In [2]:

```
#PASSING LIST

number_list = [1, 2, 3, 4, 5, 6, 7, 8]
print("Python List:", number_list)
print("Data type of the list:", type(number_list))
```

```
Python List: [1, 2, 3, 4, 5, 6, 7, 8]
Data type of the list: <class 'list'>
```

In [3]:

```
import numpy as np

number_list = [1, 2, 3, 4, 5, 6, 7, 8]

array = np.array(number_list)

print("NumPy Array:", array)

print("Type of the array:", type(array))
```

```
NumPy Array: [1 2 3 4 5 6 7 8]
Type of the array: <class 'numpy.ndarray'>
```

In [4]:

```
import numpy as np
number_list = [1, 2, 3, 4, 5, 6, 7, 8]
array = np.array(number_list)
print("NumPy Array:", array)
print("Type of array:", type(array))
print("Data type of elements:", array.dtype)
```

```
NumPy Array: [1 2 3 4 5 6 7 8]
Type of array: <class 'numpy.ndarray'>
Data type of elements: int64
```

In [9]:

```
import numpy as np
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print("Matrix:\n", matrix)
print("Shape:", matrix.shape)
print("Size:", matrix.size)
print("Data type:", matrix.dtype)
```

```
Matrix:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)
Size: 9
Data type: int64
```

In [11]:

```
import numpy as np

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

print("NumPy Array:\n", array)
print("Shape:", array.shape)
print("Dimensions:", array.ndim)
print("Data Type:", array.dtype)
print("Size:", array.size)
```

```
NumPy Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)
Dimensions: 2
Data Type: int64
Size: 9
```

In [12]:

```
import numpy as np

matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]

array = np.array(matrix)

print("NumPy Array:\n", array)
print("Shape:", array.shape)
print("Data Type:", array.dtype)
```

```
NumPy Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)
Data Type: int64
```

In [14]:

```
#OPERATORS
#ARANGE:

import numpy as np
array = np.arange(1, 11)
print("NumPy Array:", array)
```

```
NumPy Array: [ 1  2  3  4  5  6  7  8  9 10]
```

In [15]:

```
import numpy as np
array = np.arange(1, 11, 2)
print("NumPy Array:", array)
```

```
NumPy Array: [1 3 5 7 9]
```

In [16]:

```
#ZEROS AND ONES:

import numpy as np
array = np.zeros(3)
print("NumPy Array:", array)
```

```
NumPy Array: [0. 0. 0.]
```

In [17]:

```
import numpy as np
matrix = np.ones((10, 10))
print("10x10 Matrix of Ones:\n", matrix)
```

```
10x10 Matrix of Ones:
 [[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

In [18]:

```
#LINSPACE:

import numpy as np
array = np.linspace(1, 11, 25)
print("25 Evenly Spaced Numbers:\n", array)
```

```
25 Evenly Spaced Numbers:
 [ 1.          1.41666667  1.83333333  2.25        2.66666667  3.08333333
  3.5         3.91666667  4.33333333  4.75        5.16666667  5.58333333
  6.          6.41666667  6.83333333  7.25        7.66666667  8.08333333
  8.5         8.91666667  9.33333333  9.75       10.16666667 10.58333333
 11.        ]
```

In [19]:

```
#IDENTITY MATRIX:
import numpy as np
identity_matrix = np.eye(10)
print("10x10 Identity Matrix:\n", identity_matrix)
```

```
10x10 Identity Matrix:
 [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

In [20]:

```
#RANDOM PACKAGE:
import numpy as np
random_array = np.random.rand(4)
print("Random Array:", random_array)
```

```
Random Array: [0.17360459 0.51125288 0.20276922 0.64791746]
```

In [21]:

```
#RANDN:

import numpy as np
random_array = np.random.randn(2)
print("Random Array (Standard Normal):", random_array)
```

```
Random Array (Standard Normal): [0.65079643 0.42757201]
```

In [22]:

```
import numpy as np
matrix = np.random.randn(6, 6)
print("6x6 Matrix (Standard Normal):\n", matrix)
```

```
6x6 Matrix (Standard Normal):
 [[-0.5502278   0.19773991  1.94752231  0.38662984 -1.24999583  1.14619689]
 [-1.0467384  -1.17072637 -0.68863579  2.25946452  0.61874751 -0.95118145]
 [ 0.86767791  0.89813464  0.76863257  0.49723199  0.3394171   0.23425164]
 [-1.15631425 -1.83536843 -0.27288604 -0.0465998   0.5604363   0.14863507]
 [ 0.18791573 -1.12250404 -0.85192116 -0.36396696  0.67998849 -0.05383013]
 [-1.8242246  -2.47276279 -0.07227514 -0.19107476 -0.5681193   0.68887426]]
```

In [23]:

```
#RANDINT:
import numpy as np

random_int = np.random.randint(1, 6)  # upper bound is exclusive, so use 6
print(random_int)
```

```
2
```

In [24]:

```
import numpy as np

random_array = np.random.randint(1, 6, size=10)
print(random_array)
```

```
[2 5 5 4 2 4 5 4 3 3]
```

In [25]:

```
#ARRAY
import numpy as np

sequential_array = np.arange(25)
print(sequential_array)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
```

In [26]:

```
#RANDOM
import numpy as np

random_array = np.random.randint(0, 50, size=10)
print(random_array)
```

```
[26  0  3 37  4 10  5 17  1 13]
```

In [27]:

```
#SHAPE
import numpy as np

a = np.array([1, 2, 3, 4, 5, 6])
print("Array:", a)
print("Shape:", a.shape)
```

```
Array: [1 2 3 4 5 6]
Shape: (6,)
```

In [28]:

```
#RESHAPE
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6])
reshaped_a = a.reshape(2, 3)
print("Reshaped 2x3 matrix:\n", reshaped_a)
```

```
Reshaped 2x3 matrix:
 [[1 2 3]
 [4 5 6]]
```

In [29]:

```
#MINIMUM
import numpy as np
```

```
a = np.array([12, 7, 25, 3, 18])
min_value = np.min(a)
print("Array:", a)
print("Minimum value:", min_value)
```

```
Array: [12  7 25  3 18]
Minimum value: 3
```

In [30]:

```
#ARGUMENT FUNCTION
import numpy as np
r = np.array([4, 15, 9, 27, 6, 13])
max_index = np.argmax(r)
print("Array:", r)
print("Index of maximum value:", max_index)
```

```
Array: [ 4 15  9 27  6 13]
Index of maximum value: 3
```

In [31]:

```
#SLICING
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
a[:5] = 1000
print("Modified array:", a)
```

```
Modified array: [1000 1000 1000 1000 1000    6    7    8    9   10]
```

In [32]:

```
#INDEXING
import numpy as np
mat = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

print("mat =\n", mat)
```

```
mat =
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [33]:

```
#ARITHMETIC OPERATORS
import numpy as np
c = np.arange(1, 11)
print("Array c:", c)

add_result = c + c
print("Addition result:", add_result)

sub_result = c - c
print("Subtraction result:", sub_result)

mul_result = c * c
print("Multiplication result:", mul_result)

div_result = c / c
print("Division result:", div_result)
```

```
Array c: [ 1  2  3  4  5  6  7  8  9 10]
Addition result: [ 2  4  6  8 10 12 14 16 18 20]
Subtraction result: [0 0 0 0 0 0 0 0 0 0]
Multiplication result: [  1   4   9  16  25  36  49  64  81 100]
Division result: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [34]:

```
#UNIVERSAL ARRAY
import numpy as np
a = np.array([1, 2, 3, 4, 5])
b = np.array([10, 20, 30, 40, 50])
result = np.add(a, b)
print("Array a:", a)
print("Array b:", b)
print("Element-wise addition result:", result)
```

```
Array a: [1 2 3 4 5]
Array b: [10 20 30 40 50]
Element-wise addition result: [11 22 33 44 55]
```

In [35]:

```
#APPLYING TRIGONOMETRIC FUNCTION

import numpy as np
angles = np.array([0, np.pi/6, np.pi/4, np.pi/3, np.pi/2])
sine_values = np.sin(angles)
cosine_values = np.cos(angles)
print("Angles (radians):", angles)
print("Sine values:", sine_values)
print("Cosine values:", cosine_values)
```

```
Angles (radians): [0.         0.52359878 0.78539816 1.04719755 1.57079633]
Sine values: [0.         0.5        0.70710678 0.8660254  1.        ]
Cosine values: [1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01
 6.12323400e-17]
```

In [36]:

```
#EXPONENTIATION OF ARRAY ELEMENTS
import numpy as np

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

exponent = 3

powered_array = np.power(arr, exponent)

# Print the results
print("Original array:", arr)
print(f"Array elements raised to the power of {exponent}:", powered_array)
```

```
Original array: [2 3 4 5]
Array elements raised to the power of 3: [  8  27  64 125]
```

In [37]:

```
#CALCULATING THE SQUARE ROOT OF ELEMENTS
import numpy as np
arr = np.array([0, 1, 4, 9, 16, 25])
sqrt_arr = np.sqrt(arr)
print("Original array:", arr)
print("Square root of each element:", sqrt_arr)
```

```
Original array: [ 0  1  4  9 16 25]
Square root of each element: [0. 1. 2. 3. 4. 5.]
```

In [38]:

```
#MATRIX MULTIPLICATION
import numpy as np
mat = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
result = mat * mat
```

```
print("Original matrix:\n", mat)
print("Element-wise multiplication result:\n", result)
```

```
Original matrix:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Element-wise multiplication result:
 [[ 1  4  9]
 [16 25 36]
 [49 64 81]]
```

In [ ]: