D2-20/07/2023-NumPy

In [3]:

```
import numpy as np
```

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

In [7]:

```
a=np.zeros(5,dtype=np.int64)
print(a)
```

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

In [8]:

```
a=np.ones(5,dtype=np.int64)
print(a)
```

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

2. Create an array and print the output

In [9]:

```
arr=([1,2,3,4])
print(arr)
```

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

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

In [12]:

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

```
[1.06224594e-311 0.00000000e+000 1.89142482e-307]
```

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

In [14]:

```
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 [18]:

```
print(np.linspace(0,100,num=25,dtype=np.int64))
```

```
[  0   4   8  12  16  20  25  29  33  37  41  45  50  54  58  62  66  70
  75  79  83  87  91  95 100]
```

6. Access and manipulate elements in the array

In [19]:

```
arr[0]
```

Out[19]:

```
1
```

In [20]:

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

Out[20]:

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

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

In [25]:

```
print(np.arange(1,11))
```

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

In [27]:

```
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]:

```
b=np.array([1,1,1,0,1,0,1,0])
y=b.reshape(4,2)
print(y)
```

```
[[1 1]
 [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 [32]:

```
print(np.repeat(arr,2))
```

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

In [33]:

```
print(np.tile(arr,2))
```

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

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

In [34]:

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

```
(2, 3)
```

In [35]:

```
print(np.size(a))
```

```
6
```

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

In [39]:

```
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 [37]:

```
print(np.ndim(a))
```

```
2
```

14. Create an array and reshape into a new array

In [40]:

```
b=np.array([0,1,2,3,1,4,1,5])
y=b.reshape(4,2)
print(y)
```

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

15. Create a null array of size 10

In [41]:

```
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 [42]:

```
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 [44]:

```
a=arr[(arr>10)&(arr<40)]
print(a)
```

```
[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]
```

18. Use Arithmetic operator and print the output using array

In [46]:

```
z=np.array([1,2,3])
y=np.array([4,5,6])
print(z+y)
print(z-y)
print(z*y)
print(z/y)
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
```

19. Use Relational operators and print the results using array

In [48]:

```
print(arr[arr>4])
print(arr[arr<22])
print(arr[arr>=3])
print(arr[arr<=24])
```

```
[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]
[10 11 12 13 14 15 16 17 18 19 20 21]
[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]
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
```

20. Difference between python and ipython"

```
Python:
->Python is a programming language.
->It provides a straightforward interactive interpreter for executing Python code.
->The standard Python interpreter offers basic interactive features like command
execution, tab completion, and command history.
IPython (Interactive Python):
->IPython is an interactive shell Python.
->It enhances the interactive experience by providing additional features like tab
completion, syntax highlighting, and better command history.
->IPython supports running shell commands directly and introduces magic commands for
advanced functionalities like profiling, debugging, and time measurement.
```

In [ ]: