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

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

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

In [3]:
```
b = np.arange(0, 10, 2)
print("Array b:", b)
```

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

In [4]:
```
d = np.zeros((2,3))
print("Array d:\n",d)
```

```
Array d:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

In [5]:
```
e = np.ones((3,2))
print("array e:\n",e)
```

```
array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
```

In [6]:
```
f = np.eye(4)
print("Identity matrix f:\n", f)#identity matrix
```

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

In [ ]:
```
# array manipulation functions
```

In [7]:
```
a1 = np.array([1, 2, 3])
reshaped = np.reshape(a1, (1,3))
print(reshaped)
```

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

In [8]:
```
f1 = np.array([[1, 2], [3, 4]])#flatten to 1d array
flattend = np.ravel(f1)
print(flattend)
```

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

In [9]:
```
e1 = np.array([[1, 2], [3, 4]])
transpose = np.transpose(e1)#convertion to array
print(transpose)
```

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

In [10]:
```
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 [11]:
```
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 [13]:
```
square = np.power(g, 2)
print(square)
```

```
[ 1  4  9 16]
```

In [14]:
```
sqrt = np.sqrt(g)
print(sqrt)
```

```
[1.         1.41421356 1.73205081 2.        ]
```

In [18]:
```
a2 = np.array([1, 2, 3])
dot_prod = np.dot(a2,g)
print(dot_prod)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[18], line 2
      1 a2 = np.array([1, 2, 3])
----> 2 dot_prod = np.dot(a2,g)
      3 print(dot_prod)

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

In [19]:
```
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 [20]:
```
s = np.array([1, 2, 3, 4])
mean = np.mean(s)
print("Mean of s:", mean)
```

```
Mean of s: 2.5
```

In [21]:
```
std_dev = np.std(s)
print("Standard deviation of s:", std_dev)
```

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

In [22]:
```
minimum = np.min(s)
print("Min of s:", minimum)
```

```
Min of s: 1
```

In [23]:
```
maximum = np.max(s)
print("Max of s:", maximum)
```

```
Max of s: 4
```

linear algebra functions

In [24]:
```
matrix = np.array([[1, 2], [3, 4]])
```

sampling functions

In [26]:
```
random_val = np.random.rand(3)
print(random_val)
```

```
[0.1458631  0.16126346 0.55140402]
```

In [27]:
```
rand_int = np.random.randint(0, 10, (1,2))
print(rand_int)
```

```
[[4 6]]
```

In [28]:
```
rand_int = np.random.randint(0, 10, size=5)
print(rand_int)
```

```
[0 1 9 5 0]
```

In [33]:
```
np.random.seed(0)
rand_int=np.random.randint(0, 10, size=5)
print(rand_int)
```

```
[5 0 3 3 7]
```

boolean and logical functions

In [34]:
```
logic = np.array([True, False, True])
all_true = np.all(logic)
print(all_true)
```

```
False
```

In [35]:
```
logic = np.array([False, False, True])
all_true = np.all(logic)
print(all_true)
```

```
False
```

In [36]:
```
logic = np.array([False, False, False])
all_true = np.all(logic)
print(all_true)
```

```
False
```

In [37]:
```
logic = np.array([False, False, False])
all_true = np.any(logic)
print(all_true)
```

```
False
```

In [38]:
```
logic = np.array([True, False, False])
all_true = np.any(logic)
print(all_true)
```

```
True
```

set operations

In [39]:
```
a = np.array([1, 2, 3, 4])
b = np.array([3,4,5,6])
intersection = np.intersect1d(a,b)
print(intersection)
```

```
[3 4]
```

In [41]:
```
union = np.union1d(a,b)
print(union)
```

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

array attributes

In [43]:
```
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)
print(size)
print(dimensions)
print(dtype)
```

```
(3,)
3
1
int32
```

other functions

In [44]:
```
a
```

Out[44]:
```
array([1, 2, 3])
```

In [45]:
```
copied_arr = np.copy(a)
print(copied_arr)
```

```
[1 2 3]
```

In [47]:
```
arr_size = a.nbytes
arr_size
```

Out[47]:
```
12
```

In [50]:
```
shared = np.shares_memory(a, copied_arr)
print("Do a and copied_arr share memory?", shared)
```

```
Do a and copied_arr share memory? False
```

In [ ]:
```

```