1.Array creation function

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

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

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

In [6]:
```
b=np.arange(0,10,2)
print("Array b:",b)
```

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

In [8]:
```
c=np.zeros((2,3))
print("Array c:\n",c)
```

```
Array c:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

In [10]:
```
d=np.zeros((2,3),dtype=int)
print("Array d:\n",d)
```

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

In [12]:
```
e=np.ones((3,2))
print("Array e:\n",e)
```

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

In [13]:
```
e=np.ones((3,2),dtype=int)
print("Array e:\n",e)
```

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

In [15]:
```
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 [16]:
```
f=np.eye(4,dtype=int)
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]]
```

2.Array Manipulation

In [18]:
```
a1=np.array([1,2,3])
reshaped=np.reshape(a1,(1,3))
print("reshaped array:",reshaped)
```

```
reshaped array: [[1 2 3]]
```

In [20]:
```
a1=np.array([1,2,3])
reshaped=np.reshape(a1,(3,1))
print("reshaped array:\n",reshaped)
```

```
reshaped array:
 [[1]
 [2]
 [3]]
```

In [23]:
```
f1=np.array([[1,2],[3,4]])
flattened=np.ravel(f1)
print("flattened array:",flattened)
```

```
flattened array: [1 2 3 4]
```

In [24]:
```
f1=np.array([[1,2],[3,4],[7,8],[9,10],[9,5]])
flattened=np.ravel(f1)
print("flattened array:",flattened)
```

```
flattened array: [ 1  2  3  4  7  8  9 10  9  5]
```

In [26]:
```
e1=np.array([[1,2],[3,4],[5,6]])
transpose=np.transpose(e1)
print("transpose array:\n",transpose)
```

```
transpose array:
 [[1 3 5]
 [2 4 6]]
```

In [27]:
```
a2=np.array([1,2])
b2=np.array([3,4])
stacked=np.vstack([a2,b2])
print("stacked array:\n",stacked)
```

```
stacked array:
 [[1 2]
 [3 4]]
```

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

In [3]:
```
a2=np.array([1,2])
b2=np.array([3,4])
c2=np.array([5,6])
stacked=np.vstack([a2,b2,c2])
print("stacked array:\n",stacked)
```

```
stacked array:
 [[1 2]
 [3 4]
 [5 6]]
```

3.Mathematical functions

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

```
Added 2 to g: [3 4 5 6]
```

In [5]:
```
squared=np.power(g,2)
print("squared g:",squared)
```

```
squared g: [ 1  4  9 16]
```

In [7]:
```
sqrt_val=np.sqrt(g)
print("square root of g:",sqrt_val)
```

```
square root of g: [1.         1.41421356 1.73205081 2.        ]
```

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

In [12]:
```
a=np.array([1,2,3])
```

In [10]:
```
print(a1)
print(g)
```

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

In [13]:
```
print(a)
print(a1)
```

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

In [15]:
```
a3=np.array([1,2,3])
dot_product=np.dot(a1,a)
print("Dot product of a1 and a:",dot_product)
```

```
Dot product of a1 and a: 14
```

4.Statistical Functions

In [16]:
```
s=np.array([1,2,3,4])
mean=np.mean(s)
print("Mean of s:",mean)
```

```
Mean of s: 2.5
```

In [17]:
```
std_dev=np.std(s)
print("Standard deviation of s:",std_dev)
```

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

In [18]:
```
minimum=np.min(s)
print("Min of s:",minimum)
```

```
Min of s: 1
```

In [19]:
```
maximum=np.max(s)
print("Max of s:",maximum)
```

```
Max of s: 4
```

5.Linear Algebra Functions

In [20]:
```
matrix=np.array([[1,2],[3,4]])
```

In [21]:
```
matrix
```

Out[21]:
```
array([[1, 2],
       [3, 4]])
```

6.Random Sampling Functions

In [22]:
```
random_vals=np.random.rand(3)
print("Random values:",random_vals)
```

```
Random values: [0.84907769 0.58906818 0.81818808]
```

In [23]:
```
np.random.seed(0)
random_vals=np.random.rand(3)
print("Random values:",random_vals)
```

```
Random values: [0.5488135  0.71518937 0.60276338]
```

In [24]:
```
random_ints=np.random.randint(0,10,size=5)
print("Random integers:",random_ints)
```

```
Random integers: [3 7 9 3 5]
```

In [25]:
```
np.random.seed(0)
random_ints=np.random.randint(0,10,size=5)
print("Random integers:",random_ints)
```

```
Random integers: [5 0 3 3 7]
```

7.Boolean and Logical Functions

In [26]:
```
logical_test=np.array([True,False,True])
all_true=np.all(logical_test)
print("All elements true:",all_true)
```

```
All elements true: False
```

In [27]:
```
logical_test=np.array([False,False,False])
all_true=np.all(logical_test)
print("All elements true:",all_true)
```

```
All elements true: False
```

In [28]:
```
logical_test=np.array([False,False,True])
all_true=np.all(logical_test)
print("All elements true:",all_true)
```

```
All elements true: False
```

In [29]:
```
logical_test=np.array([True,True,True])
all_true=np.all(logical_test)
print("All elements true:",all_true)
```

```
All elements true: True
```

In [30]:
```
any_true=np.any(logical_test)
print("Any elements True:",any_true)
```

```
Any elements True: True
```

8.Set Operations

In [32]:
```
set_a=np.array([1,2,3,4])
set_b=np.array([3,4,5,6])
intersection=np.intersect1d(set_a,set_b)
print("intersection of a and b:",intersection)
```

```
intersection of a and b: [3 4]
```

In [33]:
```
union=np.union1d(set_a,set_b)
print("union of a and b:",union)
```

```
union of a and b: [1 2 3 4 5 6]
```

9.Array attribute Functions

In [36]:
```
a=np.array([1,2,3])
shape=a.shape
size=a.size
dimensions=a.ndim
dtype=a.dtype
print("Shape of a:",shape)
print("size of a:",size)
print("Number of dimensions of :",dimensions)
print("Data type of a:",dtype)
```

```
Shape of a: (3,)
size of a: 3
Number of dimensions of : 1
Data type of a: int64
```

10.Other Functions

In [37]:
```
a=np.array([1,2,3])
copied_array=np.copy(a)
print("Copied array:",copied_array)
```

```
Copied array: [1 2 3]
```

In [38]:
```
array_size_in_bytes=a.nbytes
print("Size of a in bytes:",array_size_in_bytes)
```

```
Size of a in bytes: 24
```

In [39]:
```
shared=np.shares_memory(a,copied_array)
print("Do a and copied_array share memory?",shared)
```

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

In [ ]:
```
```