20/07/2023

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

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

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

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

2. Create an array and print the output

In [7]:
```
arr = np.array([1,2,3,6,8,3,7,10,34,65])
print(arr)
```

```
[ 1  2  3  6  8  3  7 10 34 65]
```

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

In [9]:
```
arr1 = np.arange(21)
print(arr1)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

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

In [10]:
```
ab = np.arange(1,20,4)
print(ab)
```

```
[ 1  5  9 13 17]
```

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

In [31]:
```
abc = np.linspace(1,50,num=10,dtype=np.int64)
print(abc)
```
```
[ 1  6 11 17 22 28 33 39 44 50]
```

6. Access and manipulate elements in the array

In [21]:
```
arr1 = np.array([22,3,4,9,10,24,6])
arr1[1:5]
```
Out[21]:
```
array([ 3,  4,  9, 10])
```

In [32]:
```
arr1[1]
```
Out[32]:
```
3
```

In [24]:
```
arr1[-1]
```
Out[24]:
```
6
```

In [33]:
```
arr1[-2]
```
Out[33]:
```
24
```

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

In [30]:
```
arr2 = np.array([[1,2,3,4],[9,8,7,6]])
print(arr2)
```
```
[[1 2 3 4]
 [9 8 7 6]]
```

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

In [34]:
```
xyz = np.arange(25)
abcd = np.linspace(5,95,num=9,dtype=np.int64)
print(xyz)
print(abcd)
```
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24]
[ 5 16 27 38 50 61 72 83 95]
```

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

In [42]:
```
sdf = np.array([])
print(np.shape(sdf))
```

```
(0,)
```

In [47]:
```
sdfg = np.array([1])
print(np.shape(sdfg))
```

```
(1,)
```

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

In [40]:
```
lkk = np.array([33,44,55,66])
print(np.repeat(lkk,2))
```

```
[33 33 44 44 55 55 66 66]
```

In [41]:
```
print(np.tile(lkk,3))
```

```
[33 44 55 66 33 44 55 66 33 44 55 66]
```

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

shape: print number of colume and row in array

In [50]:
```
qwe = np.array([[2,3,4,5],[6,7,8,9]])
print(np.shape(qwe))
```

```
(2, 4)
```

size : print number of element in array

In [51]:
```
print(np.size(qwe))
```

```
8
```

12. Create an array that indicates the total

In [52]:
```
print(np.size(qwe))
```
```
8
```

13. To find the number of dimensions of the array

In [57]:
```
print(np.ndim(qwe))
```
```
2
```

14. Create an array and reshape into a new array

In [55]:
```
rty = np.array([11,12,13,14,15,16,17,18,19,20])
print(rty.reshape(5,2))
```
```
[[11 12]
 [13 14]
 [15 16]
 [17 18]
 [19 20]]
```

In [56]:
```
print(rty.reshape(2,5))
```
```
[[11 12 13 14 15]
 [16 17 18 19 20]]
```

15. Create a null array of size 10

In [59]:
```
po = np.zeros(10,dtype=np.int64)
print(po)
```
```
[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 [63]:
```
mnb = np.arange(10,50)
print(mnb)
```
```
[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]
```

In [64]:
```
ab = mnb[mnb % 7 == 0]
print(ab)
```

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

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

In [66]:
```
nbv = np.arange(10,50)
ac = nbv[(nbv > 20) & (nbv < 30)]
print(ac)
```

```
[21 22 23 24 25 26 27 28 29]
```

18. Use Arithmetic operator and print the output using array

In [68]:
```
bvc = np.arange(1,11)
cxz = np.arange(11,21)
print(bvc)
print(cxz)
print(bvc+cxz)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[11 12 13 14 15 16 17 18 19 20]
[12 14 16 18 20 22 24 26 28 30]
```

In [70]:
```
print(bvc-cxz)
print(bvc*cxz)
print(bvc/cxz)
```

```
[-10 -10 -10 -10 -10 -10 -10 -10 -10 -10]
[ 11  24  39  56  75  96 119 144 171 200]
[0.09090909 0.16666667 0.23076923 0.28571429 0.33333333 0.375
 0.41176471 0.44444444 0.47368421 0.5       ]
```

19. Use Relational operators and print the results using array

In [73]:
```
hg = np.array([10,20,30,40,50])
print(hg[hg<30])
print(hg[hg>30])
```

```
[10 20]
[40 50]
```

20. Difference between python and ipython

python

python extension is .py.python is a general-purpose programming language. It was created in the late 1980s by Guido van Rossum. It is now one of the most popular languages in the world. It is routinely used by system administrators and web developers. Also, many scientists are using Python thanks to libraries such as NumPy, SciPy, pandas, and matplotlib. The ease of use of Python and its dynamic nature make it a very productive language.

ipython

IPython is an interactive command-line terminal for Python. It was created by Fernando Perez in 2001. IPython offers an enhanced read-eval-print loop (REPL) environment particularly well adapted to scientific computing.

In [ ]:

```

```