In [33]:

```
# To find current working dictionary.
import os
os.getcwd()
```

Out[33]:

```
'C:\\Users\\Admin'
```

In [34]:

```
os.chdir("F:\\python for data science\\jupyter_notebook_program")
```

In [35]:

```
import os
os.getcwd()
```

Out[35]:

```
'F:\\python for data science\\jupyter_notebook_program'
```

In [1]:

```
import numpy as np
```

In [4]:

```
np.array([5,9,13]) # it accept only integer . list is coverted in to array.
```

Out[4]:

```
array([ 5,  9, 13])
```

In [6]:

```
print(type[5,9,13])
```

```
type[5, 9, 13]
```

In [8]:

```
a=np.array([[1,2],[3,4],[5,6]])
print(" Type of a is : ", a.dtype)
print("\n",'*'*30)
print('\n', a)
```

```
 Type of a is :  int32

 ******************************

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

In [15]:

```
a=np.array([[1,2],[3,4],[5,6]],dtype=float)
print('\n', a)
print(a)
```

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

In [17]:

```
np.array(np.mat('4 5 6; 7 8 9')) # creating array using matrix
```

Out[17]:

```
array([[4, 5, 6],
       [7, 8, 9]])
```

In [20]:

```
a=[1,2,3,4,5,6]
c=np.array(a)
print(a)
print(type(c))
```

```
[1, 2, 3, 4, 5, 6]
<class 'numpy.ndarray'>
```

In [24]:

```
b=np.asarray(a)        # np.asarray is used to copy one array in to another variables
print(b)
print(type(b))
```

```
[1 2 3 4 5 6]
<class 'numpy.ndarray'>
```

In [ ]:

```
# NUMPY DATA TYPES
```

In [28]:

```
my_mat = [[1,2,3],[4,5,6],[7,8,9]]
mat=np.array(my_mat)
print('Type/class of this object is : ',type(mat))
print('Here is the matrix : \n', mat ,'\n .....................................\n')
print('The dimension of this matrix is : ', mat.ndim , sep=" ")
print('The size of matrix is : ', mat.size, sep=' ')
print('The datatypes of this matrix is ', mat.dtype)
```

```
Type/class of this object is :  <class 'numpy.ndarray'>
Here is the matrix :
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
 .....................................

The dimension of this matrix is :  2
The size of matrix is :  9
The datatypes of this matrix is  int32
```

In [5]:

```
np.ceil(29.3) # it converts intger into float value
```

Out[5]:

```
30.0
```

In [8]:

```
#arange and linespces
a=np.arange(5,40,5) # (staart stop and stepsize)from start to stop at  it write number
a
```

Out[8]:

```
array([ 5, 10, 15, 20, 25, 30, 35])
```

In [20]:

```
a=np.arange(2,10.5,0.4)
a
```

Out[20]:

```
array([ 2. ,  2.4,  2.8,  3.2,  3.6,  4. ,  4.4,  4.8,  5.2,  5.6,  6. ,
        6.4,  6.8,  7.2,  7.6,  8. ,  8.4,  8.8,  9.2,  9.6, 10. , 10.4])
```

In [22]:

```
a[::-1]   # reverse order
```

Out[22]:

```
array([10.4, 10. ,  9.6,  9.2,  8.8,  8.4,  8. ,  7.6,  7.2,  6.8,  6.4,
        6. ,  5.6,  5.2,  4.8,  4.4,  4. ,  3.6,  3.2,  2.8,  2.4,  2. ])
```

In [23]:

```
print("Every fifth number from 50 to 5 in reverse order \n", np.arange(50,0,-5))
```

```
Every fifth number from 50 to 5 in reverse order
 [50 45 40 35 30 25 20 15 10  5]
```

In [3]:

```
# line spaces   (it divides start and end point into number of element that is mension
print(np.linspace(10,40,5))
```

```
[10.  17.5 25.  32.5 40. ]
```

In [7]:

```
print(np.linspace(4.5,25.29,num=4, endpoint=True, retstep= True))
# IT Divides start point and end point in to number mension.
# It will find step size and include last element
```

```
(array([ 4.5 , 11.43, 18.36, 25.29]), 6.93)
```

In [35]:

```
# Matrix creation   it prints matrix of value zero having 5 rows and 6 columns
print("Matrix of zeros can printed like this\n ")
print(np.zeros((5,6)))
```

```
Matrix of zeros can printed like this

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

In [34]:

```
print("Matrix of one can printed like this\n ")
print(np.ones((5,6)))
```

```
Matrix of one can printed like this

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

In [32]:

```
print("vector of zeros\n")
print(np.zeros(5))
```

```
vector of zeros

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

print("Matrix of 5 can printed like this\n ") print(5* np.ones((3,4)))

In [39]:

```
# construct a diogonal matrix
x=print(np.arange(30).reshape(5,6))
x
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]
```

In [46]:

```
# Random number generation
np.random.seed(6) # each number is generated with same seed .check the following with
print("Random number generation from uniform distribution\n")
print(np.random.rand(4,4))
```

```
Random number generation from uniform distribution

[[0.89286015 0.33197981 0.82122912 0.04169663]
 [0.10765668 0.59505206 0.52981736 0.41880743]
 [0.33540785 0.62251943 0.43814143 0.73588211]
 [0.51803641 0.5788586  0.6453551  0.99022427]]
```

In [45]:
```
print("Random number generation from uniform distribution\n")
print(np.random.rand(4,4))
```

```
Random number generation from uniform distribution

[[0.56679677 0.11291833 0.06277624 0.57472422]
 [0.48548527 0.3014221  0.03979556 0.63389414]
 [0.12433568 0.01160584 0.27666659 0.30154554]
 [0.18031763 0.06697796 0.77923395 0.40074847]]
```

In [47]:
```
print("Number from normal distribution with zero mean and standard devition 1")
print(np.random.randn(4,4))  # randn always give normal distribution curve
```

```
Number from normal distribution with zero mean and standard devition 1
[[-0.33588161  1.23773784  0.11112817  0.12915125]
 [ 0.07612761 -0.15512816  0.63422534  0.810655  ]
 [ 0.35480861  1.81259031 -1.3564758  -0.46363197]
 [ 0.82465384 -1.17643148  1.56448966  0.71270509]]
```

In [48]:
```
print("randaom number integer",np.random.randint(5,10,6))
```

```
randaom number integer [6 8 6 5 5 7]
```

In [49]:
```
print("randaom number integer",np.random.randint(5,200,100))
# it will print 100 random number between 5 and 200
```

```
randaom number integer [140  41  80 189  38 119  73 166  20  52 130 154  75 134  14 1
29 181 106
  47  45  82  78 198  68  71 158 154  73 162 147  27   8 166  41 157  65
 191 153 107 116 126 161  55 116  39  33  60  51  97  68 113 177   6 150
  96  12 183 173  82  44 125  85  81  50 177 125  84  62 165 179 156 155
 181  44  42 112 131  38 174 160  27  56 100  76 138  77  84  44 104 175
  10  62 104 125 101  28 129  92  50 164]
```

In [53]:
```
print("randaom number integer matrix can be printed like this\n",np.random.randint(5,2
```

```
randaom number integer matrix can be printed like this
 [[ 19 129  99 159  69  96 135]
 [  7  69  44  16  91  43 175]
 [149  14 154  34 196 191  29]
 [199  25  39 104  46  67  41]
 [ 68 196 181 132 138  31 172]
 [102  38  43  63  29 166 174]]
```

In [56]:
```
#real time example
while True:
    otp1= np.random.randint(1000,10000,1)
    print('your OTP is: ',otp1)
    user_otp=(int(input("Enter your one time time password: ")))
    if otp1==user_otp:
        print(' you have successfully login')
        break
    else:
        print('you have entered incorrect OTP..ENTER IT AGAIN')
        continue
```

```
your OTP is:  [4292]
Enter your one time time password: 4000
you have entered incorrect OTP..ENTER IT AGAIN
your OTP is:  [4469]
Enter your one time time password: 4469
 you have successfully login
```

In [3]:
```
#Reshaping in other ways
import numpy as np
from numpy.random import randint as ri
a=ri(1,99,30)
a
```

Out[3]:
```
array([28, 98, 47, 97, 68, 89, 37, 69,  4, 78, 90,  2, 54, 40, 96, 71, 26,
       43, 30, 90, 10, 56, 16, 58, 51, 96, 22,  7, 79,  7])
```

In [60]:
```
c=a.reshape(5,6) # it will fix all 30 element in to 5 x 6 matrix. we can take 15 x 2 ,
c
```

Out[60]:
```
array([[79, 25, 30, 41, 59, 25],
       [71, 98,  6, 12, 43, 12],
       [45,  2,  8,  8, 97, 70],
       [64,  6,  9, 46, 75,  4],
       [37, 40, 67, 43, 23, 82]])
```

In [9]:
```
#Reshaping in other ways
import numpy as np
from numpy.random import randint as ri
a=ri(1,99,30)
a
c=a.reshape(2,3,5) # it will fix all 30 element in to 5 x 6 matrix. we can take 15 x 2
c
```

Out[9]:
```
array([[[80, 60, 53, 83, 90],
        [45, 54, 33, 39, 39],
        [83, 74, 84, 80, 18]],

       [[78, 51, 66, 93, 17],
        [50, 93, 23, 31, 32],
        [35, 28, 76, 84, 27]]])
```

In [61]:
```
print("The min of c is: ", c.min())
print("The max of c is: ", c.max())
print("The mean of c is: ", c.mean())
```

```
The min of c is:  2
The max of c is:  98
The mean of c is:  40.9
```

In [15]:
```
# sorting
M =ri(1,100,25).reshape(5,5)     # IT WILL PRINT MATRIX OF RANDOM INTEGER
print("\n 5x5 matrix of randon integer\n","-"*50,'\n', M)
```

```
 5x5 matrix of randon integer
 --------------------------------------------------
 [[69 18 96 89 38]
 [98 60 44 92 28]
 [53  9 90 13 46]
 [52 70 80 94 49]
 [80 64 97 74 22]]
```

In [16]:
```
print("\n Here is sorting of matrix along each row \n","-"*50,'\n',np.sort(M))
print("\n Here is sorting of matrix along each COLOUM \n","-"*50,'\n',np.sort(M, axis=
 Here is sorting of matrix along each row
 --------------------------------------------------
 [[18 38 69 89 96]
 [28 44 60 92 98]
 [ 9 13 46 53 90]
 [49 52 70 80 94]
 [22 64 74 80 97]]

 Here is sorting of matrix along each COLOUM
 --------------------------------------------------
 [[52  9 44 13 22]
 [53 18 80 74 28]
 [69 60 90 89 38]
 [80 64 96 92 46]
 [98 70 97 94 49]]
```

In [ ]:

```
# Indexing and slicing
```

In [17]:

```
arr=np.arange(13,30)
print("Array", arr)
```

```
Array [13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]
```

In [18]:

```
print("Element at 7th index",arr[7])
```

```
Element at 7th index 20
```

In [19]:

```
print("The element from 5th to 8th index are",arr[5:8])
```

```
The element from 5th to 8th index are [18 19 20]
```

In [20]:

```
print("Element form 4th index to last", arr[4:])
```

```
Element form 4th index to last [17 18 19 20 21 22 23 24 25 26 27 28 29]
```

In [27]:

```
print("element from last backword are", arr[-1:-17:-1])
print("element from last backword are", arr[-1::-2])
```

```
element from last backword are [29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14]
element from last backword are [29 27 25 23 21 19 17 15 13]
```

In [33]:

```
arr=np.arange(0,21,2)
print("New array :",arr)
```

```
New array : [ 0  2  4  6  8 10 12 14 16 18 20]
```

In [34]:

```
print("Element at 2nd ,4th and 8th index are: ", arr[[2,4,8]])
```

```
Element at 2nd ,4th and 8th index are:  [ 4  8 16]
```

In [37]:

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

Out[37]:

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

In [38]:

```
print('The element at 1st row and 2nd column of matrix are','-'*50,'\n',mat[1][2])
# index will start form 0 1 2 in both row and column wise
```

```
The element at 1st row and 2nd column of matrix are ---------------------------------
-----------------
 6
```

In [40]:

```
print('The element at 0 th row and 2nd column of matrix are','-'*50,'\n',mat[0][2])
# To select single element of martix
```

```
The element at 0 th row and 2nd column of matrix are --------------------------------
------------------
 3
```

In [49]:

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

Out[49]:

```
2
```

In [50]:

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

Out[50]:

```
1
```

In [52]:

```
print("Entire element at row 2",'\n',mat[2])
```

```
Entire element at row 2
 [7 8 9]
```

In [53]:

```
print("Entire element at column 1",'\n',mat[:,1:])
# position of column (,) is very important
# To select multiple element of martix
```

```
Entire element at column 1
 [[2 3]
 [5 6]
 [8 9]]
```

In [54]:

```
print("Entire element at column 2",'\n',mat[:,2:])
```

```
Entire element at column 2
 [[3]
 [6]
 [9]]
```

In [57]:

```
print("Entire element at row 1 and column 1",'\n',mat[1:,1:])
```

```
Entire element at row 1 and column 1
 [[5 6]
 [8 9]]
```

In [58]:

```
mat
```

Out[58]:

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

In [66]:

```
mat[1:2,1:2]
#matrix with row indices 1 and 2 and coloum 3 and 4
```

Out[66]:

```
array([[5]])
```

In [61]:

```
mat[0:2,0:2] # row 0 and row 1 :col0:col1  (# 0:2 means count row up to 1 . 2 will be
```

Out[61]:
```
array([[1, 2],
       [4, 5]])
```

In [67]:
```
mat[0:1,0:2]
```

Out[67]:
```
array([[1, 2]])
```

In [68]:
```
# updating the martrix
print("original matrix is \n",mat)
```

```
original matrix is
 [[ 1  2  3]
 [ 4  5 35]
 [ 7  8  9]]
```

In [64]:
```
mat[1][2]=35
print(mat)
```

```
[[ 1  2  3]
 [ 4  5 35]
 [ 7  8  9]]
```

In [69]:
```
mat[0][1]=135
mat
```

Out[69]:
```
array([[  1, 135,   3],
       [  4,   5,  35],
       [  7,   8,   9]])
```

In [7]:
```
# Subsetting
mat=np.array(ri(10,100,15).reshape(3,5))
mat
```

Out[7]:
```
array([[68, 74, 14, 52, 58],
       [34, 71, 42, 84, 43],
       [92, 40, 24, 81, 35]])
```

In [8]:
```
mat>50
```

Out[8]:
```
array([[ True,  True, False,  True,  True],
       [False,  True, False,  True, False],
       [ True, False, False,  True, False]])
```

In [11]:
```
mat[mat > 50]
```

Out[11]:
```
array([68, 74, 52, 58, 71, 84, 92, 81])
```

In [12]:
```
mat==58
```

Out[12]:
```
array([[False, False, False, False,  True],
       [False, False, False, False, False],
       [False, False, False, False, False]])
```

In [13]:
```
mat[mat==58]
```

Out[13]:
```
array([58])
```

In [18]:
```
print(np.where(mat==58))
```

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

In [19]:
```
print(np.where(mat==35))
```

```
(array([2], dtype=int64), array([4], dtype=int64))
```

In [ ]:
```
# matrix operation (universal combination)
```

In [36]:
```
mat1= np.array(ri(1,7,9).reshape(3,3))
mat2=np.array(ri(1,10,9).reshape(3,3))
print("\n-----------------------------------------------------------\nThe first matri
print("\n----------------------------------------------------------\nThe second matri
```

```
-----------------------------------------------------------
The first matrix is
 [[2 1 5]
 [5 3 4]
 [5 6 6]]

----------------------------------------------------------
The second matrix is
 [[6 3 3]
 [1 6 3]
 [4 8 6]]
```

In [37]:
```
print('The addition of two matrix is','mat1 + mat2','=','\n',mat1 + mat2)
```

```
The addition of two matrix is mat1 + mat2 =
 [[ 8  4  8]
 [ 6  9  7]
 [ 9 14 12]]
```

In [38]:
```
print('The multiplication of two matrix is','mat1 x mat2','=','\n',mat1*mat2)
```

```
The multiplication of two matrix is mat1 x mat2 =
 [[12  3 15]
 [ 5 18 12]
 [20 48 36]]
```

In [39]:
```
3*mat1-2*mat2
```

Out[39]:
```
array([[-6, -3,  9],
       [13, -3,  6],
       [ 7,  2,  6]])
```

In [40]:
```
3*mat1+2*mat2
```

Out[40]:
```
array([[18,  9, 21],
       [17, 21, 18],
       [23, 34, 30]])
```

In [41]:
```
mat1/mat2
```

Out[41]:
```
array([[0.33333333, 0.33333333, 1.66666667],
       [5.        , 0.5       , 1.33333333],
       [1.25      , 0.75      , 1.        ]])
```

In [56]:
```
#broadcasting
start=np.ones((3,3))
start
```

Out[56]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [54]:

```
one_row=np.array([1,9,3])
one_row
```

Out[54]:

```
array([1, 9, 3])
```

In [57]:

```
print(start + one_row)
```

```
[[ 2. 10.  4.]
 [ 2. 10.  4.]
 [ 2. 10.  4.]]
```

In [58]:

```
print(one_row + one_row.T)
```

```
[ 2 18  6]
```

In [59]:

```
print(one_row * one_row.T)
```

```
[ 1 81  9]
```

In [ ]:

```
#ARRAY MATH
```

In [63]:

```
mat1= np.array(ri(10,17,9).reshape(3,3))
mat2=np.array(ri(20,100, 9).reshape(3,3))
print("\n------------------------------------------------------------\nThe first matrix
print("\n-----------------------------------------------------------\nThe second matrix
```

```
------------------------------------------------------------
The first matrix is
 [[11 12 15]
 [11 10 11]
 [15 13 14]]

-----------------------------------------------------------
The second matrix is
 [[92 66 28]
 [28 80 34]
 [63 75 82]]
```

In [66]:

```
print("\n------------------------------------------------------------\nThe square root
print("\n-----------------------------------------------------------\nThe square of mat
```

```
------------------------------------------------------------
The square root first matrix is
 [[3.31662479 3.46410162 3.87298335]
 [3.31662479 3.16227766 3.31662479]
 [3.87298335 3.60555128 3.74165739]]

-----------------------------------------------------------
The square of matrix is
 [[8464 4356  784]
 [ 784 6400 1156]
 [3969 5625 6724]]
```

In [67]:

```
 a = np.array([1,2,3,5,8])

 print (a.ndim)
```

```
1
```

In [70]:

```
a = np.array([1, 2, 3], dtype = complex)
```

```
a
```

Out[70]:

```
array([1.+0.j, 2.+0.j, 3.+0.j])
```

In [71]:

```
dt = dt = np.dtype('i4')

print (dt)
```

```
int32
```

In [72]:

```
import numpy as np

a = np.array([1,2,3,5,8])

b = np.array([0,3,4,2,1])

c = a + b

c = c*a

print (c[2])
```

```
21
```

In [73]:

```
import numpy as np

a = np.array([[1,2,3],[0,1,4]])

b = np.zeros((2,3), dtype=np.int16)

c = np.ones((2,3), dtype=np.int16)

d = a + b + c

print (d[1,2] )
```

```
5
```

In [74]:

```
import numpy as np

ary = np.array([1,2,3,5,8])

ary = ary + 1

print (ary[1])
```

```
3
```

In [75]:

```
import numpy as np

a = np.array([[1,2,3],[0,1,4]])

print (a.size)
```

```
6
```

In [ ]: