MODULE:3
DATA VISUALIZATION-THE MATPLOTLIB MODULE

The Matplotlib is a Python package that produces high quality figures in a variety of picture formats like png,emf,eps, pdf, ps,raw,svg,etc. It also provides many functions for matrix manupulation. We can generate plots, histograms, properties, axis properties,etc.The data points to the plottng function are supplied as Python lists or Numpy arrays.

If we import matplotlib module, then the plotting function from sub-module *pyplot* and matrix manupulation function from the sub-module *mlab will be* available.

Different ways of importing graphics package & modules:

1.Import matplotlib.pyplot as plt

2.from pylab import*

plotting simple graphs using matplotlib module

eg;1

```
import matplotlib.pyplot as plt

x=[1,2,3,4,5]

y=[6,7,8,9,10]

plt.xlabel('x')

plt.ylabel('y')

plt.title('simple graph')

plt.plot(x,y)

plt.show()
```

eg:2

```
import matplotlib.pyplot as plt

plt.xlabel('x')
plt.ylabel('y')
plt.title('simple graph')
plt.plot(range(10),range(10))
plt.show()
```

simple plots using pylab module:

```
#simple graph using pylab module
from pylab import*      # * indicates that all functions inside the pylab module has been enabled
x=[1,2,3,4,5]
y=[6,7,8,9,10]
xlabel('x')
ylabel('y')
title('simple graph')
plot(x,y)
show()
```

eg:4

```
simple graph using pylab module
from pylab import*
xlabel('x')
ylabel('y')
title('simple graph')
plot(range(10),range(10))
show()
```

eg:5

```
#simple graph using matplotlib module
import matplotlib.pyplot as plt
from numpy import*
x=linspace(0,2*pi,200)
plt.plot(x,sin(x))
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('sin curve')
plt.axhline(y=0)
plt.axvline(x=pi)
plt.show()
```

eg:6

```
#simple graph using pylab module
from pylab import*
x=linspace(0,2*pi,200)
plot(x,sin(x),'g')
xlabel('x')
ylabel('sin(x)')
title('sin curve')
axhline(y=0)
axvline(x=pi)
#axis([0,pi,0,1])
show()
```

eg:7

```
#simple graph using pylab module
from pylab import*
x=linspace(0,2*pi,200)
plot(x,cos(x))
xlabel('x')
ylabel('sin(x)')
title('cos curve')
axhline(y=0)
axvline(x=pi)
show()
```

eg:8

```
#simple graph using pylab module
from pylab import*
x=linspace(0,2*pi,200)
plot(x*180/pi,tan(x))
xlabel('x in degrees')
ylabel('tan(x)')
title('tan curve')
axhline(y=0)
axvline(x=pi*180/pi)
show()
```

eg:9

```
from pylab import*
#multiple curves
x=linspace(0,2*pi,200)
plot(degrees(x),sin(x),'g')
plot(degrees(x),cos(x),'r')
plot(degrees(x),tan(x),'c')
axhline(y=0)
axvline(x=pi*180/pi)
title('sin/cos/tan curve')
xlabel('x in degrees')
ylabel('sin/cos/tan ')
show()
```

Multiple plots:

Matplotlib/pylab allows us to have multiple plots in the same window, using the subplot() function as shown in the example given below:

eg;9

```
#subplots
import matplotlib.pyplot as plt
from numpy import*
x=linspace(0,2*pi,200)
subplot(3,3,1)
plot(degrees(x),sin(x),'g')
xlabel('x in degrees')
```

```
ylabel('sin(x)')
title('sin curve')
axhline(y=0)
axvline(x=pi*180/pi)
subplot(3,3,2)
plot(degrees(x),cos(x),'r')
xlabel('x in degrees')
ylabel('cos(x)')
title('cos curve')
axhline(y=0)
axvline(x=pi*180/pi)
subplot(3,3,3)
axhline(y=0)
axvline(x=pi*180/pi)
plot(degrees(x),tan(x),'c')
xlabel('x in degrees')
ylabel('tan(x)')
title('tan curve')
subplot(3,3,7)

plot(x,log(x),'g')
xlabel('x')
ylabel('log(x)')
title('log(x) curve')
subplot(3,3,8)
plot(x,exp(x),'r')
xlabel('x')
ylabel('exp(x)')
title('exp(x) curve')
subplot(3,3,9)
plot(x,exp(-x),'c')
xlabel('x ')
ylabel('exp(-x)')
title('exp(-x) curve')
show()
```

polar plots;

polar co-ordinates locate a point on a plane with one distance and one angle. The distance 'r' is measured from the origin. The angle θ is measured from some agreed starting point. Use the positive part of the x-axis as the starting point for measuring angles. Measure positive angles anti-clockwise from the positive x-axis and negative angles clockwise from it.

Matplotlib supports polar plots, using the polar (θ,r) function. Let us plot a circle using polar() function. For every point on the circle, the value of radius is the same but the polar angle θ changes from 0 to 2π. Both the co-ordinate arguments must be arrays of equal size. Since θ is having100 points (in the given example), r also must have the same number. This array can be generated using the ones() function.

```
#polar plot
from pylab import*
theta=linspace(0,2*pi,100)
r=5*ones(100)  #r=radius of the circle
polar(theta,r)
show()
```

Pie charts

An example of a pie chart is given below. The percentage of differnt items and their names are given as arguments. The o/p is shown in fig.

```
#pie chart,
from pylab import*
labels='WBC','RBC','PLATELETS','PLASMA'
fracs=[2,35,8,55]
pie(fracs,labels=labels)

show()
```

plot of sin,log,exp functions

```
from pylab import*

x=linspace(0,2*pi,200)
plot(x,sin(x),'m')
figure(1)
xlabel('x')
ylabel('sin(x)')
title('sin curve')
axhline(y=0)
axvline(x=pi)

figure(2)
plot(x,log(x),'g')
xlabel('x')
ylabel('log(x)')
title('log(x) curve')

figure(3)
plot(x,exp(x),'r')
xlabel('x')
ylabel('exp(x)')
title('exp(x) curve')

show()
```

Bessel function

```
#bessel
from scipy import*
from pylab import*
xx=[]
jnx=[]
n=input('enter the order of Bessel fn:eg:0,1,2,3....')
for x1 in range(0,200):
        x=x1/10.
        jn=0.
        for k in range(100):
                jn=jn+(((-1)**k)*((x/2.)**(n+2*k)))/(factorial(k)*factorial(n+k))
        xx.append(x)
        jnx.append(jn)
#       print '%2d\t%6.3f\t%13.6f'%(n,x,jn)
plot(xx,jnx)
axhline(y=0)
xlabel('Jn(x)')
ylabel('x')
title('BESSEL FUNCTION')
text(5,.3,'n=%2d'%(n))
show()
```

Legendre function

```
#legendre
from scipy import*
from pylab import*
xx=[]
pnx=[]
n=input('enter the order of Legendre fn:eg:0,1,2,3....')
if n%2==0:
        s=n/2
else:
        s=(n-1)/2
```

```
 for x1 in range(-100,100):
        x=x1/100.
        pn=0.
        for k in range(s+1):
            pn=pn+(((-1)**k)*(factorial(2*n-2*k))*((x)**(n-2*k)))/((2**n)*factorial(k)*factorial(n-k)*factorial(n-2*k))
        xx.append(x)
        pnx.append(pn)
        print '%2d\t%6.3f\t%13.6f'%(n,x,pn)
plot(xx,pnx)
axhline(y=0)
xlabel('pn(x)')
ylabel('x')
axhline(y=0)
axvline(x=0)
title('LEGENDRE FUNCTION')
text(0,0,'n=%2d'%(n))
show()
```

Gamma function:

```
from scipy.special import gamma as Gamma
from pylab  import*
def f(x):
   return Gamma(x)
x=linspace(-6,6,512)
y=f(x)
gca().set_autoscale_on(False)
plot (x,y,color='red')
xlabel('x')
ylabel('gamma')
axhline(y=0)
#axvline(x=0)
title('GAMMA FUNCTION')
axis([-6,6,-100,100])
```

grid(True)

show()

GAUSSIAN FUNCTION/DISTRIBUTION(NORMAL FUNCTION/DISTRIBUTION)

The probability density function of the normal distribution, first derived by De Morvie and 200 years later by both Gauss and Laplace independently is often called the “bell curve' because of its characteristic shape.

The normal distribution occures often in nature. For ex:, it describes the commonly occuring distributions of samples influenced by a large no. Of tiny random disturbances.

The probability density for the Gaussian distribution is $P(x)= 1/sqrt(2*pi*\sigma^2)*exp-(x-\mu)^2/2\ \sigma^2$ where μ is the mean, σ is the std.deviation , σ^2 is called the variance. The function has its peak at the mean and it's 'spread' increases with std.deviation. (The function reaches its

0.607 times its max.at $x+\sigma$ and $x-\sigma$)

```
import matplotlib.pyplot as plt

import numpy as np

mu, sigma = 0, 0.01# mean and standard deviation

s = np.random.normal(mu, sigma, 1000)



count, bins, ignored = plt.hist(s, 30, normed=True)

plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *

            np.exp( - (bins - mu)**2 / (2 * sigma**2) ),

        linewidth=2, color='r')

plt.show()
```

Parametric plots

The circle can be represented using the equations x=acosθ and y=asinθ. To get the complete circle, θ should var from zero to 2π radians. The following program illustrates this,.

```
#parametric plot
from pylab import*
r=10.
th=linspace(0,2*pi,200)
x=r*cos(th)
y=r*sin(th)
plot(x,y)
show()
```

Note: changing the range of θ to less than 2π radians will result in an arc.