## 4. diag ==> This function creates a two dimensional array with all the diagonal elements as the given value and rest are zero.

In [2]:
```
import numpy as np
a = np.diag([1,4,8,9])
a
```

Out[2]:
```
array([[1, 0, 0, 0],
       [0, 4, 0, 0],
       [0, 0, 8, 0],
       [0, 0, 0, 9]])
```

## 5. randint ==> This function is used to generate a random number between a given range.

## syntax == randint(min_value, max_value , total_numbers)

In [9]:
```
import numpy as np
a = np.random.randint(1,10,3)
a
```

Out[9]:
```
array([1, 7, 4])
```

## 6. rand() ==> This function is used to generate a random number between 0 to 1 .

In [11]:
```
import numpy as np
a = np.random.rand(5)
a
```

Out[11]:
```
array([0.33209445, 0.73741912, 0.79128029, 0.44168981, 0.14195926])
```

## 7. randn() ==> This function is used to generate a random number from -3 to close 3 . This may return positive or negative numbers as well

## syntax == random.randn(numbers of values)

In [12]:
```
import numpy as np
a = np.random.randn(5)
a
```

Out[12]:
```
array([-0.29988293, -0.75863977,  0.18269922,  0.26973985, -0.24720463])
```

# Reshaping Data

In [22]:
```
import numpy as np
a = np.random.randint(1,50,12)
a
```

Out[22]:
```
array([33, 22, 41, 12, 30, 27,  2,  3, 12, 24, 19, 42])
```

In [14]:
```
# n(rows) * n(columns) = n(total_elements)
```

In [15]:
```
a.shape
```

Out[15]:
```
(12,)
```

In [23]:
```
a = a.reshape(2,6)
a
```

Out[23]:
```
array([[33, 22, 41, 12, 30, 27],
       [ 2,  3, 12, 24, 19, 42]])
```

In [24]:
```
a = a.reshape(6,2)
a
```

Out[24]:
```
array([[33, 22],
       [41, 12],
       [30, 27],
       [ 2,  3],
       [12, 24],
       [19, 42]])
```

In [25]:
```
a = a.reshape(4,3)
a
```

Out[25]:
```
array([[33, 22, 41],
       [12, 30, 27],
       [ 2,  3, 12],
       [24, 19, 42]])
```

In [26]:
```
a = a.reshape(12,1)
a
```

Out[26]:
```
array([[33],
       [22],
       [41],
       [12],
       [30],
       [27],
       [ 2],
       [ 3],
       [12],
       [24],
       [19],
       [42]])
```

In [27]:
```
import numpy as np
a = np.random.randint(1,100,32)
a
```

Out[27]:
```
array([ 7, 28, 52, 77, 75, 45, 31, 51, 39, 31, 19, 41, 29, 15, 39, 68,  4,
       27, 52, 97, 42, 82, 62, 28, 38, 73, 21, 96, 26, 58, 48, 54])
```

In [32]:
```
a.shape
```

Out[32]:
```
(32,)
```

In [34]:
```
a = a.reshape(32,1)
a
```

Out[34]:
```
array([[ 7],
       [28],
       [52],
       [77],
       [75],
       [45],
       [31],
       [51],
       [39],
       [31],
       [19],
       [41],
       [29],
       [15],
       [39],
       [68],
       [ 4],
       [27],
       [52],
       [97],
       [42],
       [82],
       [62],
       [28],
       [38],
       [73],
       [21],
       [96],
       [26],
       [58],
       [48],
       [54]])
```

In [36]:
```
a = a.reshape(16,2)
a
```

Out[36]:
```
array([[ 7, 28],
       [52, 77],
       [75, 45],
       [31, 51],
       [39, 31],
       [19, 41],
       [29, 15],
       [39, 68],
       [ 4, 27],
       [52, 97],
       [42, 82],
       [62, 28],
       [38, 73],
       [21, 96],
       [26, 58],
       [48, 54]])
```

In [37]:
```
a = a.reshape(2,16)
a
```

Out[37]:
```
array([[ 7, 28, 52, 77, 75, 45, 31, 51, 39, 31, 19, 41, 29, 15, 39, 68],
       [ 4, 27, 52, 97, 42, 82, 62, 28, 38, 73, 21, 96, 26, 58, 48, 54]])
```

In [38]:
```
a = a.reshape(8,4)
a
```

Out[38]:
```
array([[ 7, 28, 52, 77],
       [75, 45, 31, 51],
       [39, 31, 19, 41],
       [29, 15, 39, 68],
       [ 4, 27, 52, 97],
       [42, 82, 62, 28],
       [38, 73, 21, 96],
       [26, 58, 48, 54]])
```

In [39]:
```
a = a.reshape(4,8)
a

# so on........
```

Out[39]:
```
array([[ 7, 28, 52, 77, 75, 45, 31, 51],
       [39, 31, 19, 41, 29, 15, 39, 68],
       [ 4, 27, 52, 97, 42, 82, 62, 28],
       [38, 73, 21, 96, 26, 58, 48, 54]])
```

## principle of -1

In [40]:
```
a = a.reshape(-1,4)
a
```

Out[40]:
```
array([[ 7, 28, 52, 77],
       [75, 45, 31, 51],
       [39, 31, 19, 41],
       [29, 15, 39, 68],
       [ 4, 27, 52, 97],
       [42, 82, 62, 28],
       [38, 73, 21, 96],
       [26, 58, 48, 54]])
```

## Seed Function() ==> We know that randit function generates random numbers. Everytime we run the program , now set of random number is generated. So, solve problem we will use seed function

In [42]:
```
import numpy as np

np.random.seed(12)
a=np.random.randint(1,100,10)
a
```

Out[42]:
```
array([76, 28,  7,  3,  4, 68, 77, 49, 23, 50])
```

In [46]:
```
np.random.seed(111)
a=np.random.randint(1,500,30)
print(a)
a.reshape(6,5)
```

```
[341 365 469 213  87 276 170 323 119 467 297 456 442   8 269 267 450 369
 217 285 271 419  22  55 201  38 461 231 187 269]
```

Out[46]:
```
array([[341, 365, 469, 213,  87],
       [276, 170, 323, 119, 467],
       [297, 456, 442,   8, 269],
       [267, 450, 369, 217, 285],
       [271, 419,  22,  55, 201],
       [ 38, 461, 231, 187, 269]])
```

# View as Copy ==> When we slice a sub_array from an array, it may be done by two ways.

In [54]:
```
import numpy as np
a = np.array([10 , 20 , 30 , 40 , 50 , 60 , 70 , 80])
b = a[3:6]
b[:] = 0
print("a : " , a)
print("b : " , b)
```

```
a :  [10 20 30  0  0  0 70 80]
b :  [0 0 0]
```

# copy

In [58]:
```
a = np.array([10 , 20 , 30 , 40 , 50 , 60 , 70 ,80])
b = a[3:6].copy()
b[:] = 0
print("a : " , a)
print("b : " , b)
```

```
a :  [10 20 30 40 50 60 70 80]
b :  [0 0 0]
```

# conditional selection

In [59]:
```
import numpy as np
a = np.arange(1,16)
a
```

Out[59]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [60]:
```
a>10
```

Out[60]:
```
array([False, False, False, False, False, False, False, False, False,
       False,  True,  True,  True,  True,  True])
```

In [61]:
```
a<10
```

Out[61]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
       False, False, False, False, False, False])
```

In [65]:
```
b = a>10
a[b]
```

Out[65]:
```
array([11, 12, 13, 14, 15])
```

In [66]:
```
a[a%2 == 0]
```

Out[66]:
```
array([ 2,  4,  6,  8, 10, 12, 14])
```

# Operations on array

In [67]:
```
import numpy as np
a = np.arange(1,5)
a*2
```

Out[67]:
```
array([2, 4, 6, 8])
```

In [68]:
```
a+2
```

Out[68]:
```
array([3, 4, 5, 6])
```

In [69]:
```
a**2
```

Out[69]:
```
array([ 1,  4,  9, 16])
```

In [71]:
```
a = np.array([1,2,3,4]).reshape(2,2)
a
```

Out[71]:
```
array([[1, 2],
       [3, 4]])
```

In [73]:
```
b = np.array([5,6,7,8]).reshape(2,2)
b
```

Out[73]:
```
array([[5, 6],
       [7, 8]])
```

In [74]:
```
a+b
```

Out[74]:
```
array([[ 6,  8],
       [10, 12]])
```

In [75]:
```
a*b
```

Out[75]:
```
array([[ 5, 12],
       [21, 32]])
```

In [76]:
```
b-a
```

Out[76]:
```
array([[4, 4],
       [4, 4]])
```

In [77]:
```
b/a
```

Out[77]:
```
array([[5.        , 3.        ],
       [2.33333333, 2.        ]])
```

In [78]:
```
a*b
```

Out[78]:
```
array([[ 5, 12],
       [21, 32]])
```

In [79]:
```
a.dot(b)
```

Out[79]:
```
array([[19, 22],
       [43, 50]])
```

# some more important numpy functions

In [81]:
```
import numpy as np
a = np.array([10,20,30,40,50])
np.min(a)
```

Out[81]:
```
10
```

In [82]:
```
np.max(a)    #it will return maximum value of the array
```

Out[82]:
```
50
```

In [89]:
```
np.argmin(a)  # it will return the indexing of minimum value
```

Out[89]:
```
0
```

In [90]:
```
np.argmax(a)  # it will return the indexing of miaximum value
```

Out[90]:
```
4
```

In [91]:
```
np.sqrt(a)
```

Out[91]:
```
array([3.16227766, 4.47213595, 5.47722558, 6.32455532, 7.07106781])
```

In [92]:
```
np.sin(a)
```

Out[92]:
```
array([-0.54402111,  0.91294525, -0.98803162,  0.74511316, -0.26237485])
```

# Linspace() ==> This function returns value between a given range and with a same gap between consicutive elements.

In [95]:
```
import numpy as np

a = np.linspace(1,2,5)
a
```

Out[95]:
```
array([1.  , 1.25, 1.5 , 1.75, 2.  ])
```

In [102]:
```
import numpy as np

np.random.seed(122)
a = np.random.randint(1,21,9).reshape(3,3)
a
```

Out[102]:
```
array([[16, 11, 13],
       [17, 13, 16],
       [ 3, 16, 11]])
```

In [103]:
```
np.sum(a, axis = 1)
```

Out[103]:
```
array([40, 46, 30])
```

In [104]:
```
np.min(a)
```

Out[104]:
```
3
```

In [106]:
```
np.cumsum(a)
```

Out[106]:
```
array([ 16,  27,  40,  57,  70,  86,  89, 105, 116])
```

In [107]:
```
a
```

Out[107]:
```
array([[16, 11, 13],
       [17, 13, 16],
       [ 3, 16, 11]])
```

# np.unique(arr, return_index = True , return_couints = True)

return 3 array. 1 .the array with unique values. 2.The array with respective index value. 3.The array with counting of frequency.

In [108]:
```
a = np.array([10,20,30,40,10,20,50])
np.unique(a , return_index = True , return_counts = True)
```

Out[108]:
```
(array([10, 20, 30, 40, 50]),
 array([0, 1, 2, 3, 6], dtype=int64),
 array([2, 2, 1, 1, 1], dtype=int64))
```

# horizontal and vertical stacking

In [97]:
```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])
a
```

Out[97]:
```
array([1, 2, 3, 4])
```

In [98]:
```
b
```

Out[98]:
```
array([5, 6, 7, 8])
```

In [99]:
```
np.hstack((a,b))
```

Out[99]:
```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

In [100]:
```
np.vstack((a,b))
```

Out[100]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```