20/07/2023

Day 2 Assignment

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

In [1]:

```
import numpy as np
```

In [8]:

```
arr=np.array(np.zeros(3,dtype=np.int8))
print(arr)
arr1=np.array(np.ones(3,dtype=np.int16))
print(arr1)
```

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

2. Create an array and print the output

In [9]:

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

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

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

In [18]:

```
a=np.array(np.random.rand(3,3))
print(a)
```

```
[[0.4951493  0.96792639 0.1021326 ]
 [0.91860373 0.86036017 0.61325031]
 [0.65620383 0.63053195 0.27269732]]
```

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

In [24]:

```
b=np.array(np.linspace(1,20,10,dtype=np.int32))
print(b)
```

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

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

In [21]:

```
c=np.array(np.arange(10))
print(c[::2])
```

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

6. Access and manipulate elements in the array

In [25]:

```
c[1]
```

Out[25]:

```
1
```

In [26]:

```
c[8]
```

Out[26]:

```
8
```

In [30]:

```
c[1]=11
c[9]=20
print(c)
```

```
[ 0 11  2  3  4  5  6  7  8 20]
```

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

In [31]:

```
d=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(d)
```

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

In [42]:

```
print(np.shape(d))
```

```
(3, 3)
```

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

In [37]:

```
print(np.array(np.arange(0,20,2)))
print(np.linspace(0,18,10,dtype=np.int64))
```

```
[ 0  2  4  6  8 10 12 14 16 18]
[ 0  2  4  6  8 10 12 14 16 18]
```

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

In [44]:

```
arv=np.array(np.random.rand(1,2))
print(arv)
print(np.shape(arv))
```

```
[[0.44733122 0.71630629]]
(1, 2)
```

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

In [49]:

```
print(c.repeat(3))#c=[0,11,2,3,4,5,6,7,8,20]
print(np.tile(c,3))
```

```
[ 0  0  0 11 11 11  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7
  8  8  8 20 20 20]
[ 0 11  2  3  4  5  6  7  8 20  0 11  2  3  4  5  6  7  8 20  0 11  2  3
  4  5  6  7  8 20]
```

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

In [53]:

```
print(np.shape(d))#using numpy library to access the shape() function to find the shape o
print(np.size(d))#using numpy library to access size() function to find the size of an ar
```

```
(3, 3)
9
```

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

In [56]:

```
dd=np.array([2,0,1,0,4,0,9,2])
print(dd)
print(np.size(dd))
```

```
[2 0 1 0 4 0 9 2]
8
```

13. To find the number of dimensions of the array

In [62]:

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

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

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

14. Create an array and reshape into a new array

In [66]:

```
arr3=np.array([7,2,9,2,1,8])
print(arr3.reshape(3,2))
```

```
[[7 2]
 [9 2]
 [1 8]]
```

15. Create a null array of size 10

In [69]:

```
arr4=np.array(np.zeros(10,dtype=np.int32))
print(arr4)
```

```
[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 [73]:

```
arr5=np.array(np.arange(10,50))
condition=arr5[arr5%7=0]
print(condition)
```

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

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

In [76]:

```
arr6=np.array(np.arange(1,101))
condition1=arr6[(arr6%2==0)&(arr6%5==0)]
print(condition1)
```

```
[ 10  20  30  40  50  60  70  80  90 100]
```

18. Use Arithmetic operator and print the output using array

In [81]:

```
arr7=np.array([10,20,30])
arr8=np.array([40,50,60])
sum1=arr7+arr8
print("Addition is:",sum1)
```

```
Addition is: [50 70 90]
```

19. Use Relational operators and print the results using array

In [82]:

```
arr9=np.array([9,18,27,36,45])
print(arr9[arr9>20])
```

```
[27 36 45]
```

20. Difference between python and ipython

In [83]:

```
''' .ipynb is very useful for data analysis or machine learning project
that requires a lot of interactive exploration and visualization whereas,
.py is useful to developing software projects that requires a more structured and modular
```

Out[83]:

```
' .ipynb is very useful for data analysis or machine learning project \nth
at requires a lot of interactive exploration and visualization whereas,\n.
py is useful to developing software projects that requires a more structur
ed and modular approach. '
```

In [ ]:

```

```