20/07/2023

Creates an array with zeros and ones

In [2]:

```
import numpy as np
```

In [4]:

```
array = np.zeros(5)
array[0] = 1
array[2] = 1
print(array)
```

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

Create an array.

In [13]:

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

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

Creates an array with random values.

In [5]:

```
array = np.random.randint(0, 10, 5)
print(array)
```

```
[3 8 8 6 7]
```

Creates an array with the range of values with even intervals.

In [6]:

```
array = np.arange(0, 10, 2)
print(array)
```

```
[0 2 4 6 8]
```

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

In [8]:

```
array = np.linspace(0, 10, 5)
print(array)
```

```
[ 0.   2.5  5.   7.5 10. ]
```

Access and manipulate elements in the array.

In [9]:

```
array = np.arange(0, 10)
print(array[2])
print(array[2:5])
array[2] = 100
print(array)
```

```
2
[2 3 4]
[  0   1 100   3   4   5   6   7   8   9]
```

Creates a 2-dimensional array and checks the shape of the array.

In [10]:

```
array = np.arange(0, 10).reshape(2, 5)
print(array)
print(array.shape)
```

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

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

In [11]:

```
array = np.arange(0, 10, 0.1)
print(array)
array = np.linspace(0, 10, 10)
print(array)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.  8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.8 8.9
 9.  9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.8 9.9]
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444  5.55555556
  6.66666667  7.77777778  8.88888889 10.        ]
```

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

In [12]:

```
array = np.random.randint(0, 1, (5, 5))
print(array)
```

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

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

In [14]:

```
array = np.arange(0, 5)
repeated_array = np.repeat(array, 2)
print(repeated_array)
tiled_array = np.tile(array, (2, 2))
print(tiled_array)
```

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

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

In [15]:

```
array = np.arange(0, 10)
print(array.shape)
print(array.size)
```

```
(10,)
10
```

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

In [16]:

```
array = np.arange(0, 10)
array_size = np.ones(array.shape) * array.size
print(array_size)
```

```
[10. 10. 10. 10. 10. 10. 10. 10. 10. 10.]
```

To find the number of dimensions of the array.

In [17]:

```
array = np.arange(0, 10)
print(array.ndim)
```

```
1
```

create a null array of size 10.

In [18]:

```
array=np.zeros(10)
print(array)
```

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

Create any array and check any two conditions.

In [19]:

```
array = np.arange(10, 49)
even_numbers = array % 2 == 0
greater_than_20 = array > 20
print(array[even_numbers & greater_than_20])
```

```
[22 24 26 28 30 32 34 36 38 40 42 44 46 48]
```

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

In [20]:

```
array = np.arange(10, 49)
remainders_of_zero = array % 7 == 0
print(array[remainders_of_zero])
```

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

Use Arithmetic operator and print the output using array.

In [21]:

```
array = np.arange(10, 49)
print(array + 10)
print(array * 2)
print(array / 2)
```

```
[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 50 51 52 53 54 55 56 57 58]
[20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66
 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96]
[ 5.   5.5  6.   6.5  7.   7.5  8.   8.5  9.   9.5 10.  10.5 11.  11.5
 12.  12.5 13.  13.5 14.  14.5 15.  15.5 16.  16.5 17.  17.5 18.  18.5
 19.  19.5 20.  20.5 21.  21.5 22.  22.5 23.  23.5 24. ]
```

Use Relational operators and print the results using array

```
In [22]: array = np.arange(10, 49)
         print(array < 20)
         print(array > 20)
         print(array <= 20)
         print(array >= 20)
```

```
[ True  True  True  True  True  True  True  True  True  True False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False]
[False False False False False False False False False False False  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True]
[ True  True  True  True  True  True  True  True  True  True  True False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False]
[False False False False False False False False False False  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True]
```

Difference between python and ipython.

```
In [23]: print("Python is a programming language, while IPython is an interactive shell for Python.")
```

```
Python is a programming language, while IPython is an interactive shell for Python.
```

```
In [ ]: 
```