

# Chapter 5 :



## Informatics

## Practices

**Class XII (As per  
CBSE Board)**

**New  
Syllabus  
2019-20**

An illustration of a laptop computer with a white body and a black keyboard. The screen is tilted back and displays the text "Numpy - Array" in a bold, red, sans-serif font. The background of the screen is a light orange color. The laptop is set against a background of binary code (0s and 1s) in a light orange color.

**Numpy -  
Array**

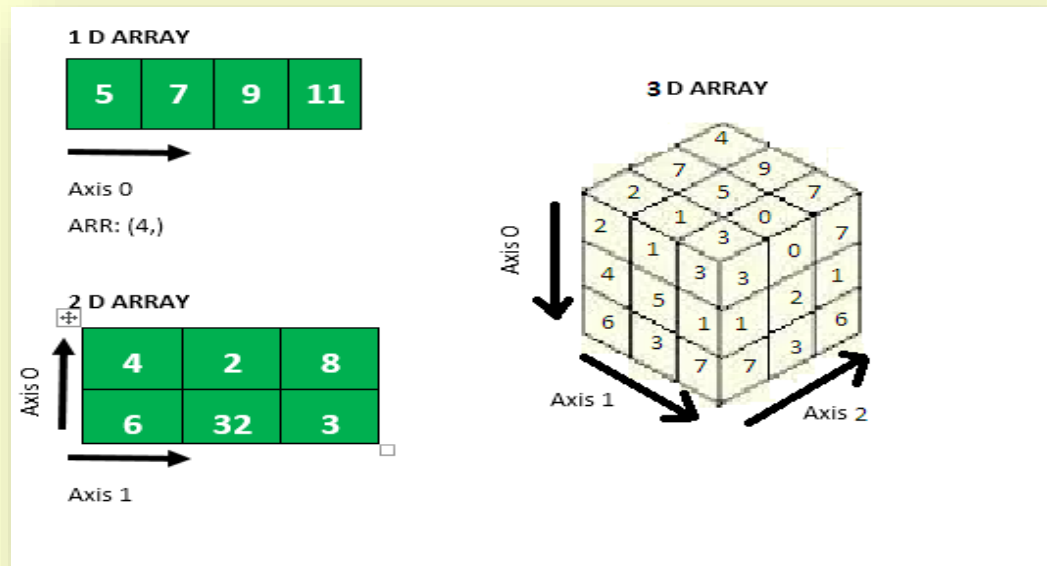
**Visit : [python.mykvs.in](http://python.mykvs.in) for regular updates**

# NUMPY - ARRAY

**NumPy** stands for Numerical Python. It is the core library for scientific computing in Python. It consists of multidimensional array objects, and tools for working with these arrays.

## Arrays

**Numpy Array** is a grid of values with same type, and is indexed by a tuple of nonnegative integers. The number of dimensions of it, is the rank of the array; the shape of an array depends upon a tuple of integers giving the size of the array along each dimension.



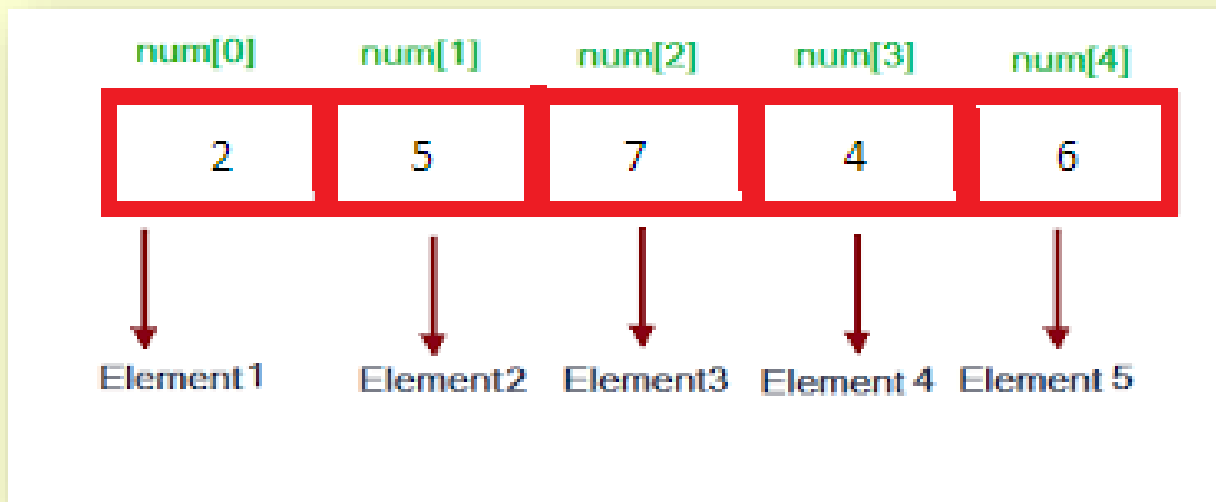
**Note:-** Before numpy based programming, it must be installed. It can be installed using `>pip install numpy` command at command prompt

# NUMPY - ARRAY

---

## 1 D ARRAY

Any arrays can be single or multidimensional. The number of subscript/index determines dimensions of the array. An array of one dimension is known as a one-dimensional array or 1-D array



In above diagram num is an array ,it's first element is at 0 index position ,next element is at 1 and so on till last element at n-1 index position. At 0 index position value is 2 and at 1 index position value is 5.

# NUMPY - ARRAY

---

## 1 D ARRAY

### Creation of 1D array

One dimension array can be created using array method with list object with one dimensional elements.

### e.g.program

```
import numpy as np
a = np.array([500, 200, 300])
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])
a[0] = 150
print(a)
```

# Create a 1D Array  
# Prints "<class 'numpy.ndarray'>"  
# Prints "(3,)" means dimension of array  
# Prints "500 200 300"  
# Change an element of the array

# NUMPY - ARRAY

---

## 1 D ARRAY

Creation of 1D array Using functions

```
import numpy as np
```

```
p = np.empty(5)    # Create an array of 5 elements with random values  
print(p)
```

```
a1 = np.zeros(5)   # Create an array of all zeros float values  
print(a1)         # Prints "[0. 0. 0. 0. 0.]"
```

```
a2 = np.zeros(5, dtype = np.int) # Create an array of all zeros int values  
print(a2)         # Prints "[0. 0. 0. 0. 0.]"
```

```
b = np.ones(5)     # Create an array of all ones  
print(b)          # Prints "[1. 1. 1. 1. 1.]"
```

```
c = np.full(5, 7)  # Create a constant array  
print(c)          # Prints "[7 7 7 7 7]"
```

```
e = np.random.random(5) # Create an array filled with random values  
print(e)
```

# NUMPY - ARRAY

---

## 1 D ARRAY

### Difference between Numpy array and list

NUMPY ARRAY	LIST
Numpy Array works on homogeneous types	Python list are made for heterogeneous types
Python list support adding and removing of elements	numpy.Array does not support adding and removing of elements
Can't contain elements of different types	can contain elements of different types
smaller memory consumption	more memory consumption
better runtime	Runtime not speedy

# NUMPY - ARRAY

---

## 1 D ARRAY

Create 1D from string

```
import numpy as np
data = np.fromstring('1 2', dtype=int, sep=' ')
print(data)
```

Note:- in fromstring dtype and sep argument can be changed.

Create 1D from buffer

numpy array from range

numpy.arange(start, stop, step, dtype)

#program 1

```
import numpy as np
```

```
x = np.arange(5) #for float value specify dtype = float as argument
```

```
print(x) #print [0 1 2 3 4]
```

#program 2

```
import numpy as np
```

```
x = np.arange(10,20,2)
```

```
print (x) #print [10 12 14 16 18]
```

# NUMPY - ARRAY

---

## 1 D ARRAY

Create 1D from array

Copy function is used to create the copy of the existing array.

e.g.program

```
import numpy as np
x = np.array([1, 2, 3])
y = x
z = np.copy(x)
x[0] = 10
print(x)
print(y)
print(z)
```

Note that, when we modify x, y changes, but not z:



# NUMPY - ARRAY

---

## 1 D ARRAY SLICES

Slicing of numpy array elements is just similar to slicing of list elements.

**e.g.program**

```
import numpy as np
data = np.array([5,2,7,3,9])
print (data[:])           #print [5 2 7 3 9]
print(data[1:3])         #print [2 7]
print(data[:2])          #print [5 2]
print(data[-2:])         #print [3 9]
```

# NUMPY - ARRAY

---

## 1 D ARRAY JOINING

Joining of two or more one dimensional array is possible with the help of concatenate() function of numpy object.

**e.g.program**

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([5, 6])
c=np.concatenate([a,b,a])
print(c) #print [1 2 3 5 6 1 2 3]
```

# NUMPY - ARRAY

---

Print all subsets of a 1D Array

If A {1, 3, 5}, then all the possible/proper subsets of A are { }, {1}, {3}, {5}, {1, 3}, {3, 5}

e.g.program

```
import pandas as pd
import numpy as np
def sub_lists(list1):
    # store all the sublists
    sublist = [[]]
    # first loop
    for i in range(len(list1) + 1):
        # second loop
        for j in range(i + 1, len(list1) + 1):
            # slice the subarray
            sub = list1[i:j]
            sublist.append(sub)
    return sublist
x = np.array([1, 2, 3,4])
# driver code
print(sub_lists(x))
```

**OUTPUT**

```
[[], array([1]), array([1, 2]),
array([1, 2, 3]), array([1, 2, 3, 4]),
array([2]), array([2, 3]), array([2, 3, 4]),
array([3]), array([3, 4]), array([4])]
```

# NUMPY - ARRAY

---

Basic arithmetic operation on  
1D Array  
e.g.program

```
import numpy as np
x = np.array([1, 2, 3,4])
y = np.array([1, 2, 3,4])
z=x+y
print(z) #print [2 4 6 8]
z=x-y
print(z) #print [0 0 0 0]
z=x*y
print(z) #print [ 1  4  9 16]
z=x/y
print(z) #print [1.  1.  1.  1.]
z=x+1
print(z) #print [2 3 4 5]
```

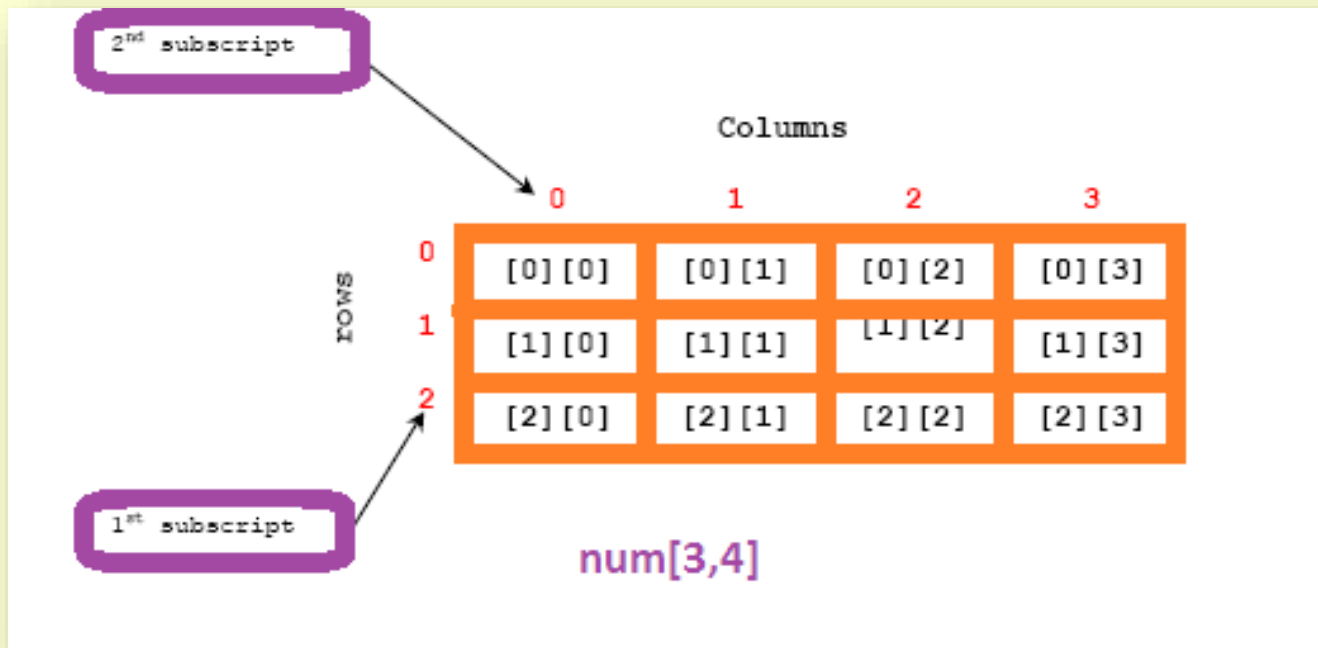
Aggregate operation on 1D  
Array  
e.g.program

```
import numpy as np
x = np.array([1, 2, 3,4])
print(x.sum()) #print 10
print(x.min()) #print 1
print(x.max()) #print 4
print(x.mean())#print 2.5
print(np.median(x))#print 2.5
```

# NUMPY - ARRAY

## 2 D ARRAY

An array of one dimension/index/subscript is known as a one-dimensional array or 1-D array



In above diagram num is an array of two dimension with 3 rows and 4 columns. subscript of rows is 0 to 2 and columns is 0 to 3.

# NUMPY - ARRAY

---

## 2 D ARRAY

### Creation of 2D array

Two dimension array can be created using array method with list object with two dimensional elements.

### e.g.program

```
import numpy as np
a = np.array([[3, 2, 1],[1, 2, 3]]) # Create a 2D Array
print(type(a)) # Prints "<class 'numpy.ndarray'>"
print(a.shape) # Prints (2, 3)
print(a[0][1]) # Prints 2
a[0][1] = 150 # Change an element of the array
print(a) # prints [[ 3 150  1] [ 1  2  3]]
```

# NUMPY - ARRAY

---

## 2 D ARRAY

Creation of 2D array Using functions

```
import numpy as np
```

```
p = np.empty([2,2])    # Create an array of 4 elements with random values  
print(p)
```

```
a1 = np.zeros([2,2])  # Create 2d array of all zeros float values  
print(a1)            # Prints [[0. 0.][0. 0.]
```

```
a2 = np.zeros([2,2], dtype = np.int)  # Create an array of all zeros int values  
print(a2)                              # Prints [[0 0] [0 0]]
```

```
b = np.ones([2,2])    # Create an array of all ones  
print(b)              # Prints [[1. 1.] [1. 1.]
```

```
c = np.full([2,2], 7) # Create a constant array  
print(c)              # Prints [[7 7] [7 7]]
```

```
e = np.random.random([2,2])    # Create 2d array filled with random values  
print(e)
```

# NUMPY - ARRAY

---

## 2D ARRAY

Creation of 2D array from 1D array

We can create 2D array from 1d array using reshape() function.

e.g. program

```
import numpy as np
A = np.array([1,2,3,4,5,6])
B = np.reshape(A, (2, 3))
print(B)
```

OUTPUT

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



# NUMPY - ARRAY

---

## 2 D ARRAY SLICES

Slicing of numpy 2d array elements is just similar to slicing of list elements with 2 dimension.

**e.g.program**

```
import numpy as np
A = np.array([[7, 5, 9, 4],
              [ 7, 6, 8, 8],
              [ 1, 6, 7, 7]])
print(A[:2, :3])      #print elements of 0,1 rows and 0,1,2 columns
print(A[:3, ::2])     #print elements of 0,1,2 rows and alternate column
                      #position
print(A[::-1, ::-1]) #print elements in reverse order
print(A[:, 0])        #print all elements of 0 column
print(A[0, :])        #print all elements of 0 rows
print(A[0])           #print all elements of 0 row
```

# NUMPY - ARRAY

---

## 2 D ARRAY JOINING

### e.g.program

```
import numpy as np
A = np.array([[7, 5],
              [1, 6]])
```

```
# concatenate along the first axis
```

```
print(np.concatenate([A, A]))
```

```
# concatenate along the second  
axis (zero-indexed)
```

```
print(np.concatenate([A, A], axis=1))
```

```
x = np.array([1, 2])
```

```
# vertically stack the arrays
```

```
print(np.vstack([x, A]))
```

```
# horizontally stack the arrays
```

```
y = np.array([[99],  
              [99]])
```

```
print(np.hstack([A, y]))
```

**OUTPUT**

→ **[[7 5]**

**[1 6]**

**[7 5]**

**[1 6]]**

→ **[[7 5 7 5]**

**[1 6 1 6]]**

→ **[[1 2]**

**[7 5]**

**[1 6]]**

→ **[[ 7 5 99]**

**[ 1 6 99]]**

# NUMPY - ARRAY

## 2 D ARRAY – ARITHMATIC OPERATION

Arithmetic operation over 2d array is possible with add,subtract,multiply,divide () functions.

### E.G.PROGRAM

```
import numpy as np
```

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

```
print(a)
```

OUTPUT

```
[[7 5 9]  
 [2 6 8]]
```

```
b = np.array([10,10,10])
```

```
c=np.add(a,b) # c=a+b, similar
```

```
print(c)
```

```
[[17 15 19]  
 [12 16 18]]
```

```
c=np.subtract(a,b) # c=a-b, similar
```

```
print(c)
```

```
[[ -3 -5 -1]  
 [-8 -4 -2]]
```

```
c=np.multiply(a,b) # c=a*b, similar
```

```
print(c)
```

```
[[70 50 90]  
 [20 60 80]]
```

```
c=np.divide(a,b) # c=a/b, similar
```

```
print(c)
```

```
[[0.7 0.5 0.9]  
 [0.2 0.6 0.8]]
```

**Note:-**

1. if both 2d arrays are with same dimension[matrix form] then one to one arithmetic operation will be performed.
2. No of elements of a dimension must match otherwise error message thrown

# NUMPY - ARRAY

## 2 D ARRAY – ARITHMATIC OPERATION

Arithmetic operation over 2d array can be done with single value also.

### E.G.PROGRAM

```
import numpy as np  
a = np.array([[7, 5, 9],  
             [ 2, 6, 8]])
```

```
print(a)
```

OUTPUT

```
[[7 5 9]  
 [2 6 8]]
```

```
c=np.add(a,2)
```

```
print(c)
```

```
[[ 9  7 11]  
 [ 4  8 10]]
```

```
c=np.subtract(a,2)
```

```
print(c)
```

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

```
c=np.multiply(a,2)
```

```
print(c)
```

```
[[14 10 18]  
 [ 4 12 16]]
```

```
c=np.divide(a,2)
```

```
print(c)
```

```
[[3.5 2.5 4.5]  
 [1.  3.  4.  ]]
```

# NUMPY - ARRAY

---

## 2 D ARRAY – Mathematical Functions

Maths functions like power,abs,ceil,floor,around and trigonometric functions like sin,cos,tan,asin etc are supported by numpy

### E.G.PROGRAM

```
import numpy as np
```

```
a = np.array([[7.333, 5.223],  
             [ 2.572, 6.119]])
```

```
print(np.power(a,2))
```

**OUTPUT**

```
[[53.772889 27.279729]  
 [ 6.615184 37.442161]]
```

```
print(np.ceil(a))
```

```
[[8. 6.]  
 [3. 7.]]
```

```
print(np.floor(a))
```

```
[[7. 5.]  
 [2. 6.]]
```

```
print(np.around(a,1))
```

```
[[7.3 5.2]  
 [2.6 6.1]]
```

# NUMPY - ARRAY

---

## 2 D ARRAY

### VARIANCE

The average of the **squared** differences from the Mean.

#### STEPS

1. find the mean of values
2. Subtract value with mean value then square the result, sum all results of each value
3. Find the average of sum value

#### e.g. program

```
import numpy as np
b = np.array([600,470,170,430,300])
print(b.mean()) # print 394.0
print(np.var(b,ddof=0)) # print 21704.0
```

*NOTE :- Variance is calculated like  $(206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2)/5$   
206=600-394 and so on for other values*

# NUMPY - ARRAY

## 2 D ARRAY COVARIANCE

**Covariance** is a statistical measure that shows whether two variables are related by measuring how the variables change in relation to each other.

### FORMULA

$$\text{Cov}(X, Y) = \frac{\sum(X_i - \bar{X})(Y_j - \bar{Y})}{n - 1}$$

### Example

YEAR	ITEM1 PRICE	ITEM 2 PRICE	variance A	variance B	AxB
2015	1000	130	-266.667	10	-2666.67
2016	1200	110	-66.6667	-10	666.6667
2017	1600	120	333.3333	10	3333.333
	1266.666667	120			1333.333
				covariance=	666.6667

*In above calculation value of A is calculated by subtracting item1 price from item1 mean price so on for B and all values. Covariance by dividing sum of AXB by n-1 (number of years)*

# NUMPY - ARRAY

---

## 2 D ARRAY

## COVARIANCE

e.g. program

```
import numpy as np  
a = np.array([1000, 1200, 1600])  
b = np.array([130, 110, 120])  
  
print(np.cov(a,b,bias=True)[0, 1])
```

## OUTPUT

-666.6666666666666



# NUMPY - ARRAY

---

## 2 D ARRAY

### CORRELATION

Correlation is the scaled measure of covariance. Besides, it is dimensionless. In other words, the correlation coefficient is always a pure value and not measured in any units.

#### Formula

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

$\text{Cov}(X, Y)$  – the covariance between the variables X and Y

$\sigma_X$  – the standard deviation of the X-variable

$\sigma_Y$  – the standard deviation of the Y-variable

# NUMPY - ARRAY

---

## 2 D ARRAY

## CORRELATION

e.g.program

```
import numpy as np
a = np.array([1000,1200,1600])
b = np.array([130,110,120])
print(np.corrcoef(a, b))
```

## OUTPUT

```
[[ 1.         -0.32732684]
 [-0.32732684  1.         ]]
```

# NUMPY - ARRAY

---

## 2 D ARRAY

### Linear Regression

Linear regression is a method used to find a relationship between a dependent variable and independent variable(s).

#### Types

- 1. Simple Linear Regression:** There is only one independent variable in it. E.g. the price of the house depend only one field that is the size of the plot.
- 2. Multiple Linear Regression:** There is more independent variable in it. E.g. the price of the house depend one field that is the size of the plot and number of rooms.

**Linear Equation:**  $Y=aX + b$

**a:** Slope of the line

**b:** Constant (Y-intercept, where  $X=0$ )

**X:** Independent variable

**Y:** Dependent variable

# NUMPY - ARRAY

## 2 D ARRAY

### Linear Regression

#### E.g. program

```
import numpy as np
import matplotlib.pyplot as plt
def estcoefficients(x,y):
    n=np.size(x)
    meanx, meany = np.mean(x), np.mean(y)
    sy = np.sum(y*x - n*meany*meanx)
    sx = np.sum(x*x - n*meanx*meanx)
    a=sx/sy
    b=meany-a*meanx
    return(a,b)
def plotregline(x,y,b):
    plt.scatter(x,y,color="r",marker="o",s=30)
    ypred=b[0]+b[1]*x
    plt.plot(x,ypred,color="g")
    plt.xlabel('SIZE')
    plt.ylabel('COST')
    plt.show()
x=np.array([10,20,30,40,50]) # independent variable
y=np.array([400,800,1100,1700,2100]) # dependent variable
b=estcoefficients(x,y)
plotregline(x,y,b)
```

