D2-20/07/2023_Numpy

In [1]:

```
import numpy as np
```

1.Create an array with zeros and ones

In [7]:

```
a=np.zeros(3,dtype=np.int64)
b=np.ones(3,dtype=np.int64)
print(a)
print(b)
```

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

```
2.Create an array and print the output
```

In [8]:

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

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

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

In [12]:

```
print(np.empty(3))
```

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

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

In [17]:

```
d=np.arange(2,11,+2)
print(d)
```

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

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

In [18]:

```
print(np.linspace(1,50,num=20,dtype=np.int64))
```

```
[ 1  3  6  8 11 13 16 19 21 24 26 29 31 34 37 39 42 44 47 50]
```

6. Access and manipulate elements in the array

In [19]:

```
arr=np.array([10,20,30,40,50])
print(arr)
```

```
[10 20 30 40 50]
```

In [20]:

```
arr[0]
```

Out[20]:

```
10
```

In [21]:

```
arr[0:4]
```

Out[21]:

```
array([10, 20, 30, 40])
```

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

In [23]:

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

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

In [24]:

```
print(np.shape(a))
```

```
(2, 3)
```

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

In [25]:

```
arr=np.arange(1,11,+2)
print(arr)
```

```
[1 3 5 7 9]
```

In [26]:

```
print(np.linspace(1,10,num=5))
```

```
[ 1.    3.25  5.5   7.75 10.  ]
```

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

In [31]:

```
a=np.array([1,0,0,1,0,1])
b=a.reshape(2,3)
print(b)
```

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

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

In [32]:

```
print(np.repeat(arr,3))
```

```
[1 1 1 3 3 3 5 5 5 7 7 7 9 9 9]
```

In [33]:

```
print(np.tile(arr,3))
```

```
[1 3 5 7 9 1 3 5 7 9 1 3 5 7 9]
```

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

In [34]:

```
arr=np.array([10,20,30,40,50])
print(np.shape(arr))
```

```
(5,)
```

In [35]:

```
print(np.size(arr))
```

```
5
```

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

In [37]:

```
arr=np.array([10,20,30,40,50])
arr1=np.array([np.size(arr)])
print(arr1)
```

```
[5]
```

13. To find the number of dimensions of the array

In [40]:

```
print(np.ndim(b))
```

```
2
```

14. Create an array and reshape into a new array

In [41]:

```
a=np.array([1,0,6,8,44,12])
b=a.reshape(2,3)
print(b)
```

```
[[ 1  0  6]
 [ 8 44 12]]
```

15. Create a null array of size 10

In [43]:

```
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 [47]:

```
arr=np.arange(10,50)
print(arr)
```

```
[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 [48]:

```
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 [49]:

```
a=arr[(arr>20)&(arr<30)]
print(a)
```

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

18. Use Arithmetic operator and print the output using array

In [54]:

```
a=np.array([10,20,30])
b=np.array([50,60,70])
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

```
[ 60  80 100]
[-40 -40 -40]
[ 500 1200 2100]
[0.2        0.33333333 0.42857143]
```

19. Use Relational operators and print the results using array

In [55]:

```
a=np.array([10,20,30])
print(a[a>20])
print(a[a<20])
print(a[a>=20])
print(a[a<=20])
```

```
[30]
[10]
[20 30]
[10 20]
```

20. Difference between python and ipython"

```
ipython is interactive shell of python and python is a programming language.
ipython - interactive command-line terminal of python.
ipython as many features than python.
ipython efficentive in testing,debugging.
it has tab-machanism.
it has read-eval-print loop (REPL).
the ipython read the input from the user and give the result back to user. It works line by line.
ipython is can mange unstructured and semi-structured but python can only handle structured data.
ipython is used for data visulization process.
```

In [ ]: