Array with zeros and ones

In [6]:

```
import numpy as np
```

In [11]:

```
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 [19]:

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

Out[19]:

```
array([   5,   55,  555, 5555])
```

Create an array whose initial content is random

In [14]:

```
print(np.empty(5))
```

```
[0.00000000e+000 1.00713714e-311 1.00713714e-311 1.00713714e-311
 1.38997796e+218]
```

Array with the range of values with even interval

In [18]:

```
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 [22]:

```
print(np.linspace(1,50,num=5))
```

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

Access and manipulate elements in the array

In [23]:

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

```
[ 55 555]
```

In [24]:

```
c[1]=10
print(c)
```

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

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

In [27]:

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

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

Out[27]:

```
(2, 3)
```

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

In [28]:

```
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 a given shape

In [33]:

```
shape=(3,4)
g= np.random.random(shape)
print(g)
```

```
[[0.36304744 0.97460891 0.59805882 0.89539644]
 [0.4149155  0.33783074 0.64561982 0.39354709]
 [0.98286216 0.6507239  0.1412075  0.14004377]]
```

Repeat each element of an array by a specified number of times using repeat() and tile() functions

In [39]:

```
h=np.repeat(c,3)
print(h)
i=np.tile(d,3)
print(i)
```

```
[   5    5    5   10   10   10  555  555  555 5555 5555 5555]
[[1 2 3 1 2 3 1 2 3]
 [4 5 6 4 5 6 4 5 6]]
```

The shape and size of an array

In [41]:

```
print(d.shape)
print(np.size(d))
```

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

An array that indicates the total number of elements in an array

In [42]:

```
np.array([np.size(d)])
```

Out[42]:

```
array([6])
```

To find the number of dimensions of the array

In [44]:

```
np.ndim(d)
```

Out[44]:

```
2
```

Create an array and reshape into a new array

In [45]:

```
j=np.arange(10)
j.reshape(5,2)
```

Out[45]:

```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

Create a null array of size 10

In [46]:

```
np.zeros(10,dtype=np.int64)
```

Out[46]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

Create any array with values ranging from 10 to 49 and print the numbers whose remainders are zero when divided by 7

In [48]:

```
k=np.arange(10,50)
print(k)
for _ in k:
    if(_%7==0):
        print(_)
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
14
21
28
35
42
49
```

Create an array and check any two conditions and print the output

In [49]:

```
for z in k:
    if(z%2==0):
        print(z)
    elif(z%3==0):
        print(z)
```

```
10
12
14
15
16
18
20
21
22
24
26
27
28
30
32
33
34
36
38
39
40
42
44
45
46
48
```

Use Arithmetic operator and print the output using array

In [50]:

```
a+b
```

Out[50]:

```
array([1, 1, 1], dtype=int64)
```

Use Relational operators and print the results using array

In [51]:

```
arr=np.arange(0,50)
v=arr[(arr>10)&(arr<31)]
v
```

Out[51]:

```
array([11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
       28, 29, 30])
```

Difference between python and ipython

```
Python:

Python is a general-purpose, high-level programming language.
It is the core language itself and does not include any specific interactive features.
Python can be executed in various ways, such as through scripts or interactive
command-line sessions.
When running Python code in a standard interactive shell or script, you type commands
and see output, but there might be limitations in terms of history management and code
exploration.


IPython (Interactive Python):

IPython is an interactive command-line shell specifically designed for Python with
enhanced features.
It provides an interactive environment with additional capabilities such as command
history, tab-completion, and easy access to help/documentation.
IPython also allows you to run system commands directly, access shell commands, and
interact with the operating system more conveniently.
It supports features like magic commands, which are special commands prefixed with
''%'or '%%' that offer extra functionality and control.
```