BISHOP SCOTT BOYS' SCHOOL
(Affiliated to CBSE, New Delhi) Affiliation No.: 330726, School Campus: Chainpur, Jaganpura, By-Pass, Patna 804453.
Phone Number: 7061717782, 9798903550. ,
Web: www.bishopscottboysschool.com Email: info@bishopscottboysschool.com

STUDY COURSE MATERIAL

IP

SESSION-2020-21

CLASS-XII

TOPIC: MS ACCESS 2010

DAY-1

❖ TEACING MATERIAL

NUMPY - ARRAY

NumPy stands for Numerical Python.It is the core library for scientific computing in Python. It consist of multidimensional array objects, and tools for working with these 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:- Befor numpy based programming ,it must be installed. It can be installed using >pip install numpy command at command prompt

.

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])          # Create a 1D Array
print(type(a))                         # Prints "<class 'numpy.ndarray'>"
print(a.shape)                         # Prints "(3,)" means dimension of array
print(a[0], a[1], a[2])                # Prints "500 200 300"
a[0] = 150                             # Change an element of the array
print(a)
```

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

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]
```

❖ VIDEO-LINKS

LINK-1

https://www.youtube.com/watch?v=i87tYJz6T5g&list=PLw1djS7l2tHOBNU4PmZCdQWTQc0XR8reI

❖ PPT LINKS

LINK-1

https://www.slideshare.net/PyData/introduction-to-numpy

❖ DOCUMENTS LINKS

https://drive.google.com/drive/u/1/folders/1omfOhFknN8JqQEjV_EU5vw9YTOii6d6g

DAY-2

❖ TEACING MATERIAL

```
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)
```

❖ VIDEO-LINKS

https://www.youtube.com/watch?v=i87tYJz6T5g&list=PLw1djS7l2tHOBNU4PmZCdQWTQc0XR8reI

DAY-3

❖ VIDEO-LINKS

MUST WATCH
https://www.youtube.com/watch?v=i87tYJz6T5g&list=PLw1djS7l2tHOBNU4PmZCdQWTQc0XR8reI

❖ PPT LINKS

https://www.slideshare.net/okmomwalking/access-2010-unit-a-ppt-15608211

DAY-4

❖ DOCUMENTS LINKS

https://drive.google.com/drive/u/1/folders/1omfOhFknN8JqQEjV_EU5vw9YTOii6d6g

❖ VIDEO-LINKS

https://www.youtube.com/watch?v=i87tYJz6T5g&list=PLw1djS7l2tHOBNU4PmZCdQWTQc0XR8reI

DAY-5

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
print(x.max()) #print 4
print(x.mean())#print 2.5
print(np.median(x))#print 2.5
```

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

❖ DOCUMENTS LINKS

Go through the course book.

❖ VIDEO-LINKS

https://www.youtube.com/watch?v=i87tYJz6T5g&list=PLw1djS7l2tHOBNU4PmZCdQWTQc0XR8reI

EXERCISE:

Practice this all in your system

```
import numpy as np
a = np.array([1,2,3,4])
print(a.shape) # Gives you the dimension of the array- (4 rows for 1 D array)
```

```
(4,)
```

In [2]:
```
import numpy as np
b = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(b.shape)
print(b)
print(b[0]) # Prints the contents of first row as [1,2,3]
print(b[0][0]) #Prints the contents of first row and first Column i.e. 1
print(b[1][0]) # Prints the contents of second row and first Column i.e. 4
print(b[2][0]) # Prints the contents of third row and first Column i.e. 7
print(b[2,2]) # or Prints the contents of third row and third Column i.e. 7
```

```
(3, 3)
[[1 2 3]
 [456]
 [7 8 9]]
[123]
1
4
7
9
```

In [21]:
```
import numpy as np
b = np.zeros((3,3)) # Creates a 3x3 matrix with
zeros print(b)
```

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

In [22]:
```
import numpy as np
b = np.ones((3,3)) # Creates a 3x3 matrix with ones
print(b)
```

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

In [26]:
```
import numpy as np
b = np.full((2,2), 8) # Creates a 2x2 constant matrix with
8 print(b)
```

```
[[8 8]
 [8 8]]
```

In [28]:
```
import numpy as np
b = np.eye(3) # Creates an identity matrix
print(b)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [29]:
```
import numpy as np
b = np.random.random((2,2)) # Creates an random number
matrix print(b)
```

```
[[0.0133473  0.97649951]
 [0.61791382 0.44710574]]
```

In [35]:
```
#Using arange function to create a numpy
array import numpy as np
b = np.arange(10) # creates an array of integers from 0 to
9 print(b)
a = np.arange(1,11, dtype=float) # creates an array of floats from 1 to
10 print(a)
c = np.arange(2,3, 0.1)
print(c) # Creates an array with interval of 0.1
```

```
[0123456789]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9]
```

In [78]:
```
#Slicing of numpy arrays
#Resultant array will always be a sub-array of the original
array import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b = a[0:2] # slice consists of only first two row (excludes row three)
print('b',b)
b = a[1:2] # Slice consists of only second row
print('b',b)
b = a[0:1,0:4] # Slice consists of entire first row (first row all columns)
print('b',b)
b = a[0:1,0:2] # Slice consists of first row and first two
columns print('b',b)
b = a[0:1,2:4] # Slice consists of first row and columns three and
four print('b',b)
```

```
b [[1 2 3 4]
 [5678]]
b [[5 6 7 8]]
b [[1 2 3 4]]
b [[1 2]]
b [[3 4]]
```

In [70]:
```
#Slicing of numpy arrays
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b = a[1:2,0:4] # Slice consists of second row all
columns print('b',b)
b = a[1:2,1:2] # Slice consists of second row second
column print('b',b)
b = a[:2,0:2] # Slice consists of two rows first[1,2] and
second[5,6] print('b',b)
b = a[:2,1:3] # Slice consists of two rows first[2,3] and
second[6,7] print('b',b)
b = a[1:3,2:4] # Slice consists of two rows first[7,8] and
second[11,12] print('b',b)
```

```
b [[5 6 7 8]]
b [[6]]
b [[1 2]
 [5 6]]
b [[2 3]
 [6 7]]
b [[7 8]
 [11 12]]
```

In [74]:
```
#Indexing will create a new arbitry array from the original array
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
print(a.shape)
b = a[[0,1,2],[0,1,0]] # The resultant array will have shape (3,)
print(b)
print(b.shape)
```

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

In [77]:
```
#Changing the values in an array resulting from slice
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b = a[0:2,1:3] #Sliced array
print(b)
b[0,0]=20 # Chaning th value in the slice
print(b)
print(a) # The change is getting reflected in the original array
```

```
 [[2 3]
  [6 7]]
[[20  3]
 [ 6  7]]
[[ 1  20  3  4]
 [ 5   6  7  8]
 [ 9  10 11 12]]
```

In [85]:
```
#Joining two numpy arrays
import numpy as np
a = np.array([[1,2],[2,4]])
b = np.array([[7,8]])
c = np.concatenate((a,b), axis=0)
print(c)
d = np.concatenate((a,b.T), axis=1)
print(d)
```

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

In [91]:
```
#Numpy arithmatic operations
import numpy as np
a = np.array([[1,2],[3,4]], dtype=np.int32)
b = np.array([[5,6],[7,8]], dtype=np.int32)
print(a+b)
print(np.add(a,b))
print(a-b)
print(np.subtract(a,b))
print(a*b) # Multiplies corresponding elements
print(np.multiply(a,b))
print(a/b)
print(np.divide(a,b))
print(np.sqrt(a))
# Actual Matrix multiplication
print(a.dot(b))
print(np.dot(a,b))
```

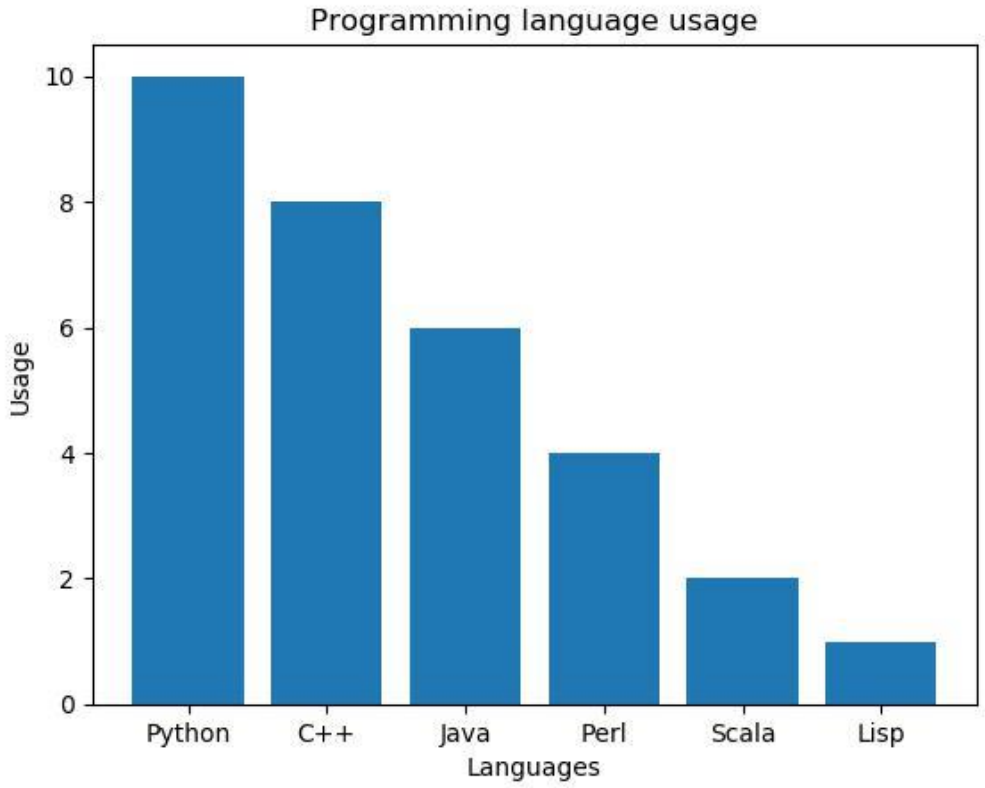
```
 [[ 6 8]
  [1012]]
 [[ 6 8]
  [1012]]
 [[-4-4]
  [-4-4]]
 [[-4-4]
  [-4-4]]
 [[ 512]
  [2132]]
 [[ 512]
  [2132]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
 [[1.        1.41421356]
 [1.73205081 2.        ]]
 [[1922]
  [4350]]
 [[1922]
  [4350]]
```

In [94]:
```
#numpy sum() function
a = np.array([[1,2],[3,4]], dtype=np.int32)
print(np.sum(a, axis=0))
print(np.sum(a, axis=1))
```

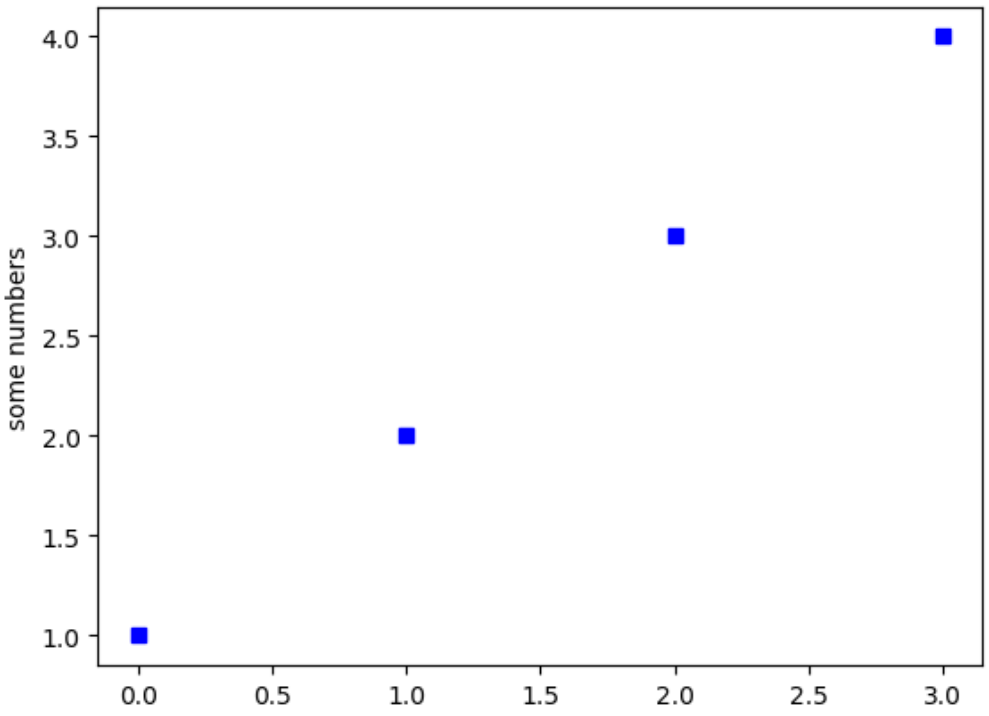
```
[4 6]
[3 7]
```

In [152]:
```
#Barchart using matplotlib
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
#import matplotlib.pyplot as plt

objects = ('Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp')
y_pos = np.arange(len(objects))
performance = [10,8,6,4,2,1]
height = performance
width = 0.8
bottom = 0
plt.bar(y_pos, height, width, bottom, align='center', alpha=1.0)
plt.xticks(y_pos,objects)
plt.ylabel('Usage')
plt.xlabel('Languages')
plt.title('Programming language usage')
```

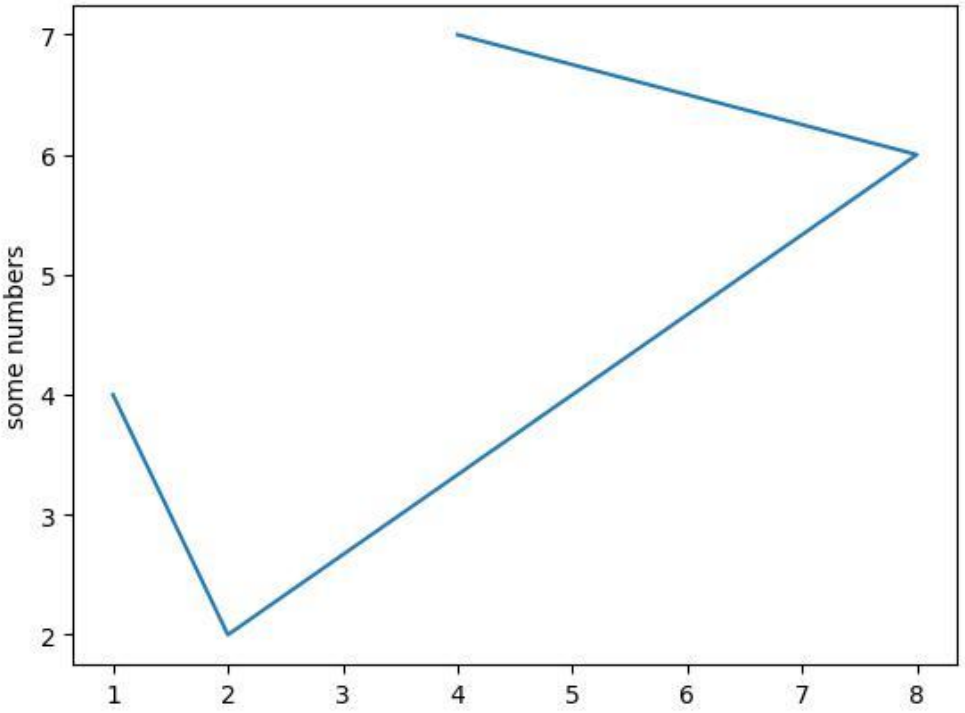
```
plt.show()
```



In [160]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4],'bs') # 'ro' for round 'g^' for triange and 'bs' for square. By
 default line

plt.ylabel('some numbers')
plt.show()
```



In [164]:
```
import matplotlib.pyplot as plt
plt.plot([1, 2, 8, 4],[4,2,6,7])
plt.ylabel('some numbers')
plt.show()
```



In [5]:
```
import matplotlib.pyplot as plt
t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
plt.plot(t, s, lw=3)
plt.show()
```