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 [6]:
```
e = np.ones((3,2))
print("Array e:\n",e)
```

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

In [7]:
```
e = np.ones((3,2),dtype =int)
print("Array e:\n",e)
```

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

In [8]:
```
f = np.eye((4))
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.]]
```

In [9]:
```
a1= np.array([1,2,3])
reshaped = np.reshape(a1,(1,3))
print(reshaped)
```

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

In [11]:
```
f1= np.array([[1,2],[3,4]])
flattend = np.ravel(f1)
print(flattend)
```

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

In [12]:
```
e1= np.array([[1,2],[3,4]])
transpose = np.transpose(e1)
print(transpose)
```

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

In [13]:
```
a2 = np.array([1,2])
b2 = np.array([3,4])
stacked = np.vstack([a2,b2])
print(stacked)
```

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

In [14]:
```
a2 = np.array([1,2])
b2 = np.array([3,4])
stacked = np.hstack([a2,b2])
print(stacked)
```

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

In [ ]:
```

```

In [15]:
```
g = np.array([1,2,3,4])
added = np.add(g,2)
print(added)
```

```
[3 4 5 6]
```

In [18]:
```
squared = np.power(g,2)
print(squared)
```

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

In [19]:
```
sqrt_val = np.sqrt(g)
print(sqrt_val)
```

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

In [21]:
```
print(a)
print(a1)
```

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

In [ ]:
```

```

In [22]:
```
a3 = np.array([1,2,3])
dot_product =np.dot(a1,a)
print(dot_product)
```

```
14
```

In [24]:
```
a3 = np.array([1,2,3])
dot_product =np.dot(a1,g)
print(dot_product)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[24], line 2
      1 a3 = np.array([1,2,3])
----> 2 dot_product =np.dot(a1,g)
      3 print(dot_product)

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

In [ ]:

In [ ]:

In [25]:
```
s = np.array([1,2,3,4])
mean =np.mean(s)
print(mean)
```

```
2.5
```

In [26]:
```
s = np.array([1,2,3,4])
mean =np.var(s)
print(mean)
```

```
1.25
```

In [28]:
```
std_dev =np.std(s)
print(std_dev)
```

```
1.118033988749895
```

In [ ]:

In [30]:
```
minimum =np.min(s)
print(minimum)
```

```
1
```

In [ ]:

In [ ]:

In [31]:
```
maximum =np.max(s)
print(maximum)
```

```
4
```

In [ ]:

In [ ]:

In [32]:
```
random_vals = np.random.rand(3)
print(random_vals)
```

```
[0.8684898  0.33173624 0.6544892 ]
```

In [34]:
```
np.random.seed(0)
random_vals = np.random.rand(3)
print(random_vals)
```

```
[0.5488135  0.71518937 0.60276338]
```

In [35]:
```
np.random.seed(0)
random_vals = np.random.randint(0,10,size=5)
print(random_vals)
```

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

In [ ]:

In [ ]:

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

```
False
```

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

```
True
```

In [ ]:

In [ ]:

In [ ]:

In [39]:
```
set_a = np.array([1,2,3,4])
set_b = np.array([3,4,5,6])

inersection = np.intersect1d(set_a,set_b)
print(inersection)
```

```
[3 4]
```

In [40]:
```
union = np.union1d(set_a,set_b)
print(union)
```

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

In [ ]:

In [41]:
```
 # 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: int64
```

In [42]:
```
# 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 [43]:
```
# 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: 24
```

In [44]:
```
# 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 [ ]: