loopFunctionNumpy

July 19, 2024

[1]:
```
#loops

#for loops we know the number of times it will execute.

numbers = [1, 4, 6, 7, 76, 546, 43, 43,]

for i in numbers:
    print(i)
```

```
1
4
6
7
76
546
43
43
```

[2]:
```
#we are not sure how many times the loop will run to print a function

numba = 0

while numba < 10:
    print(numba)
    numba += 1
```

```
0
1
2
3
4
5
6
7
8
9
```

[3]:
```
i = 5

while i < 5:
    print(i)
    i +=1
```

[40]:
```
#Check this code and make corrections

#function is a block of code that can be reused in a program

#using python to find the factorial of a number
#factorial of 5 = 5*4*3*2*1
#factotiaal of zero is 1

number = int(input('Enter a number to find its factorial : '))
factorial = 1

if number == 0:
    print('print the factorial of 0 is 1')

else:
    for i in range(1, number+1):
        factorial = factorial*i
    print('The factorial of ',number,' is' ,factorial)
```

```
Enter a number to find its factorial : 5
The factorial of  5  is 120
```

factorial function

[5]:
```
def factorial_value(num):
    factorial = 1

    if num == 0:
        return factorial
    else:
        for i in range(1, num + 1):
            factorial *= i
        return factorial

print(factorial_value(5))
```

```
120
```

[6]:
```
 import numpy as np
```

List vs Numpy- Time Taken

Time taken by a list

[7]:
```
from time import process_time
```

[8]:
```
python_list = [i for i in range(10000)]

start_time = process_time()

python_list = [i+5 for i in python_list]

end_time = process_time()

print(end_time - start_time)
```

```
0.0
```

[9]:
```
np_array = np.array([i for i in range(10000)])
start_time = process_time()
np_array += 5
end_time = process_time()
print(end_time - start_time)
```

```
0.0
```

1 Performing operations in numpy arrays

[10]:
```
list1 = [1,2,3,4,5]
print(list1)
type(list1)
```

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

[10]:
```
list
```

[11]:
```
#arrays are simlar to matrics

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

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

[11]:
```
numpy.ndarray
```

[12]:
```
#creating a one dimentional array

a=np.array([1,2,3,4])
print(a)
a.shape   #it gives the number of rows and columns.

#it is a one dimesional one row four column
```

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

[12]:
```
(4,)
```

[13]:
```
b = np.array([(1,2,3,4,5), (0,6,7,8,9)])
print(b)
b.shape
```

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

[13]:
```
(2, 5)
```

[14]:
```
c = np.array([(1,2,3,4),(5,6,7,8)],dtype=float)
print(c)

#convert it to float ,dtype.float
```

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

Initial Placeholders in Numpy Array

[15]:
```
#create an array of zeros

x = np.zeros((4,5))
print(x)
```

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

[16]:
```
# create a numpy arrays of ones

y = np.ones((3,3))
print(y)
```

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

[17]:
```
#arrays with a particular vaue

c = np.full((4,5),5)
print(c)
```

```
[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

[18]:
```
#creating and identity matrix
#identity matrix means all diagonal value will be 1 other values will be zero
#It has same rows and columns
#use in various cases for programing


a = np.eye(4)
print(a)
```

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

[19]:
```
#create a numpy array with random values

b = np.random.random((3, 4))
print(b)
```

```
[[0.23431874 0.34512351 0.71755832 0.12534741]
 [0.41242199 0.65501121 0.49410837 0.73729994]
 [0.39789543 0.48926357 0.032683   0.06092061]]
```

[20]:
```
#creating a numpy values within a specific range
#The values will be the range of 10 and 100

c = np.random.randint(10, 100,(3,4))
print(c)
```

```
[[96 36 79 25]
 [99 58 32 40]
 [90 44 39 31]]
```

[21]:
```
#Array of evenly space values
d = np.linspace(10,30,5)
#it meas starting with 10 and spacing with 5 and stop at 30
print(d)
```

```
[10. 15. 20. 25. 30.]
```

[22]:
```
#Array of evenly space values that is arrange depending how you arrange your␣
 ↪numbers

e = np.arange(10,30,5)
print(e)
```

```
[10 15 20 25]
```

[23]:
```
#Converting a list to a numpy array
list2 = [3,6,87,8]
np_array = np.array(list2)

print(list2)
print(np_array)

type(np_array)
```

```
[3, 6, 87, 8]
[ 3  6 87  8]
```

[23]:
```
numpy.ndarray
```

[24]:
```
#Annalyzing  a numpy array

g = np.random.randint(10, 90,(5,5))
print(g)
```

```
[[78 74 52 52 69]
 [73 51 52 89 43]
 [11 41 10 10 73]
 [49 37 49 76 25]
 [24 36 57 28 34]]
```

[25]:
```
#Array dimension
print(g.shape)
```

```
(5, 5)
```

[26]:
```
#Number of dimension
print(g.ndim)   #it refers to rows and columns
```

```
2
```

[27]:
```
#Number of elements in an array
print(g.size)
```

```
25
```

[28]:
```
#Checking the data types

print(g.dtype)
```

```
int32
```

[29]:
```
#Mathematical operations in array

list1 = [1, 2, 3, 4, 5]
list3 = [5,6, 7, 8, 9, 10]
```

```
print(list1 + list3)  #It concanate or joins two list together
```

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

[30]:
```
a = np.random.randint(0,10,(3,3))
b = np.random.randint(10,20,(3,3))

print(a)
print(b)
```

```
[[2 2 5]
 [7 0 3]
 [7 8 3]]
[[12 19 14]
 [10 15 11]
 [13 10 18]]
```

[31]:
```
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
```

```
[[14 21 19]
 [17 15 14]
 [20 18 21]]
[[-10 -17  -9]
 [ -3 -15  -8]
 [ -6  -2 -15]]
[[24 38 70]
 [70  0 33]
 [91 80 54]]
[[0.16666667 0.10526316 0.35714286]
 [0.7        0.         0.27272727]
 [0.53846154 0.8        0.16666667]]
[[2 2 5]
 [7 0 3]
 [7 8 3]]
```

[32]:
```
a = np.random.randint(0,10,(3,3))
b = np.random.randint(10,20,(3,3))

print(a)
print(b)
```

```
[[1 9 1]
 [6 7 3]
 [4 7 3]]
```

```
[[18 11 18]
 [14 12 15]
 [16 14 14]]
```

[33]:
```
#Another method
print(np.add(a,b))
print(np.subtract(a,b))
```

```
[[19 20 19]
 [20 19 18]
 [20 21 17]]
[[-17  -2 -17]
 [ -8  -5 -12]
 [-12  -7 -11]]
```

2 Array Manipulation

[34]:
```
#Method 1
array = np.random.randint(0,10,(3,3))
print(array)
```

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

[35]:
```
trans = np.transpose(array)
print(trans)
```

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

[36]:
```
#Method2
array2 = np.random.randint(0,10,(3,3))
print(array2)
```

```
[[9 3 6]
 [1 6 3]
 [3 7 0]]
```

[37]:
```
trans2 = array2.T
print(trans2)
```

```
[[9 1 3]
 [3 6 7]
 [6 3 0]]
```

[38]:
```
#Reshapping Array
```

```
a = np.random.randint(0,10,(2,3))
print(a)
print(a.shape)
```

```
[[5 3 1]
 [7 8 5]]
(2, 3)
```

[39]:
```
b = a.reshape(3,2)
print(b)
print(b.shape)
```

```
[[5 3]
 [1 7]
 [8 5]]
(3, 2)
```

[ ]:
```

```