Array with zeros and ones

In [4]:
```
import numpy as np
```

In [5]:
```
a=np.zeros(3,dtype=np.int64)
b=np.ones(3,dtype=np.int64)
print(a)
print(b)
```

```
[0 0 0]
[1 1 1]
```

Create an array

In [6]:
```
c=np.array([5,55,555,5555])
c
```

Out[6]:
```
array([   5,   55,  555, 5555])
```

Create an array whose initial content is random

In [7]:
```
print(np.empty(5))
```

```
[2.12199579e-314 3.96813213e-312 4.58492919e-321 3.31031344e-312
 9.44122346e-314]
```

Array with the range of values with even interval

In [8]:
```
print(np.arange(1,50,2))
```

```
[ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47
 49]
```

Array with values that are spaced linearly in a specified interval

In [9]:
```
print(np.linspace(1,50,num=5))
```

```
[ 1.   13.25 25.5  37.75 50.  ]
```

Access and manipulate elements in the array

In [10]:
```
print(c[1:3])
```

```
[ 55 555]
```

In [11]:
```
c[1]=10
print(c)
```

```
[   5   10  555 5555]
```

Create a 2-dimensional array and check the shape of the array

In [12]:
```
d=np.array([[1,2,3],[4,5,6]])
print(d)
d.shape
```

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

Out[12]:
```
(2, 3)
```

Using the arange() and linspace() function to evenly space values in a specified interval

In [13]:
```
e=np.arange(1,11)
f=np.linspace(1,2,num=8)
print(e)
print(f)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[1.         1.14285714 1.28571429 1.42857143 1.57142857 1.71428571
 1.85714286 2.        ]
```

Create an array of random values between 0 and 1

In [14]:
```
array_random2 = np.random.random((2, 2))
print(array_random2)
```

```
[[0.92636296 0.02594583]
 [0.1712649  0.49669537]]
```

Repeat elements in an array using repeat() and tile()

In [15]:
```
array_repeat = np.array([1, 2, 3])
array_repeated = np.repeat(array_repeat, 3)
array_tiled = np.tile(array_repeat, 3)
print(array_repeated)
print(array_tiled)
```

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

Shape and size of an array

In [19]:
```
print(d.shape)
print(np.size(d))
```

```
(2, 3)
6
```

Number of dimensions in an array

In [20]:
```
print(np.ndim(d))
```

```
2
```

Reshape an array

In [21]:
```
array_reshaped = np.arange(1, 10).reshape(3, 3)
print(array_reshaped)
```

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

Create a null array of size 10

In [22]:
```
null_array = np.zeros(10)
print(null_array)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Array with numbers divisible by 7

In [24]:
```
array_divisible_by_7 = np.arange(10, 50)[np.arange(10, 50) % 7 == 0]
print(array_divisible_by_7)
```

```
[14 21 28 35 42 49]
```

Perform operations using arrays

In [25]:
```
array_op1 = np.array([1, 2, 3])
array_op2 = np.array([3, 2, 1])
print(array_op1 + array_op2)
print(array_op1 - array_op2)
print(array_op1 * array_op2)
print(array_op1 / array_op2)
```

```
[4 4 4]
[-2  0  2]
[3 4 3]
[0.33333333 1.         3.        ]
```

Relational operations using arrays

In [26]:
```
array_rel1 = np.array([1, 2, 3])
array_rel2 = np.array([2, 2, 2])
print(array_rel1 == array_rel2)
print(array_rel1 != array_rel2)
print(array_rel1 > array_rel2)
print(array_rel1 < array_rel2)
```

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

Use Arithmetic operators and print the output using arrays

In [27]:
```
array_arithmetic = np.array([1, 2, 3])
array_arithmetic += 2
print(array_arithmetic)
```

```
[3 4 5]
```

Use Relational operators and print the results using arrays

In [28]:
```
array_rel1 = np.array([1, 2, 3])
array_rel2 = np.array([2, 2, 2])
print(array_rel1 == array_rel2)
print(array_rel1 != array_rel2)
print(array_rel1 > array_rel2)
print(array_rel1 < array_rel2)
```

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