IMPORTING NUMPY

In [1]:
```
import numpy as np
print(np.__version__)#DUNDER METHOD #it is used for python version finding
```

```
1.20.1
```

In [2]:
```
a= np.arange(10,50) #arange function which is used to enter the array arange(initial
print("the required array is:")
print(a)
```

```
the required array is:
[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 [3]:
```
nz = np.nonzero([1,2,0,0,4,0])#index of non zero elements of a given list
print(nz)
```

```
(array([0, 1, 4], dtype=int64),)
```

In [4]:
```
b=np.random.rand(10,10)#it is used for getting random values(array(rows,columns))
print(b)
bmax,bmin=b.max(),b.min()#assigning the max and the min value
print("the max value is:",bmax)
print("the min value is:",bmin)
```

```
[[0.94895147 0.49969726 0.57496932 0.26460693 0.27409156 0.04324705
  0.467055   0.43940688 0.31496969 0.66192045]
 [0.38667188 0.24154895 0.57659236 0.37293087 0.80090546 0.45822087
  0.1232969  0.80268002 0.63461534 0.70194781]
 [0.73092101 0.83568097 0.7517411  0.60844299 0.64394151 0.97183622
  0.9178251  0.31315845 0.18065084 0.36403282]
 [0.1241375  0.13342538 0.59119912 0.88218655 0.44295346 0.29264519
  0.9059195  0.35042821 0.52947686 0.4111906 ]
 [0.99062118 0.11769776 0.90537233 0.36900004 0.64091175 0.43196751
  0.31185169 0.05481292 0.96537783 0.25163671]
 [0.67064175 0.64110362 0.29241371 0.09938774 0.27674688 0.63187294
  0.97258085 0.31111425 0.8720835  0.34329545]
 [0.87719283 0.11301791 0.53453974 0.76855833 0.57877122 0.55041669
  0.84400533 0.20289677 0.74254282 0.90231646]
 [0.00154566 0.09714696 0.27498414 0.91043001 0.33946242 0.32374477
  0.80961353 0.7947287  0.21371201 0.3265467 ]
 [0.350859   0.98303104 0.98714177 0.26902911 0.90771434 0.83829165
  0.5556882  0.96658876 0.85170912 0.66494449]
 [0.39063889 0.54233001 0.86327097 0.32299434 0.54839876 0.99399908
  0.01355774 0.74504019 0.62749921 0.01048043]]
the max value is: 0.9939990751669004
the min value is: 0.0015456579389089287
```

In [5]:
```
c= np.random.rand(30,)
print(c)
m = c.mean()
print ("the mean of vector of size 30 is:",m)# it gives the mean
```

```
[0.25849622 0.91425568 0.39775423 0.33954032 0.94053912 0.40716373
 0.50485842 0.92161384 0.01757558 0.97224158 0.7939866  0.59154221
 0.87207172 0.62145546 0.95463462 0.02122731 0.02375239 0.70447618
 0.68990878 0.47142326 0.09869069 0.11203556 0.68850651 0.5518136
 0.21639426 0.81006738 0.25363466 0.25628601 0.77923969 0.30659071]
the mean of vector of size 30 is: 0.5163925437826583
```

In [6]:
```
m1=np.random.rand(5,3)
print(m1)
m2=np.random.rand(3,2)
print(m2)
z=np.dot(m1,m2)# we use dot function for  multiplication of two matrix
print("the product of m2 and m1 is:\n",z)
```

```
[[0.17290764 0.38014779 0.88238395]
 [0.71102301 0.57058793 0.50110142]
 [0.75831988 0.62782708 0.84867024]
 [0.05590464 0.33464649 0.43814575]
 [0.9585497  0.75565524 0.3154823 ]]
[[0.44528445 0.56517018]
 [0.52668115 0.24672014]
 [0.58142763 0.97852443]]
the product of m2 and m1 is:
 [[0.79025216 1.05494661]
 [0.9084796  1.03296452]
 [1.16177307 1.41392194]
 [0.45589551 0.54289599]
 [1.00824677 1.03688622]]
```

In [7]:
```
arr1=np.random.randint(0,5,6)#random.randint(min_value,max_value,no_of_elements)
print(arr1)
arr2=np.random.randint(0,5,6)
print(arr2)
print(np.intersect1d(arr1, arr2))#intersect1d is a fuction used to find the elements
```

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

In [8]:
```
import numpy as np
x = np.random.randint(0,2,6)#(INITIAL VAL,FINAL VAL(NOT INCLUDED),NUMBER OF TERMS)
print("First array:")
print(x)
y = np.random.randint(0,2,6)
print("Second array:")
print(y)
print("Test above two arrays are equal or not!")
array_equal = np.allclose(x, y)#to check each element of array are same or not "allc
print(array_equal)
```

```
First array:
[1 0 0 1 1 1]
Second array:
[0 0 0 0 0 1]
Test above two arrays are equal or not!
False
```

In [9]:
```
A = np.random.randint(0,10,(3,4,3,4))# here random.randint(min_val,max_value,(dimens
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)#sum of the two over last two axis
print(sum)
```

```
[[64 55 62 65]
 [47 70 58 44]
 [58 76 49 47]]
```

In [10]:
```
A = np.ones((5,5,3))#print all ones values
B = 2*np.ones((5,5))#prints all 2* ones values
print(A * B[:,:,None])# now since we have already has to multiply the two dimensiona
```

```
[[[2. 2. 2.]
  [2. 2. 2.]
```

```
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]]
```

In [11]:

```
ar1=np.arange(1,25).reshape(8,3)
print(ar1)
sub_ar1=np.split(ar1,4)#the split function splits it into small chunks
print(sub_ar1)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]]
[array([[1, 2, 3],
       [4, 5, 6]]), array([[ 7,  8,  9],
       [10, 11, 12]]), array([[13, 14, 15],
       [16, 17, 18]]), array([[19, 20, 21],
       [22, 23, 24]])]
```

In [12]:

```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
for i in arr:
    if (i%2!=0):
        print(arr[i])
```

```
1
3
5
7
9
```

In [13]:

```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
for i in arr:
    if (i%2!=0):
        arr[i]=-1
print(arr)
```

```
[ 0 -1  2 -1  4 -1  6 -1  8 -1]
```

In [14]:
```
ar1d=np.arange(1,11)
print(ar1d)
print("converting 1D array to 2D")
ar2d=ar1d.reshape(2,5)
print(ar2d)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
converting 1D array to 2D
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

In [15]:
```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

# Answers
# Method 1:
stack=np.concatenate([a, b], axis=0)
print(stack)

# # Method 2:
# np.vstack([a, b])

# # Method 3:
# np.r_[a, b]
```

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

In [16]:
```
org_arr=np.arange(9).reshape(3,3)
print("Original array:")
print(org_arr)
org_arr[[0, 1],:] = org_arr[[1, 0],:]
print("\nAfter swapping arrays by rows:")
print(org_arr)
```

```
Original array:
[[0 1 2]
 [3 4 5]
 [6 7 8]]

After swapping arrays by rows:
[[3 4 5]
 [0 1 2]
 [6 7 8]]
```

In [17]:
```
from scipy import stats
array = np.random.randint(1,10,10)
print(array)

r1 = np.mean(array)
print("\nMean: ", r1)

r2 = np.std(array)
print("\nstd: ", r2)

r3 = np.median(array)
print("\nmedian: ", r3)
r4=stats.mode(array)
print("\nmode:",r4)
```

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

```
Mean:  4.6

std:  2.8

median:  4.0

mode: ModeResult(mode=array([2]), count=array([3]))
```