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

1.Create an array with zeros and ones and print the output

In [4]:
```
arr=np.zeros(3)
print(arr)
arr1=np.ones(3)
print(arr1)
```

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

2.Create an array and print the output

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

```
[1 2 3]
```

3. Create an array whose initial content is random and print the output

In [6]:
```
print(np.empty(2))
```

```
[ 1.34327943e+020 -5.35278720e-221]
```

4. Create an array with the range of values with even intervals

In [12]:
```
a=np.arange(2,11,2)
print(a)
```

```
[ 2  4  6  8 10]
```

5. create an array with values that are spaced linearly in a specified interval

In [11]:
```
b=np.linspace(1,20,num=10,dtype=np.int64)
print(b)
```

```
[ 1  3  5  7  9 11 13 15 17 20]
```

6. Access and manipulate elements in the array

In [12]:
```
c=np.array([1,2,3,4,5])
print(c[0:3])
```

```
[1 2 3]
```

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

In [18]:
```
d=np.array([[1,2,3],[4,5,6]])
print(d)
```

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

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

```
(2, 3)
```

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

In [21]:
```
e=np.arange(1,11)
print(e)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

In [23]:
```
f=np.linspace(1,11,num=5,dtype=np.int64)
print(f)
```

```
[ 1  3  6  8 11]
```

9. Create an array of random values between 0 and 1 in a given shape

In [24]:
```
g=np.array([1,0,1,0,1,0])
b=g.reshape(3,2)
print(b)
```

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

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

In [26]:
```
print(np.repeat(arr,3))
```

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

In [27]:
```
print(np.tile(arr,3))
```

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

11. How do you know the shape and size of an array?

In [29]:
```
print(np.shape(d))
```

```
(2, 3)
```

In [30]:
```
print(np.size(d))
```

```
6
```

12. Create an array that indicates the total number of elements in an array

In [16]:
```
arr=np.array([1,2,3,4,5,])
arr1=np.array([np.size(arr)])
print(arr1)
```

```
[5]
```

13. To find the number of dimensions of the array

In [5]:
```
print(np.ndim(arr))
```

```
1
```

14. Create an array and reshape into a new array

In [6]:
```
i=np.arange(6)
print(i)
```

```
[0 1 2 3 4 5]
```

In [7]:
```
j=i.reshape(3,2)
print(j)
```

```
[[0 1]
 [2 3]
 [4 5]]
```

15. Create a null array of size 10

In [9]:
```
print(np.zeros(10,dtype=np.int64))
```

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

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

In [14]:
```
arr=np.arange(10,50)
arr1=arr[arr%7==0]
print(arr1)
```

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

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

In [15]:
```
cond=arr1[(arr1>10)&(arr1<50)]
print(cond)
```

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

18. Use Arithmetic operator and print the output using array

In [18]:
```
m=np.array([1,2,3,4])
n=np.array([5,4,3,2])
print(m+n)
print(m-n)
print(m*n)
print(m/n)
```

```
[6 6 6 6]
[-4 -2  0  2]
[5 8 9 8]
[0.2 0.5 1.  2. ]
```

19. Use Relational operators and print the results using array

In [19]:
```
print(m[m>3])
print(m[m<4])
print(m[m>=12])
print(m[m<=32])
```

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

20. Difference between python and ipython

```
python:
    ->raditional text-based interactive mode or in script files.
    ->like running Python code interactively, history, and basic tab-
completion.
    ->Python comes pre-installed with most operating systems, and you can
install it separately if needed.
IPython:
    ->IPython supports rich display capabilities, allowing the display of
multimedia objects (images, videos), formatted text
```

```
    ->IPython can be installed as an additional package on top of Python, and
it integrates with the standard Python
    interpreter.
    ->IPython offers additional features such as advanced tab-completion
```

In [ ]: