NumPy stands for Numerical Python. It is one of the most important foundational packages for numerical computing & data analysis in Python. Most computational packages providing scientific functionality use NumPy's array objects as the lingua franca for data exchange

Types of Numpy Array: 1D Array, 2D Array, 3D Array.

One dimensional array

In [3]:

```python
import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a)
```

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

In [3]:

```python
b = np.array((1, 2, 3, 4, 5))
print(b)
```

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

In [5]:

```python
c = np.fromiter((a for a in range(8)), int)
print(c)
```

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

Two dimensional array

In [6]:

```python
list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
list_3 = [9, 10, 11, 12]
print(np.array([list_1, list_2, list_3]))
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [7]:

```python
print(np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]))
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [7]:

```
d = np.empty([4, 3], dtype=int)
print(d)
```

```
[[4128860 6029375 3801155]
 [5570652 6619251 7536754]
 [4259932 5046340 5111881]
 [8257609      49 3342437]]
```

In [6]:

```
#create a NumPy array using numpy.arange()
print(np.arange(1, 10))
```

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

In [13]:

```
#create a NumPy array using numpy.linspace()
print(np.linspace(1, 10, 3))
```

```
[ 1.   5.5 10. ]
```

In [15]:

```
#create a NumPy array using numpy.zeros()
print(np.zeros(7, dtype=int))
```

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

In [26]:

```
#create a NumPy array using numpy.ones()
print(np.ones(5, dtype=int))
```

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

In [52]:

```
#create a NumPy array using numpy.random.rand()
g = np.random.rand(5)
print(g)
```

```
[0.84688659 0.29959454 0.93712896 0.19527563 0.55865068]
```

In [24]:

```
#create a NumPy array using numpy.random.randint()
print(np.random.randint(5, size=5))
```

```
[1 3 3 1 1]
```

In [29]:

```
#create a NumPy array using numpy.zeros()
print(np.arange(1, 10))
```

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

```
In [30]: #create a NumPy array using numpy.ones()
         print(np.ones([4, 3], dtype = np.int32))
```

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

```
In [35]: #create a NumPy array using numpy.full()
         print(np.full([5, 5], 7, dtype = int))
```

```
[[7 7 7 7 7]
 [7 7 7 7 7]
 [7 7 7 7 7]
 [7 7 7 7 7]
 [7 7 7 7 7]]
```

```
In [47]: #create a NumPy array(identity matrix) using numpy.eye()
         e = print(np.eye(5))
```

```
[[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.]]
```

INSPECTING ARRAYS

```
In [46]: si = d.size
         print(si)
```

```
12
```

```
In [49]: len(d)
```

```
Out[49]: 4
```

```
In [50]: sh = d.shape
         print(sh)
```

```
(4, 3)
```

In [53]:
```
dt = g.dtype
print(dt)
```

```
float64
```

In [54]:
```
ch_dt = d.astype('float64')
print(ch_dt)
```

```
[[ 1.  2.  3.]
 [ 4.  5.  6.]
 [ 7.  8.  9.]
 [10. 11. 12.]]
```

In [55]:
```
lis= b.tolist()
print(lis)
```

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

Saving and loading File

In [57]:
```
sav = np.save("file", np.arange(5))
print(sav)
```

```
None
```

In [58]:
```
np.load("file.npy")
```

Out[58]:
```
array([0, 1, 2, 3, 4])
```

In [63]:
```
np.loadtxt('file.txt')
```

Out[63]:
```
array(1.23454787e+13)
```

In [6]:
```
data = np.genfromtxt("shipment_data.csv", delimiter=',')
print("Loaded Data from file.csv:\n", data)
```

```
Loaded Data from file.csv:
 [[  nan   nan   nan   nan   nan]
 [101.    nan   nan 120.5   nan]
 [102.    nan   nan  90.3   nan]
 [103.    nan   nan 150.    nan]
 [104.    nan   nan 130.7   nan]
 [105.    nan   nan  95.8   nan]]
```

Sorting Arrays

In [76]:

```python
h = np.array([6,4,2,7,8,1,9])
print(h)
```

```
[6 4 2 7 8 1 9]
```

In [75]:

```python
sor = h.sort()
print(sor)
```

```
None
```

In [80]:

```python
twoD = np.array([[6,4,2,7],[2,3,4,5]])
print(twoD)
```

```
[[6 4 2 7]
 [2 3 4 5]]
```

In [82]:

```python
#Sorting along the first axis ofthe 2D array
sor2D = np.sort(twoD, axis = 0)
print(sor2D)
```

```
[[2 3 2 5]
 [6 4 4 7]]
```

Appending in 1D array

In [84]:

```python
print(a)
arr = np.append(a, [7])
print("Array after appending:", arr)
```

```
[1 2 3 4 5]
Array after appending: [1 2 3 4 5 7]
```

In [9]:
```
#Adding the values at the end of a numpy array
arr = np.arange(1, 13).reshape(2, 6)
print("Original Array")
print(arr, "\n")
col = np.arange(5, 11).reshape(1, 6)
arr_col = np.append(arr, col, axis=0)
print("Array after appending the values column wise")
print(arr_col, "\n")
```

```
Original Array
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]

Array after appending the values column wise
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]
 [ 5  6  7  8  9 10]]
```

In [10]:
```
row = np.array([1, 2]).reshape(2, 1)
arr_row = np.append(arr, row, axis=1)
print("Array after appending the values row wise")
print(arr_row)
```

```
Array after appending the values row wise
[[ 1  2  3  4  5  6  1]
 [ 7  8  9 10 11 12  2]]
```

Statistics

In [11]:
```
#numpy.delete()
print("Original arr:", arr)
print("Shape : ", arr.shape)
```

```
Original arr: [[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
Shape :  (2, 6)
```

In [12]:
```
arr1 = [20, 2, 7, 1, 34]
```

In [13]:
```
print("mean of arr:", np.mean(arr))
```

```
mean of arr: 6.5
```

In [14]:

```
 print("median of arr:", np.median(arr))
```

```
median of arr: 6.5
```

In [15]:

```
print("Sum of arr(uint8):", np.sum(arr, dtype = np.uint8))
print("Sum of arr(float32):",np.sum(arr, dtype = np.float32))
```

```
Sum of arr(uint8): 78
Sum of arr(float32): 78.0
```

In [16]:

```
 print("maximum element:", np.max(arr))
 print("minimum element:", np.min(arr))
```

```
maximum element: 12
minimum element: 1
```

In [17]:

```
# Variation
print("var of arr:", np.var(arr))
print("var of arr(float32):",np.var(arr, dtype = np.float32))
```

```
var of arr: 11.916666666666666
var of arr(float32): 11.916667
```

In [19]:

```
# Standard deviation
print("std of arr:", np.std(arr))
print ("More precision with float32",np.std(arr, dtype = np.float32))
```

```
std of arr: 3.452052529534663
More precision with float32 3.4520526
```

Vector Math

In [20]:

```
 arr = np.array([.5, 1.5, 2.5, 3.5, 4.5, 10.1])
```

In [21]:

```
# numpy.delete()
print("Original arr:", arr)
print("Shape : ", arr.shape)
```

```
Original arr: [ 0.5  1.5  2.5  3.5  4.5 10.1]
Shape :  (6,)
```

In [22]:

```
# applying sqrt() method
print("Square-root:", np.sqrt(arr))
```

```
Square-root: [0.70710678 1.22474487 1.58113883 1.87082869 2.12132034 3.17804972]
```

In [23]:

```
# applying log() method
print("Log Value: ", np.log(arr))
```

```
Log Value:  [-0.69314718  0.40546511  0.91629073  1.25276297  1.5040774   2.31253542]
```

In [24]:

```
# applying absolute() method
print("Absolute Value:", np.absolute(arr))
```

```
Absolute Value: [ 0.5  1.5  2.5  3.5  4.5 10.1]
```

In [26]:

```
# applying sin() method
print("Sine values:", np.sin(arr))
```

```
Sine values: [ 0.47942554  0.99749499  0.59847214 -0.35078323 -0.97753012 -0.62507065]
```

In [27]:

```
# applying ceil() method
print("Ceil values:", np.ceil(arr))
```

```
Ceil values: [ 1.  2.  3.  4.  5. 11.]
```

In [28]:

```
# applying floor() method
print("Floor Values:", np.floor(arr))
```

```
Floor Values: [ 0.  1.  2.  3.  4. 10.]
```

In [29]:

```
# applying round_() method
print ("Rounded values:", np.round_(arr))
```

```
Rounded values: [ 0.  2.  2.  4.  4. 10.]
```

Coorelation coefficient

In [30]:

```
# create numpy 1d-array
array1 = np.array([0, 1, 2])
array2 = np.array([3, 4, 5])
```

In [33]:

```
rslt = np.corrcoef(array1, array2)
print(rslt)
```

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

Comparison

In [35]:

```
an_array = np.array([[1, 2], [3, 4]])
another_array = np.array([[1, 2], [3, 4]])
```

In [36]:

```
comparison = an_array == another_array
equal_arrays = comparison.all()
print(equal_arrays)
```

```
True
```

Arthimetic Operations

In [37]:

```
a = np.array([2,3,4,5])
b = np.array([6,7,8,9])
```

In [38]:

```
# Addition
np.add(a, b)
```

Out[38]:

```
array([ 8, 10, 12, 14])
```

In [39]:

```
# Subtraction
np.subtract(a, b)
```

Out[39]:

```
array([-4, -4, -4, -4])
```

In [40]:

```
# Multiply
np.multiply(a, b)
```

Out[40]:

```
array([12, 21, 32, 45])
```

In [41]:

```
# Division
np.divide(a, b)
```

Out[41]:

```
array([0.33333333, 0.42857143, 0.5       , 0.55555556])
```

```
In [42]: # Modulus
         np.mod(a, b)

Out[42]: array([2, 3, 4, 5])
```

```
In [44]: # Remainder
         np.remainder(a, b)

Out[44]: array([2, 3, 4, 5])
```

```
In [46]: # Power
         np.power(a, b)

Out[46]: array([     64,    2187,   65536, 1953125])
```

```
In [48]: # Exponent
         c = b.astype('float64')
         np.exp(a, c)

Out[48]: array([  7.3890561 ,  20.08553692,  54.59815003, 148.4131591 ])
```

Inserting Elements into the Array

```
In [50]: arr = np.asarray([1, 2, 3, 4])
```

```
In [51]: # Python Program illustrating numpy.insert()
         print("1D arr:", arr)
         print("Shape:", arr.shape)

        1D arr: [1 2 3 4]
        Shape: (4,)
```

```
In [52]:  a = np.insert(arr, 1, 9)
          print("\nArray after insertion:", a)
          print("Shape:", a.shape)

        Array after insertion: [1 9 2 3 4]
        Shape: (5,)
```

Removing Elements from Numpy Array

In [55]:
```
#numpy.delete()
object = 2
a = np.delete(arr, object)
print("\ndeleteing the value at index {} from array:\n {}".format(object,a))
print("Shape : ", a.shape)
```

```
deleteing the value at index 2 from array:
 [1 2 4]
Shape :  (3,)
```

Reshaping Array

In [57]:
```
#creating a numpy array
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9,10, 11, 12, 13, 14, 15, 16])
```

In [58]:
```
# printing array
print("Array: " + str(array))
```

```
Array: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

In [59]:
```
#converting it to 2-D from 1-D array
reshaped1 = array.reshape((4, array.size//4))
```

In [60]:
```
 #printing reshaped array
print("First Reshaped Array:")
print(reshaped1)
```

```
First Reshaped Array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

In [61]:
```
#creating another reshaped array
reshaped2 = np.reshape(array, (2, 8))
```

In [62]:
```
 # printing reshaped array
print("\nSecond Reshaped Array:")
print(reshaped2)
```

```
Second Reshaped Array:
[[ 1  2  3  4  5  6  7  8]
 [ 9 10 11 12 13 14 15 16]]
```

Resizing an Array Numpy arrays can be resized using the resize() function. It returns nothing but changes the original array.

In [4]:

```
 arr = np.array([1, 2, 3, 4, 5, 6])
```

In [64]:

```
 # Required values 12, existing values 6
arr.resize(3, 4)
print(arr)
```

```
[[1 2 3 4]
 [5 6 0 0]
 [0 0 0 0]]
```

Flatten a Two Dimensional array

In [65]:

```
list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
arr = np.array([list_1, list_2])
print(arr.flatten())
```

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

Transpose

In [66]:

```
gfg = np.array([[1, 2],[4, 5],[7, 8]])
 # before transpose
print(gfg, end ='\n\n')
 # after transpose
print(gfg.transpose(1, 0))
```

```
[[1 2]
 [4 5]
 [7 8]]

[[1 4 7]
 [2 5 8]]
```

Combining and Splitting Commands

In [68]:

```
# Combining Array
np.concatenate((a, b),axis = 0)
```

Out[68]:

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

In [72]:

```
#splitting array
np.split(arr, 2, 1)
```

Out[72]:

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

In [74]:

```
#horizantal split
np.hsplit(arr, 2)
```

Out[74]:

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

In [75]:

```
#vertical split
np.vsplit(arr, 2)
```

Out[75]:

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

Subsetting Numpy Array

In [5]:

```
 # Index values can be negative.
print(arr)
print("Elements are:", arr[np.array([1, 3, -3])])
```

```
[1 2 3 4 5 6]
Elements are: [2 4 4]
```

Slicing Numpy Array

In [6]:

```
 #a[start:stop:step]
print("a[-2:7:1] = ",arr[-2:7:1])
print("a[1:] = ",arr[1:])
```

```
a[-2:7:1] =  [5 6]
a[1:] =  [2 3 4 5 6]
```

Indexing Numpy Array: Numpy array indexing is of two types: Integer indexing and Boolean indexing

In [7]:

```
# Integer Indexing
a = np.array([[1 ,2 ],[3 ,4 ],[5 ,6 ]])
print(a[[0 ,1 ,2 ],[0 ,0 ,1]])
```

```
[1 3 6]
```

In [9]:

```
 # Boolean Indexing
a = np.array([10, 40, 80, 50, 100])
print(a[a>50])
```

```
[ 80 100]
```

Copying and Viewing Array

In [11]:

```
# copying to a new memory space
new = arr.copy()
print(new)
```

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

In [12]:

```
# shallow copy
sh =  arr.view()
print(sh)
```

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

In [2]:

```
import numpy as np
e = np.eye(5)
# np.size(e)
print(e.size)
```

```
25
```

In [ ]: