In [2]:
```
#importing numpy
import numpy as np
```

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

In [4]:
```
#zeros and ones
zeros_arr=np.zeros(5)
zeros_arr
```

Out[4]:
```
array([0., 0., 0., 0., 0.])
```

In [5]:
```
ones_arr=np.ones(5)
ones_arr
```

Out[5]:
```
array([1., 1., 1., 1., 1.])
```

In [6]:
```
#range of values  (start,stop,step)
range_arr=np.arange(3,10,5)
range_arr
```

Out[6]:
```
array([3, 8])
```

In [7]:
```
#random values
rand_arr=np.random.rand(3,3)  #3*3 random array  (here it  is a matrix)
rand_arr
```

Out[7]:
```
array([[0.36633573, 0.27129282, 0.46425682],
       [0.57419157, 0.01823356, 0.4485785 ],
       [0.54063573, 0.95180399, 0.77467397]])
```

In [52]:
```
# basic arithmetic op
arr1=np.array([1,2,3,4,5])
arr2=np.array([5,3,6,7,8])
```

In [9]:
```
result=arr1+arr2
result
```

Out[9]:
```
array([ 6,  5,  9, 11, 13])
```

In [10]:
```
result=arr1-arr2
result
```

Out[10]:
```
array([-4, -1, -3, -3, -3])
```

In [11]:
```
result=arr1*arr2
result
```

Out[11]:
```
array([ 5,  6, 18, 28, 40])
```

In [12]:
```
result=arr1/arr2
result
```

Out[12]:
```
array([0.2       , 0.66666667, 0.5       , 0.57142857, 0.625     ])
```

In [13]:
```
#element wise operatioon
result=np.square(arr1)
print(result)
result=np.sqrt(arr2)
print(result)
result=np.exp(arr2)
print(result)
```

```
[ 1  4  9 16 25]
[2.23606798 1.73205081 2.44948974 2.64575131 2.82842712]
[ 148.4131591    20.08553692  403.42879349 1096.63315843 2980.95798704]
```

In [14]:
```
# dot operator
dot_product=np.dot(arr1,arr2)
print(dot_product)
```

```
97
```

In [15]:
```
#broadcasting  adds a elemenrt each to array to array
result=arr1+5
result
```

Out[15]:
```
array([ 6,  7,  8,  9, 10])
```

In [16]:
```
#array manipulation
a=np.random.randint(4,30,(3,4))
print(a)
```

```
[[11 14 29 10]
 [26 13  8 26]
 [22 19 18 18]]
```

In [17]:
```
reshaped_arr=a.reshape(2,6)
```

In [18]:
```
print(reshaped_arr)
```

```
[[11 14 29 10 26 13]
 [ 8 26 22 19 18 18]]
```

In [19]:
```
#transpose
transposed_arr=a.T
print(transposed_arr)
```

```
[[11 26 22]
 [14 13 19]
 [29  8 18]
 [10 26 18]]
```

In [20]:
```
#flatten
flattend_arr=a.flatten()
print(flattend_arr)
```

```
[11 14 29 10 26 13  8 26 22 19 18 18]
```

In [21]:
```
#mean, median , standard deviation
mean_val=np.mean(a)
print(mean_val)
```

```
17.833333333333332
```

In [22]:
```
median_val=np.median(a)
```

```
print(median_val)
```

```
18.0
```

In [23]:
```
std_dev=np.std(a)
print(std_dev)
```

```
6.58069567413321
```

In [24]:
```
#sum min max
total_sum=np.sum(a)
print(total_sum)
```

```
214
```

In [25]:
```
min_val=np.min(a)
print(min_val)
```

```
8
```

In [26]:
```
max_val=np.max(a)
print(max_val)
```

```
29
```

In [27]:
```
#indexing and slicing
a[0]
```

Out[27]:
```
array([11, 14, 29, 10], dtype=int32)
```

In [28]:
```
print(a[0][1])
```

```
14
```

In [29]:
```
print(a[2][3])
```

```
18
```

In [30]:
```
#slicing
a[2:4]
```

Out[30]:
```
array([[22, 19, 18, 18]], dtype=int32)
```

In [31]:
```
a[1:3]
```

Out[31]:
```
array([[26, 13,  8, 26],
       [22, 19, 18, 18]], dtype=int32)
```

In [32]:
```
#logical operators  #checks each elemnets is greater than or less than the condi
bool_arr=a>3
print(bool_arr)
```

```
[[ True  True  True  True]
 [ True  True  True  True]
 [ True  True  True  True]]
```

In [33]:
```
bool_arr=a>4
print(bool_arr)
```

```
[[ True  True  True  True]
 [ True  True  True  True]
 [ True  True  True  True]]
```

In [34]:
```
bool_arr=a>5
print(bool_arr)
```

```
[[ True  True  True  True]
 [ True  True  True  True]
 [ True  True  True  True]]
```

In [37]:
```
#concatanation
combined_arr=np.concatenate((arr1,arr2),axis=0)
combined_arr
```

Out[37]:
```
array([1, 2, 3, 4, 5, 5, 3, 6, 7, 8])
```

In [40]:
```
#stacking
stacked_arr=np.vstack((arr1,arr2)) #vertically stack array
stacked_arr
```

Out[40]:
```
array([[1, 2, 3, 4, 5],
       [5, 3, 6, 7, 8]])
```

In [45]:
```
#Linear algebra
# create a matrix
matrix=np.array ([[1,3],[2,4]])
# Determinant of a matrix
determinant=np.linalg.det(matrix)
print("Determinant of matrix:",determinant)
# Inverse=np.linalg.inv(matrix)
inverse=np.linalg.inv(matrix)
print("Inverse of matrix:\n",inverse)
```

```
Determinant of matrix: -2.0
Inverse of matrix:
 [[-2.   1.5]
 [ 1.  -0.5]]
```

In [46]:
```
#Random sampling
# Generate random values between 0 and 1
random_vals = np.random.rand(7) # Array of 3 random values between 0 and 1
print("Random values:", random_vals)

# Set seed for reproducibility
np.random.seed(0)
# Generate random values between 0 and 1
random_vals = np.random.rand(7) # Array of 3 random values between 0 and 1
print("Random values:", random_vals)
# Generate random integers
rand_ints = np.random.randint(0, 10, size=5) # Random integers between 0 and
print("Random integers:", rand_ints)

# 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
print("Random integers:", rand_ints)
```

```
Random values: [0.25320467 0.87600985 0.6511826  0.29231109 0.35401085 0.46852929
 0.09590761]
Random values: [0.5488135  0.71518937 0.60276338 0.54488318 0.4236548  0.64589411
 0.43758721]
Random integers: [6 8 8 1 6]
Random integers: [5 0 3 3 7]
```

In [47]:
```python
#Set Operations
# Intersection of two arrays
set_a = np.array([7, 2, 3, 4])
set_b = np.array([3, 4, 3, 6])
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)

# Union of two arrays
union = np.union1d(set_a, set_b)
print("Union of a and b:", union)
```

```
Intersection of a and b: [3 4]
Union of a and b: [2 3 4 6 7]
```

In [48]:
```python
#Array Attribute Functions
# Array attributes
a = np.array([3, 4, 5])
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
```

In [49]:
```python
#Other Functions
# Create a copy of an array
a = np.array([4, 2, 3])
copied_array = np.copy(a) # Create a copy of array a
print("Copied array:", copied_array)

# 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)

# 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)
```

```
Copied array: [4 2 3]
Size of a in bytes: 24
Do a and copied_array share memory? False
```

In [51]:
```python
# Create an identity matrix
f = np.eye(3) # 3*3 identity matrix
print("Identity matrix f:\n", f)
```

```
Identity matrix f:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [53]:
```python
#Boolean & Logical Functions
# 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)

# 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)

# 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)

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

```
All elements True: False
All elements True: False
All elements True: False
Any elements True: False
```

In [ ]: