In [ ]:
```
Numpy >> Numerical Python

import numpy
import numpy as np

Numpy : Numpy is a base library in python that deals with numbers. it is used to perform different mathematical operations on n dimensional array.

A numpy array is a homogeneous data type array which means it assigns one perticular kind of data type to an entire array.

It is time efficients.
It has so many mathematical operations.
-Probability
-Linear algebra
- matrix related operators
-Geometry
- Statastics
- Logarithmic maths
-Random number generation
```

In [1]:
```
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
list1 = [2,3,4,5]
type(list1)
```

Out[2]:
```
list
```

In [3]:
```
array =np.array([2,3,4,5])
type(array)
```

Out[3]:
```
numpy.ndarray
```

how to check the dimensions of array

In [4]:
```
array =np.array([2,3,4,5])
array.ndim
```

Out[4]:
```
1
```

In [5]:
```
array =np.array([[2,3,4,5]])
array.ndim
```

Out[5]:
```
2
```

In [6]:
```
array =np.array([[2,3],
                 [4,5]])
array.ndim
```

Out[6]:
```
2
```

In [7]:
```
array =np.array([[[2,3],
                  [4,5]]])
array.ndim
```

Out[7]: 3

how to convert a lower dimension into higher dimension array

In [8]:
```
array =np.array([2,3,4,5],ndmin=2)
array.ndim
```

Out[8]: 2

In [9]:
```
array
```

Out[9]: array([[2, 3, 4, 5]])

In [12]:
```
array =np.array([2,3,4,5],ndmin=3)
array.ndim,array
```

Out[12]: (3, array([[[2, 3, 4, 5]]]))

how to convert a list into array

In [13]:
```
list1 = [2,3,4,5]
array = np.array(list1)
array
```

Out[13]: array([2, 3, 4, 5])

In [14]:
```
l1 = 2,3,4
arr = np.array(l1)
type(arr)
```

Out[14]: numpy.ndarray

Can an array have multiple data types?

In [21]:
```
# bcz 1 itme is string so all items are converted into string
array =np.array([2,3.9,'4',-5])
print(array)
type(array[1])
```

```
['2' '3.9' '4' '-5']
```

Out[21]: numpy.str_

In [17]:
```
array =np.array([2,3.9,4,-5])
array
```

Out[17]:
```
array([ 2. ,  3.9,  4. , -5. ])
```

In [18]:
```
array =np.array([2,3.9,4+3j,-5])
array
```

Out[18]:
```
array([ 2. +0.j,  3.9+0.j,  4. +3.j, -5. +0.j])
```

In [19]:
```
array =np.array([2,3.9,4,-5],dtype='int')
array
```

Out[19]:
```
array([ 2,  3,  4, -5])
```

In [20]:
```
array =np.array([2,3,4,-5],dtype='float')
array
```

Out[20]:
```
array([ 2.,  3.,  4., -5.])
```

In [22]:
```
array =np.array([2,3,4,-5],dtype='complex')
array
```

Out[22]:
```
array([ 2.+0.j,  3.+0.j,  4.+0.j, -5.+0.j])
```

In [23]:
```
array =np.array([2,3,4,-5],dtype='str')
array
```

Out[23]:
```
array(['2', '3', '4', '-5'], dtype='<U2')
```

In [27]:
```
# object sets the data types to default data type of the elements int the array
array =np.array([2,3,'4',-5],dtype='object')
print(array)
type(array[1])
type(array[2])
```

```
[2 3 '4' -5]
```

Out[27]:
```
str
```

In [29]:
```
import pandas as pd
array =np.array([2,3,'4',-5])
df = pd.DataFrame(array)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       4 non-null      object
dtypes: object(1)
memory usage: 160.0+ bytes
```

Indexing

```
one its 1d, it works the same way as list indexing
```

In [31]:
```
array =np.array([2,3,4,5])
array[0]
```

Out[31]:
```
2
```

In [32]:
```
array[-1]
```

Out[32]:
```
5
```

In [34]:
```
array[-1:1:-1]
```

Out[34]:
```
array([5, 4])
```

In [35]:
```
array = np.array([[[2,3,6,6],
                   [3,2,3,9],
                   [5,6,5,0]],

                  [[2,3,4,2],
                   [3,3,4,6],
                   [7,8,9,0]]])
print(array)
```

```
[[[2 3 6 6]
  [3 2 3 9]
  [5 6 5 0]]

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

In [36]:
```
array[0]
```

Out[36]:
```
array([[2, 3, 6, 6],
       [3, 2, 3, 9],
       [5, 6, 5, 0]])
```

In [38]:
```
array[0][0][1]
```

Out[38]:
```
3
```

In [39]:
```
array[1][-1][1]
```

Out[39]:
```
8
```

slicing

In [40]:
```
array =np.array([2,3,4,5])
array[1:-1]
```

Out[40]:
```
array([3, 4])
```

In [41]:
```
array =np.array([2,3,4,5,1,4,0,8,6,3,4])
array[1:10:2]
```

Out[41]:
```
array([3, 5, 4, 8, 3])
```

In [51]:
```
arr = np.array([[2, 3, 6, 6],
                [3, 2, 3, 9],
                [5, 6, 5, 0]])

arr[1:][1:][0][1:]
```

Out[51]:
```
array([6, 5, 0])
```

In [52]:
```
array = np.array([[[2,3,6,6],
                   [3,2,3,9],
                   [5,6,5,0]],

                  [[2,3,4,2],
                   [3,3,4,6],
                   [7,8,9,0]]])
print(array)
```

```
[[[2 3 6 6]
  [3 2 3 9]
  [5 6 5 0]]

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

In [55]:
```
array[0][1][0:2]
```

Out[55]:
```
array([3, 2])
```

In [62]:
```
# in this example we have 2 rows & 3 columns
arr = np.array([[2,3,4],
                [5,6,7]])
arr                         # horizontal are rows & vertical are columns
```

Out[62]:
```
array([[2, 3, 4],
       [5, 6, 7]])
```

In [63]:
```
arr[:,:]          # all rows & all columns
```

Out[63]:
```
array([[2, 3, 4],
       [5, 6, 7]])
```

In [64]:
```
arr = np.array([[2,3,4],
                [5,6,7],
                [8,9,0]])

arr[0:2,1:]           # initially we describe rows & then we describe columns
```

Out[64]:
```
array([[3, 4],
       [6, 7]])
```

In [65]:
```
array = np.array([[[2,3,6,6],
                   [3,2,3,9],
                   [5,6,5,0]],

                  [[2,3,4,2],
                   [3,3,4,6],
                   [7,8,9,0]]])
print(array[0:,1:,1:-1])
```

```
[[[2 3]
  [6 5]]

 [[3 4]
  [8 9]]]
```

In [66]:
```
array[:,1:,2:]
```

Out[66]:
```
array([[[3, 9],
        [5, 0]],

       [[4, 6],
        [9, 0]]])
```

In [68]:
```
array = np.array([[[[2,3,6,6],
                    [3,2,3,9],
                    [5,6,5,0]],

                   [[2,3,4,2],
                    [3,3,4,6],
                    [7,8,9,0]]],

                  [[[0,0,0,0],
                    [0,0,0,0],
                    [0,0,0,0]],

                   [[2,3,4,2],
                    [3,3,4,6],
                    [7,8,9,0]]]])

array[:,1:,-1:,1:-1]
```

Out[68]:
```
array([[[[8, 9]]],


       [[[8, 9]]]])
```

In [71]:
```
array[:,1:,:2,:2]
```

Out[71]:
```
array([[[[2, 3],
         [3, 3]]],


       [[[2, 3],
         [3, 3]]]])
```

Reshape

In [73]:
```
arr = np.array([1,2,3,4,5,6,7,8,9,0,1,2])
len(arr)
```

Out[73]:
```
12
```

In [74]:
```
arr = arr.reshape(1,12)
arr
```

Out[74]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2]])
```

In [75]:
```
arr = arr.reshape(12,1)
arr
```

Out[75]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9],
       [0],
       [1],
       [2]])
```

In [79]:
```
arr = arr.reshape(2,6)
arr = arr.reshape(6,2)
arr = arr.reshape(3,4)
arr = arr.reshape(4,3)
arr
```

Out[79]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9],
       [0, 1, 2]])
```

In [80]:
```
arr = np.array([1,2,3,4,5,6,7,8,9,0,2])
arr.reshape(3,4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-80-638af8a2e3aa> in <module>
      1 arr = np.array([1,2,3,4,5,6,7,8,9,0,2])
----> 2 arr.reshape(3,4)

ValueError: cannot reshape array of size 11 into shape (3,4)
```

In [81]:
```
arr = np.array([1,2,3,4,5,6,7,8,9,0,2,1])
arr.reshape(3,2,2)
```

Out[81]:
```
array([[[1, 2],
        [3, 4]],

       [[5, 6],
        [7, 8]],

       [[9, 0],
        [2, 1]]])
```

In [82]:
```
arr.reshape(2,3,2)
```

Out[82]:
```
array([[[1, 2],
        [3, 4],
        [5, 6]],

       [[7, 8],
        [9, 0],
        [2, 1]]])
```

In [83]:
```
arr.reshape(2,2,3)
```

Out[83]:
```
array([[[1, 2, 3],
        [4, 5, 6]],

       [[7, 8, 9],
        [0, 2, 1]]])
```

converting n rray into list

In [89]:
```
arr = np.array([1,2,3,4,5,6,7,8,9,0,2,1])
arr
```

Out[89]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1])
```

In [90]:
```
list(arr)
```

Out[90]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1]
```

In [91]:
```
my_list = arr.tolist()
my_list
```

Out[91]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 2, 1]
```

In [92]:
```
array= np.array([[[1, 2],
        [3, 4]],

       [[5, 6],
        [7, 8]],

       [[9, 0],
        [2, 1]]])

array.tolist()
```

Out[92]:
```
[[[1, 2], [3, 4]], [[5, 6], [7, 8]], [[9, 0], [2, 1]]]
```

how to create an empty array

In [95]:
```
array = np.array([],dtype='int')
array
```

Out[95]:
```
array([], dtype=int32)
```

creating an array of matrix zeros

In [99]:
```
arr = np.zeros((5),dtype='int')
arr
```

Out[99]:
```
array([0, 0, 0, 0, 0])
```

In [98]:
```
arr = np.zeros((5,5),dtype='int')
arr
```

Out[98]:
```
array([[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 [102]:
```
arr = np.zeros((5,2,3),dtype='int')
arr
```

Out[102]:
```
array([[[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]]])
```

creating a matrix of ones

In [103]:
```
arr = np.ones(5,dtype='float')
arr
```

Out[103]:
```
array([1., 1., 1., 1., 1.])
```

In [104]:
```
arr = np.ones((5,6),dtype='int')
arr
```

Out[104]:
```
array([[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 [105]:
```
arr = np.ones((5,2,3),dtype='int')
arr
```

Out[105]:
```
array([[[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]]])
```

creating a sequence of array using arange function

```
# similar to range function we have arange function in numpy
```

In [106]:
```
np.array(list(range(1,11)))
```

Out[106]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [107]:
```
np.arange(1,11)
```

Out[107]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [108]:
```
np.arange(1,11,2)
```

Out[108]:
```
array([1, 3, 5, 7, 9])
```

In [112]:
```
numbers = np.arange(-1,-11,-2,dtype='float')
print(numbers)
len(numbers)
```

```
[-1. -3. -5. -7. -9.]
```

Out[112]:
```
5
```

In [114]:
```
np.arange(1,10).reshape(3,3)
```

Out[114]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

eye()

```
it will return an array with ones in diagonal & zeros elsewhere
```

In [115]:
```
np.eye(3,dtype='I')
```

Out[115]:
```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]], dtype=uint32)
```

In [116]:
```
np.eye(5,dtype='I')
```

Out[116]:
```
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1]], dtype=uint32)
```

In [118]:
```
np.eye(4,5,dtype='I')
```

Out[118]:
```
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0]], dtype=uint32)
```

In [119]:
```
np.eye(4,5,dtype='I',k=2)        # positive values of k -- shift along column
```

Out[119]:
```
array([[0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0]], dtype=uint32)
```

In [120]:
```
np.eye(4,5,dtype='I',k=-2)       # negative values of k -- shift is along rows
```

Out[120]:
```
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0]], dtype=uint32)
```

Indentity()

```
It has equal number of rows & columns & ones in diagonal & zeros elsewhere
```

In [121]:
```
np.identity(3)
```

Out[121]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [123]:
```
np.identity(5,dtype='int')
```

Out[123]:
```
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1]])
```

In [126]:
```
np.identity(5,4,dtype='int')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-126-1ce2f853d460> in <module>
----> 1 np.identity((5,4),dtype='int')

~\anaconda3\lib\site-packages\numpy\core\numeric.py in identity(n, dtype)
   2112     """
   2113     from numpy import eye
-> 2114     return eye(n, dtype=dtype)
   2115
   2116

~\anaconda3\lib\site-packages\numpy\lib\twodim_base.py in eye(N, M, k, dtype, order)
    197     if M is None:
    198         M = N
--> 199     m = zeros((N, M), dtype=dtype, order=order)
    200     if k >= M:
    201         return m

TypeError: 'tuple' object cannot be interpreted as an integer
```

In [127]:
```
np.identity(5,dtype='int',k=2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-127-7a68686f1fe3> in <module>
----> 1 np.identity(5,dtype='int',k=2)

TypeError: identity() got an unexpected keyword argument 'k'
```

In [ ]:
```

```