arr[1:4])   # Elements from index 1 to 3
print(arr[:3])    # First three elements
print(arr[::2])   # Every second element
```

```
[20 30 40]
[10 20 30]
[10 30 50]
```

In [9]:
```
# 2D Array Indexing

arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Accessing elements
print(arr2[1, 2])  # Row 2, Column 3

# Slicing
print(arr2[0:2, 1:3])  # Sub-matrix
```

```
6
[[2 3]
 [5 6]]
```

Broadcasting

In [10]:
```
#Broadcasting

arr = np.array([1, 2, 3])
print(arr + 5)  # Adds 5 to each element
```

```
[6 7 8]
```

Vectorized Operations

In [11]:
```
# Vectorized Operations

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# Element-wise multiplication
print(a * b)
```

```
[ 4 10 18]
```

Conditional Statements

In [12]:
```
# Conditional Statements

arr = np.array([10, 20, 30, 40, 50])

# Boolean indexing
print(arr[arr > 30])  # Elements greater than 30
```

```
[40 50]
```

Array Manipulation

In [13]:
```
# Reshape

arr = np.arange(1, 7)
reshaped_arr = arr.reshape((2, 3))
print(arr)
print(reshaped_arr)
```

```
[1 2 3 4 5 6]
[[1 2 3]
 [4 5 6]]
```

In [14]:
```
# Flattening

arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
flattened = arr2.flatten()
print(flattened)
```

```
[[1 2 3]
 [4 5 6]]
[1 2 3 4 5 6]
```

In [15]:
```
# Transpose

arr3 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr3)
transposed = arr3.T
print(transposed)
```

```
[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

Aggregations and Statistics

In [16]:
```
# Basic Statistical Functions

arr = np.array([1, 2, 3, 4, 5])

# Sum and Mean
print(np.sum(arr))
print(np.mean(arr))

# Standard Deviation and Variance
print(np.std(arr))
print(np.var(arr))
```

```
15
3.0
1.4142135623730951
2.0
```

In [17]:
```
# Min and Max

print(np.min(arr))
print(np.max(arr))
```

```
1
5
```

In [18]:
```
# Axis-wise Operations

arr2 = np.array([[1, 2, 3], [4, 5, 6]])

# Sum across columns
print(np.sum(arr2, axis=0))

# Sum across rows
print(np.sum(arr2, axis=1))
```

```
[5 7 9]
[ 6 15]
```

Random Numbers and Distributions

In [19]:
```
# Random integers
rand_ints = np.random.randint(0, 10, size=(2, 3))
print(rand_ints)

# Random floats between 0 and 1
rand_floats = np.random.rand(2, 3)
print(rand_floats)

# Normal Distribution
norm_dist = np.random.randn(2, 3)
print(norm_dist)
```

```
[[2 4 3]
 [0 3 8]]
[[0.98279547 0.25050344 0.2776913 ]
 [0.61946616 0.88325081 0.10157034]]
[[-0.28791975 -0.66302971 -0.62931675]
 [-1.78201822  1.083215   -0.37892021]]
```

Linear Algebra with NumPy

In [20]:
```
from numpy import linalg

# Matrices
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Matrix multiplication
print(np.dot(A, B))

# Determinant
print(linalg.det(A))

# Inverse
print(linalg.inv(A))

# Eigenvalues and Eigenvectors
eigenvals, eigenvecs = linalg.eig(A)
print("Eigenvalues:", eigenvals)
print("Eigenvectors:", eigenvecs)
```

```
[[19 22]
 [43 50]]
-2.0000000000000004
[[-2.   1. ]
 [ 1.5 -0.5]]
Eigenvalues: [-0.37228132  5.37228132]
Eigenvectors: [[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
```

In [ ]: