python

UNIT – VI

Contents

- Working with NumPy/PlotPy/SciPy
- GUI Programming
  - Introduction
  - Tkinter programming
  - Tkinter widgets

Numpy

Numpy: NumPy stands for Numerical Python. NumPy is a Python library used for working with arrays.

Numpy arrays

Array are by default Homogeneous, which means data inside an array must be of the same Data type.

Element wise operation is possible.

Numpy array has the various function, methods, and variables, to ease our task of matrix computation.

Elements of an array are stored contiguously in memory.

Lists

The list can be homogeneous or heterogeneous.

Element wise operation is not possible on the list.

Python list is by default 1 dimensional. But we can create an N-Dimensional list. But then too it will be 1 D list storing another 1D list

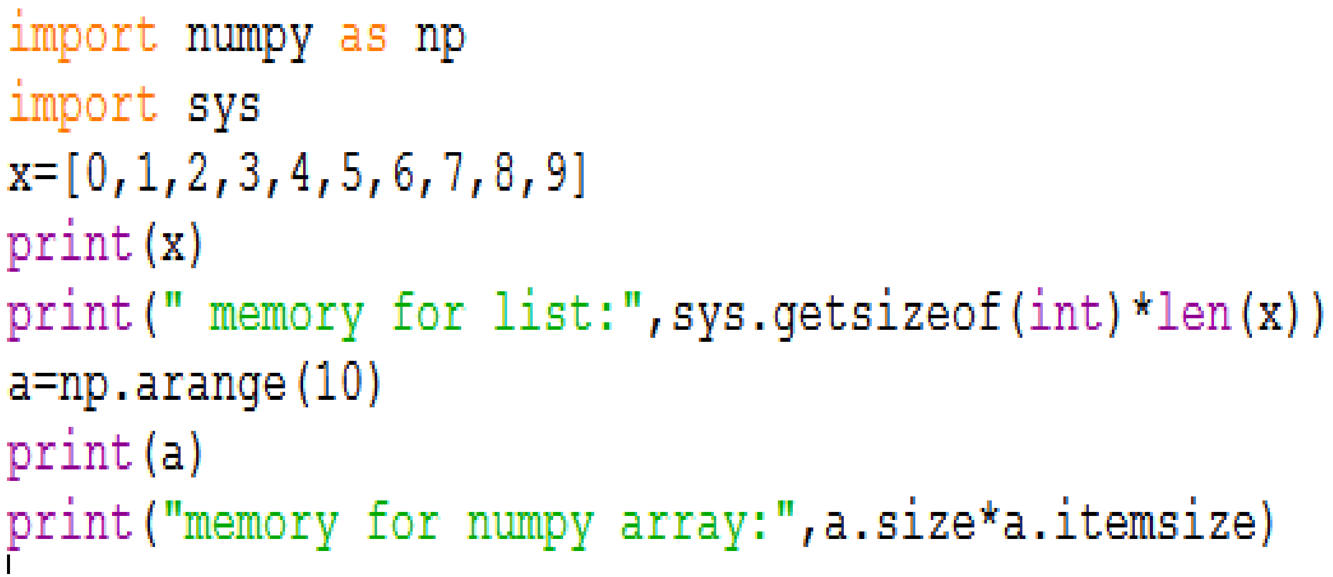
Elements of a list need not be contiguous in memory.

Advantages of using Numpy Arrays Over Python Lists:

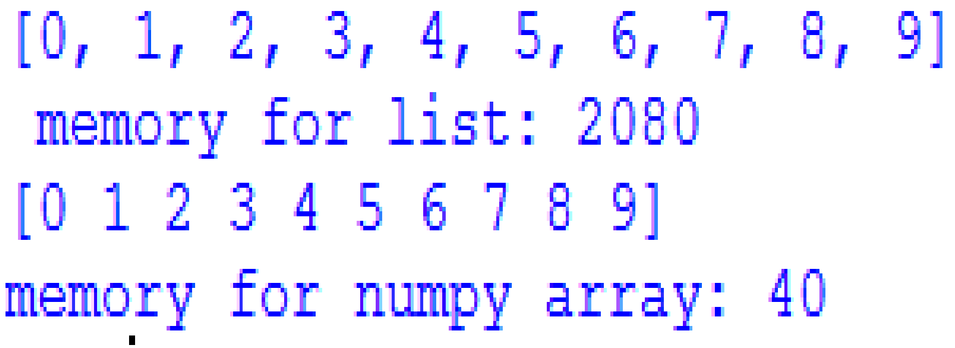
consumes less memory.

fast as compared to the python List.

convenient to use.



```
import numpy as np
import sys
x=[0,1,2,3,4,5,6,7,8,9]
print(x)
print(" memory for list:",sys.getsizeof(int)*len(x))
a=np.arange(10)
print(a)
print("memory for numpy array:",a.size*a.itemsize)
```



```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
 memory for list: 2080
[0 1 2 3 4 5 6 7 8 9]
memory for numpy array: 40
```

Numpy: We can create a N-dimensional array or ndarray in python using numpy.if n=1,it represents a 1-D.if n=2, it represents a 2-D

To work with numpy,we shoud first import numpy module into our python programs as

```
            import  numpy
```

Create arrays using numpy: Creating arrays in numpy can be done in several ways

1.array():used to create an array. When we create an array, we can specify the data type of the elements either as int or float

```
            array(lst,datatype)
```

2.linspace():used to create an array with evenly spaced points between a starting point and ending point

```
            linespace(start,stop,n)
```

3.logspace() used to create an array with evenly spaced points on a logarithmically spaced scale

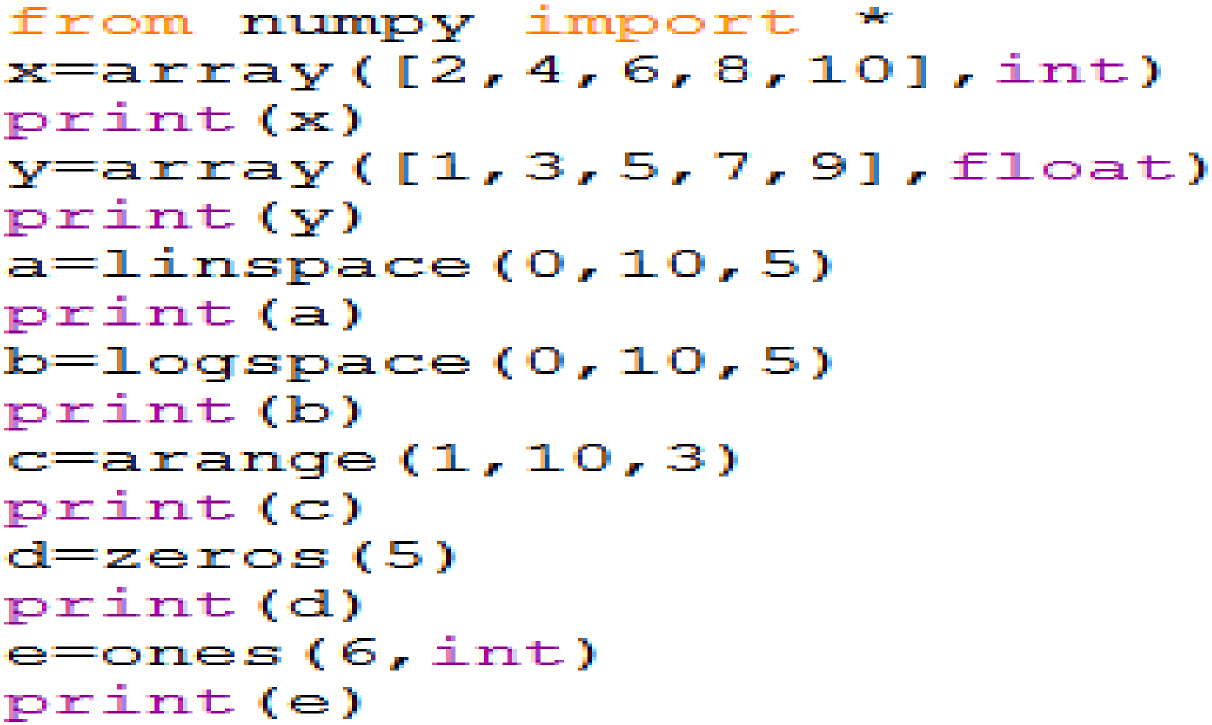
```
             logspace(start,stop,n)
```

4.arange(): it is same as range()

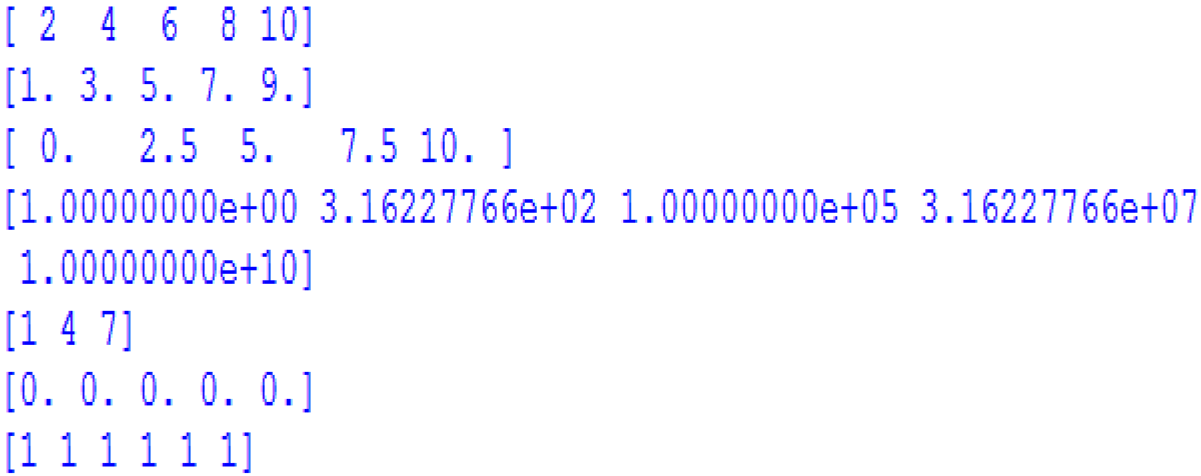
```
                          arange(start,stop,n)
```

5.zeros() and ones(): zeros(n ,datatype) ones(n ,datatype)

Example program for numpy arrays



```
from numpy import *
x=array([2,4,6,8,10],int)
print(x)
y=array([1,3,5,7,9],float)
print(y)
a=linspace(0,10,5)
print(a)
b=logspace(0,10,5)
print(b)
c=arange(1,10,3)
print(c)
d=zeros(5)
print(d)
e=ones(6,int)
print(e)
```



```
[ 2  4  6  8 10]
[1. 3. 5. 7. 9.]
[ 0.   2.5  5.   7.5 10. ]
[1.00000000e+00 3.16227766e+02 1.00000000e+05 3.16227766e+07
 1.00000000e+10]
[1 4 7]
[0. 0. 0. 0. 0.]
[1 1 1 1 1 1]
```

Array Creation

We can create an array from a regular Python list or tuple using the array function. The type of the resulting array is deduced from the type of the elements in the sequences.

EX:

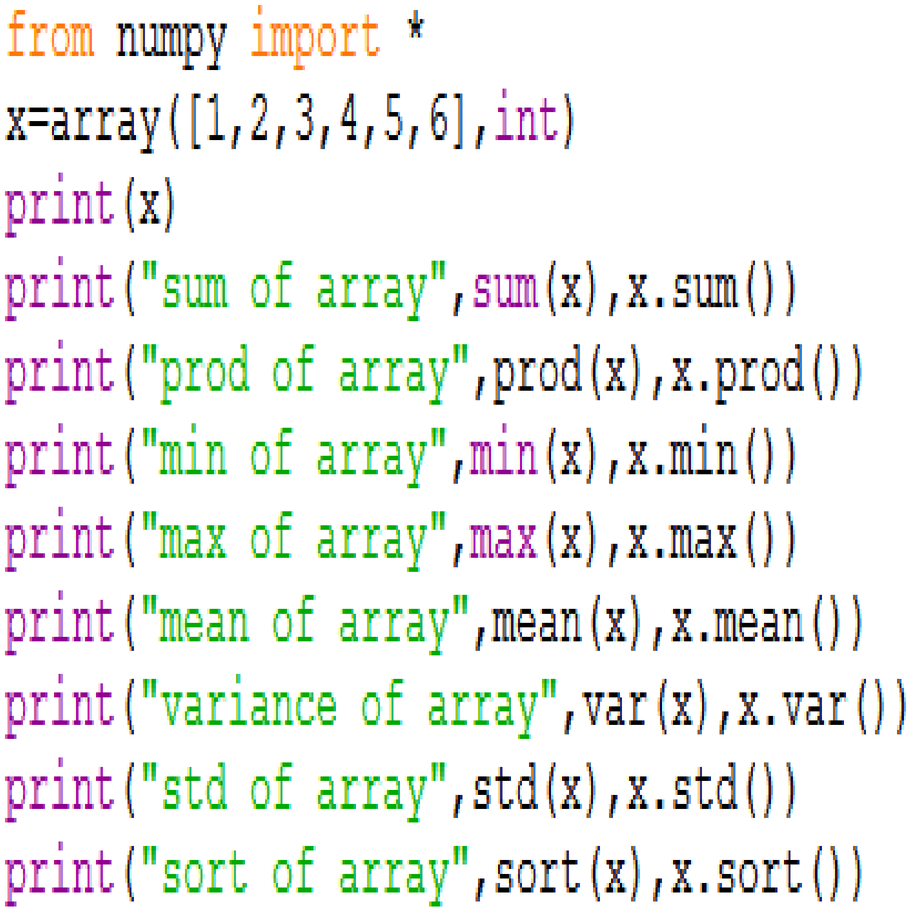
```
>>> import numpy as np
>>> a = np.array([2,3,4])
>>> a
array([2, 3, 4])
>>> a.dtype                              # dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype                              #dtype('float64')
```

- The type of the array can also be explicitly specified at creation time:

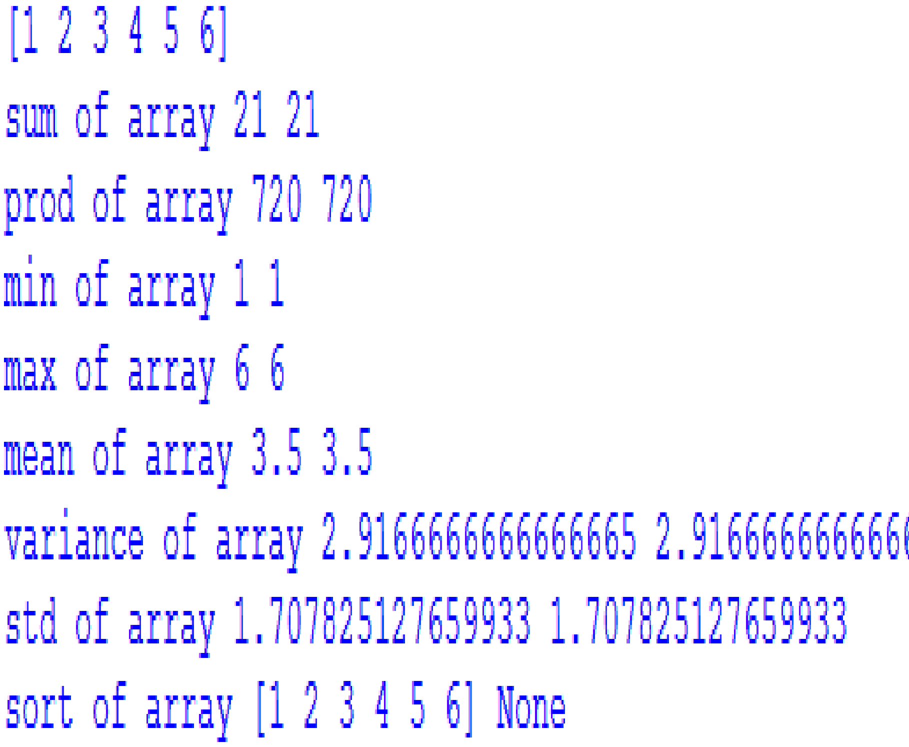
```
>>> c = np.array( [ [1,2], [3,4] ], dtype=complex )
>>> c
array([[ 1.+0.j, 2.+0.j], [ 3.+0.j, 4.+0.j]])
>>>x=np.random.randint(25,size=5)
>>> print(x)                                          #[22  3  3 20 13]
```

```
y=np.random.randint(25,size=(2,3))
>>> print(y)                                              [[22  5 15]
                                                           [23 20 17]]
a=np.random.random(5)
>>> print(a)
[0.67999849 0.38306902 0.27780021 0.23773046 0.31309768]
 a=np.random.rand(5)
>>> print(a)
        [0.70853614 0.10300682 0.9116366  0.96958943 0.8047125 ]
>>> b=np.random.rand(3,3)
>>> print(b)                    [[0.01900014 0.04698812 0.93412765]
                                 [0.10188628 0.89378112 0.19157245]
                                 [0.04150709 0.20763919 0.31780102]]
>>> c = np.arange(24).reshape(2,3,4) >>> print(c)   [[[ 0 1 2 3]
                                                     [ 4 5 6 7]
                                                     [ 8 9 10 11]]
                                                     [[12 13 14 15]
                                                     [16 17 18 19]
                                                     [20 21 22 23]]]
```

Mathematical functions in numpy



```
from numpy import *
x=array([1,2,3,4,5,6],int)
print(x)
print("sum of array",sum(x),x.sum())
print("prod of array",prod(x),x.prod())
print("min of array",min(x),x.min())
print("max of array",max(x),x.max())
print("mean of array",mean(x),x.mean())
print("variance of array",var(x),x.var())
print("std of array",std(x),x.std())
print("sort of array",sort(x),x.sort())
```



```
[1 2 3 4 5 6]
sum of array 21 21
prod of array 720 720
min of array 1 1
max of array 6 6
mean of array 3.5 3.5
variance of array 2.9166666666666665 2.9166666666666
std of array 1.707825127659933 1.707825127659933
sort of array [1 2 3 4 5 6] None
```

The attributes of an ndarray object are:

ndarray.ndim

The number of axes (dimensions) of the array.

ndarray.shape

The dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with *n*rows and *m* columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

ndarray.size

The total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

An object describing the type of the elements in the array. One can create or specify dtype's using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

ndarray.itemsize

The size in bytes of each element of the array. For example, an array of elements of type float64 has itemsize 8 (=64/8), while one of type complex32 has itemsize 4 (=32/8). It is equivalent to ndarray.dtype.itemsize.

ndarray.nbytes

The nbytes attribute gives the total number of bytes occupied by an array

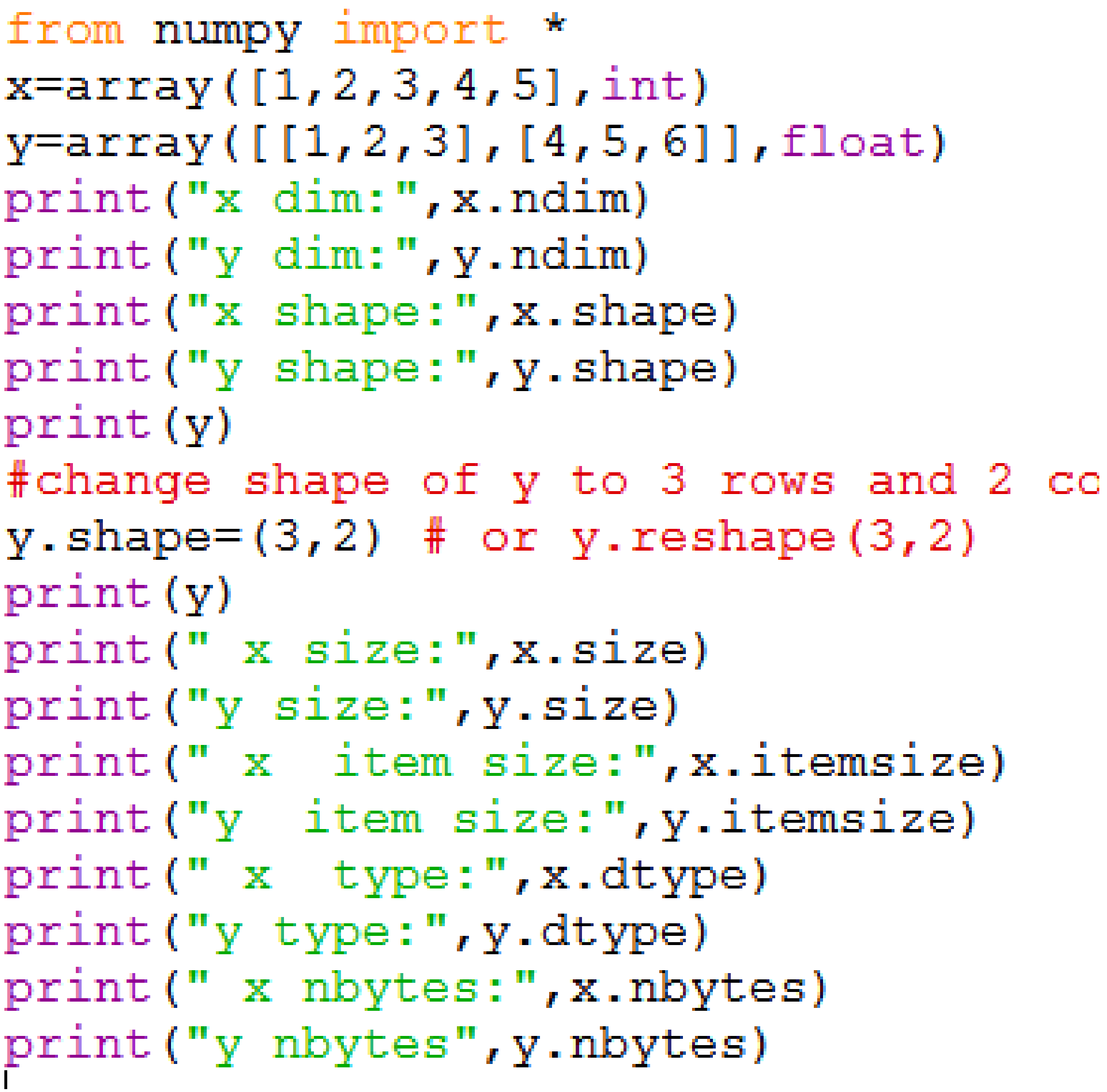
Reshape():

It is useful to change the shape of an array

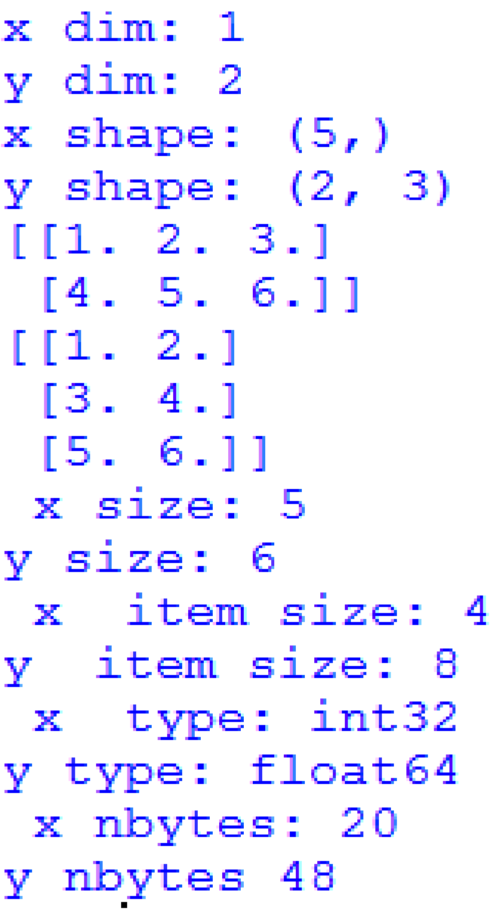
Flatten()

It is useful to return a copy of the array collapsed into one dimension

Example program



```
from numpy import *
x=array([1,2,3,4,5],int)
y=array([[1,2,3],[4,5,6]],float)
print("x dim:",x.ndim)
print("y dim:",y.ndim)
print("x shape:",x.shape)
print("y shape:",y.shape)
print(y)
#change shape of y to 3 rows and 2 co
y.shape=(3,2) # or y.reshape(3,2)
print(y)
print(" x size:",x.size)
print("y size:",y.size)
print(" x  item size:",x.itemsize)
print("y  item size:",y.itemsize)
print(" x  type:",x.dtype)
print("y type:",y.dtype)
print(" x nbytes:",x.nbytes)
print("y nbytes",y.nbytes)
```



```
x dim: 1
y dim: 2
x shape: (5,)
y shape: (2, 3)
[[1. 2. 3.]
 [4. 5. 6.]]
[[1. 2.]
 [3. 4.]
 [5. 6.]]
 x size: 5
y size: 6
 x  item size: 4
y  item size: 8
 x  type: int32
y type: float64
 x nbytes: 20
y nbytes 48
```

Working with multidimensional arrays:

The 2D arrays ,3D arrays etc. are called multi dimensional arrays
We can create multi dimensional arrays in the following ways.

1.array():is used to create a multidimensional array

a=np.array([[1,2,3,4],[5,6,7,8]])

2.Ones() and zeros() :is used to create a 2D array with several rows and columns where all the elements we be taken as 1

ones((r,c),dtype)

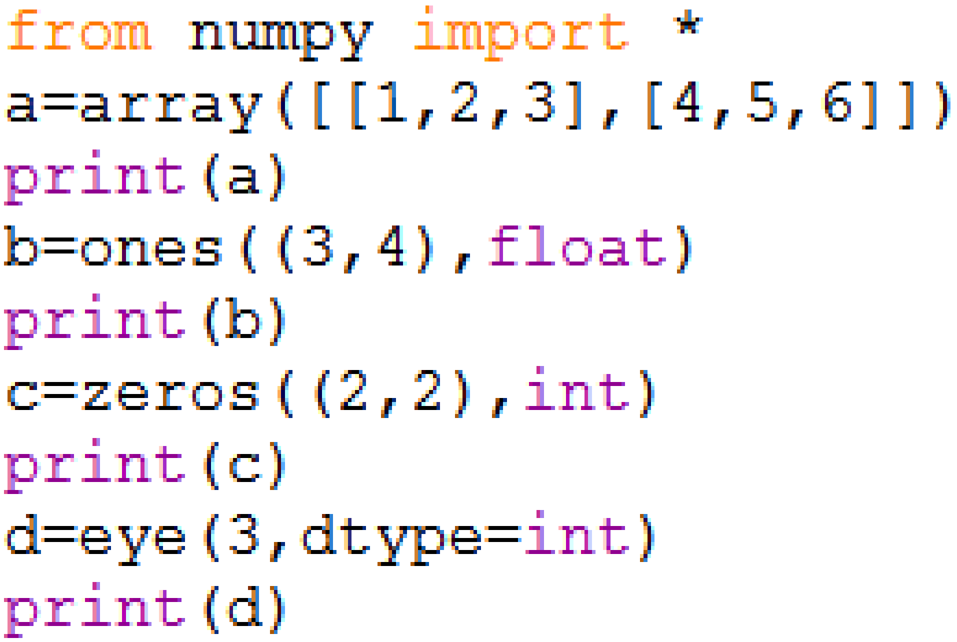
3.eye(): is used to create 2D array and fills the elements in the diagonal with 1s

eye(n,dtype=datatype)

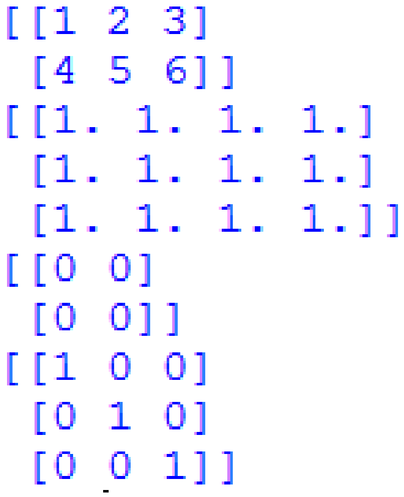
4.reshape():is used to convert a 1D array inta a multidimensional array

reshape(arrayneme,(n,r,c)

Example program



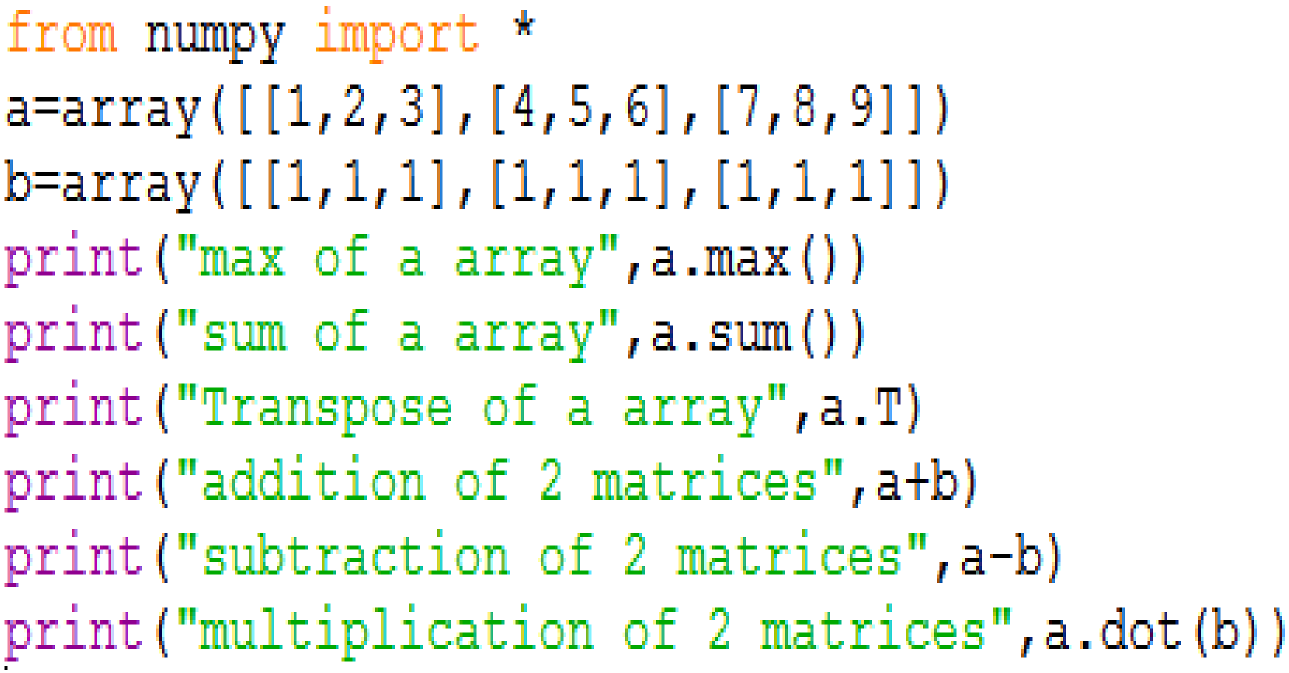
```
from numpy import *
a=array([[1,2,3],[4,5,6]])
print(a)
b=ones((3,4),float)
print(b)
c=zeros((2,2),int)
print(c)
d=eye(3,dtype=int)
print(d)
```



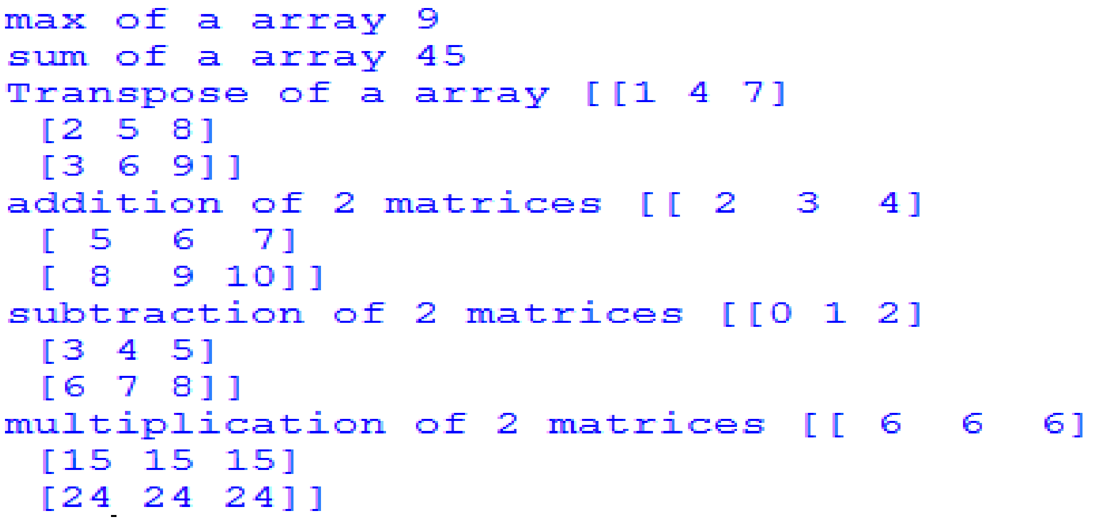
```
[[1 2 3]
 [4 5 6]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[0 0]
 [0 0]]
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.18626021, 0.34556073, 0.39676747], [ 0.53881673, 0.41919451, 0.6852195 ]])
```

Example program



```
from numpy import *
a=array([[1,2,3],[4,5,6],[7,8,9]])
b=array([[1,1,1],[1,1,1],[1,1,1]])
print("max of a array",a.max())
print("sum of a array",a.sum())
print("Transpose of a array",a.T)
print("addition of 2 matrices",a+b)
print("subtraction of 2 matrices",a-b)
print("multiplication of 2 matrices",a.dot(b))
```



```
max of a array 9
sum of a array 45
Transpose of a array [[1 4 7]
 [2 5 8]
 [3 6 9]]
addition of 2 matrices [[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
subtraction of 2 matrices [[0 1 2]
 [3 4 5]
 [6 7 8]]
multiplication of 2 matrices [[ 6  6  6]
 [15 15 15]
 [24 24 24]]
```

plotpy

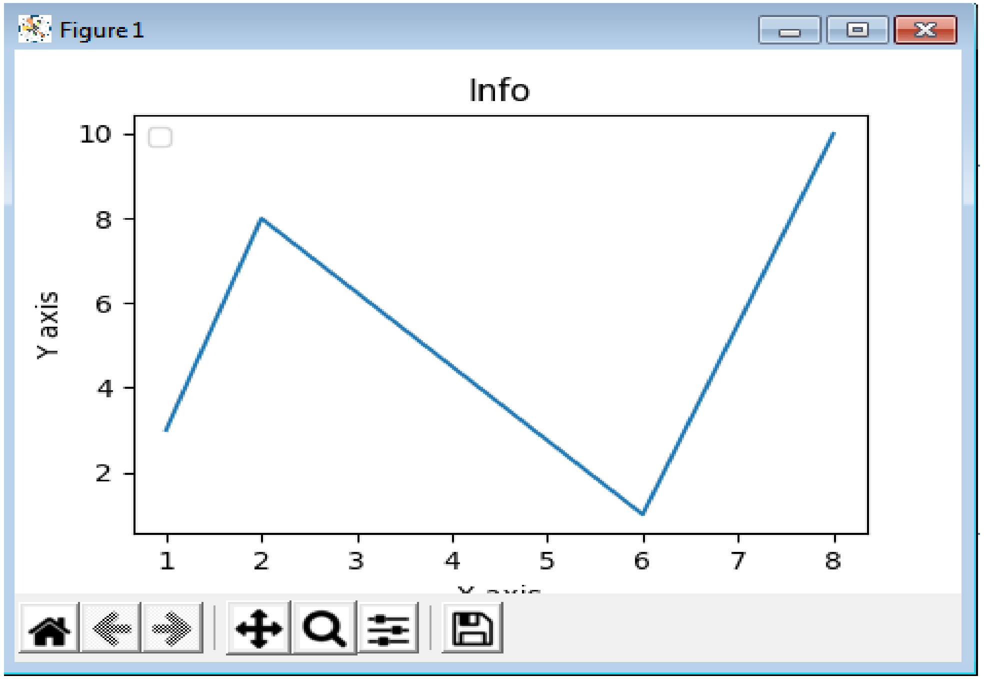
- *pyplot* is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.
- *pyplot* provides a procedural interface to the matplotlib object-oriented plotting library. It is modeled closely after Matlab.
- The majority of plotting commands in pyplot have Matlab analogs with similar arguments. Important commands are explained with interactive examples.

There are various plots which can be created using pyplot. Some of them are listed below

Plot ,Bar graphs, Histograms, Scatter plot and Pie charts

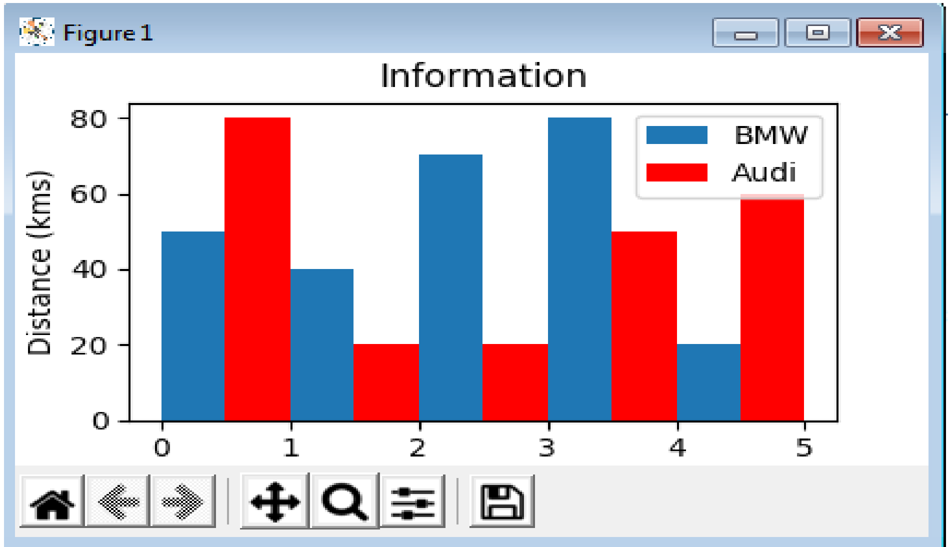
Plot:The plot() function draws a line from point to point.

```
from matplotlib import pyplot as plt
x = [1,2,6,8]
y = [3,8,1,10]
plt.plot(x,y)
plt.title('Info')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.legend()
plt.show()
```

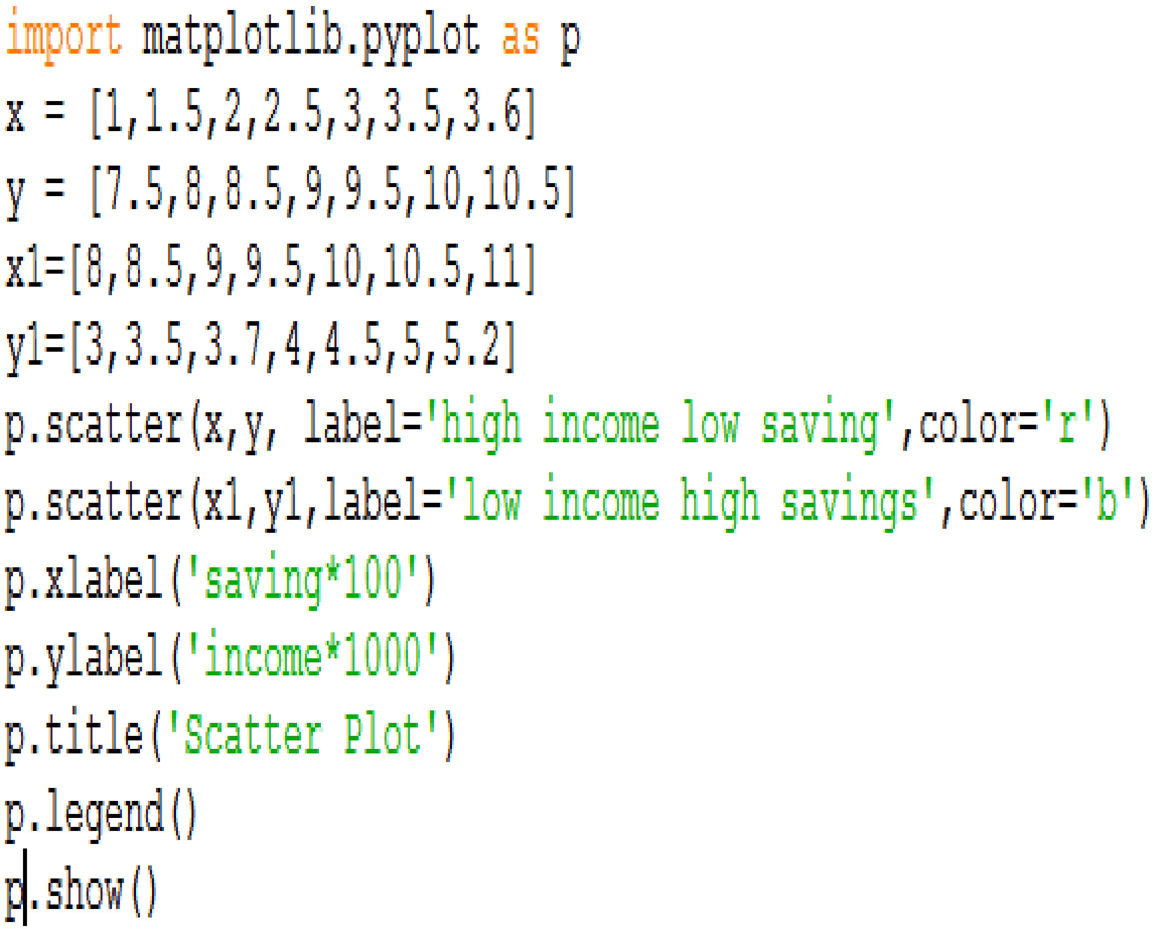


Bar graph: A bar graph uses bars to compare data among different categories. It is well suited when you want to measure the changes over a period of time. It can be represented horizontally or vertically.

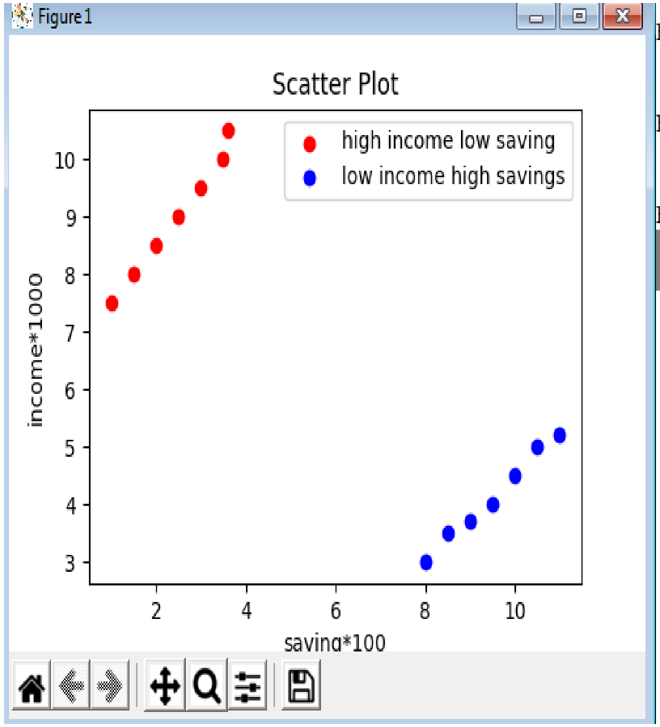
```
from matplotlib import pyplot as p
p.bar([0.25,1.25,2.25,3.25,4.25],[50,40,70,80,20],
      label="BMW",width=.5)
p.bar([.75,1.75,2.75,3.75,4.75],[80,20,20,50,60],
      label="Audi", color='r',width=.5)
p.legend()
p.xlabel('Days')
p.ylabel('Distance (kms)')
p.title('Information')
p.show()
```



Scatter plot:we need scatter plots in order to compare variables, for example, how much one variable is affected by another variable to build a relation out of it.

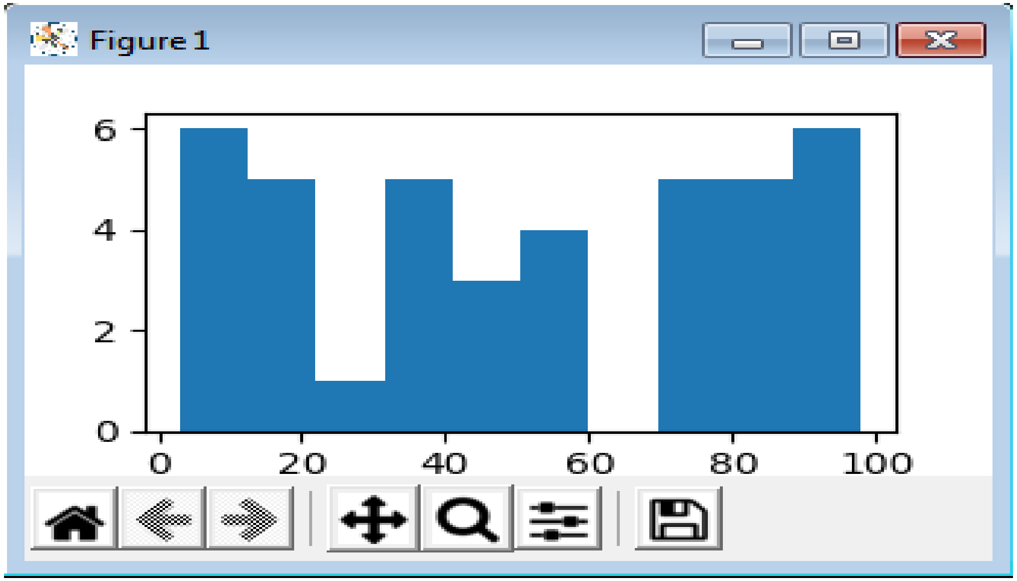


```
import matplotlib.pyplot as p
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]
x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]
p.scatter(x,y, label='high income low saving',color='r')
p.scatter(x1,y1,label='low income high savings',color='b')
p.xlabel('saving*100')
p.ylabel('income*1000')
p.title('Scatter Plot')
p.legend()
p.show()
```



Histograms are used to show a distribution whereas a bar chart is used to compare different entities. Histograms are useful when you have arrays or a very long list.

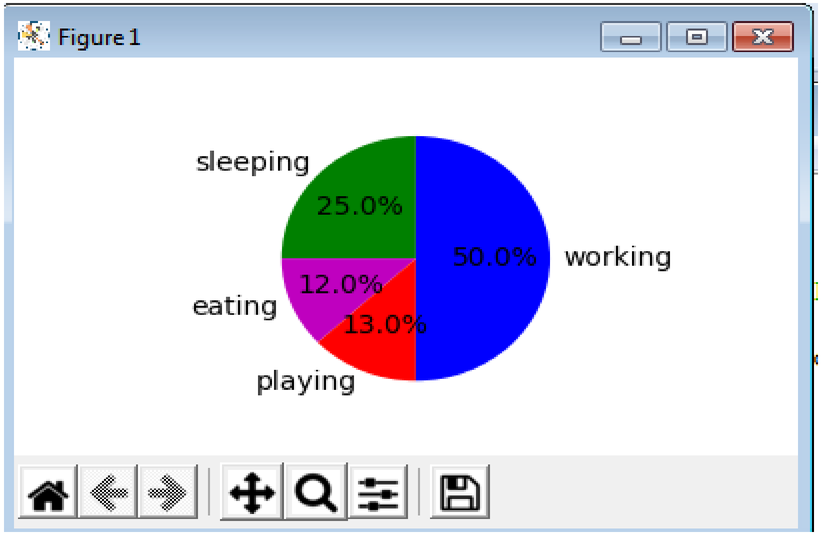
```
import matplotlib.pyplot as p
import numpy as np
age = np.random.randint(100, size=40)
print(age)
p.hist(age)
p.show()
```



Pie charts

A pie chart refers to a circular graph which is broken down into segments i.e. slices of pie. It is basically used to show the percentage or proportional data where each slice of pie represents a category.

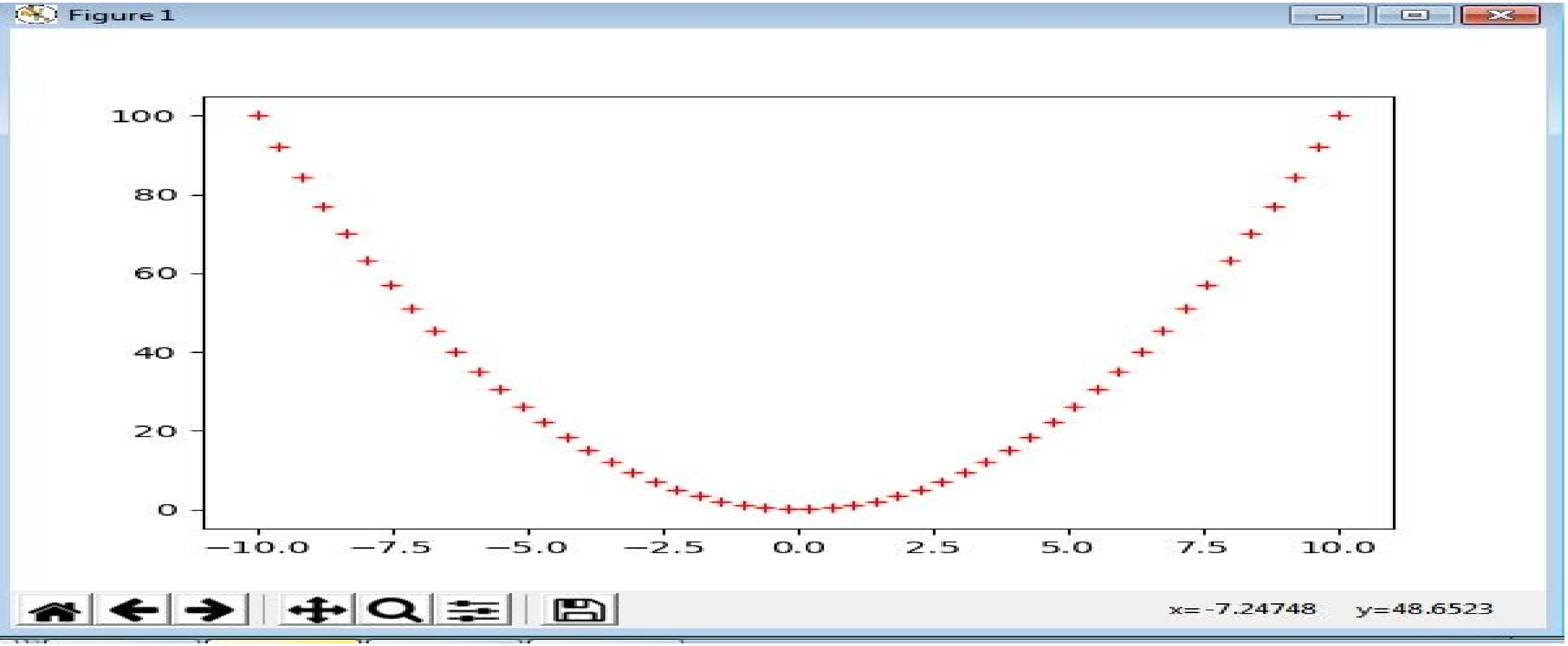
```python
import matplotlib.pyplot as plt
import numpy as np
y = np.array([25,12,13,50])
activities = ['sleeping','eating','playing','working']
cols = ['g','m','r','b']
plt.pie(y, labels = activities,colors=cols,autopct='%1.1f%%',startangle = 90)
plt.show()
```



EX 1:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-10, 10)
plt.plot(x, x**2, 'r+')
plt.show()
```

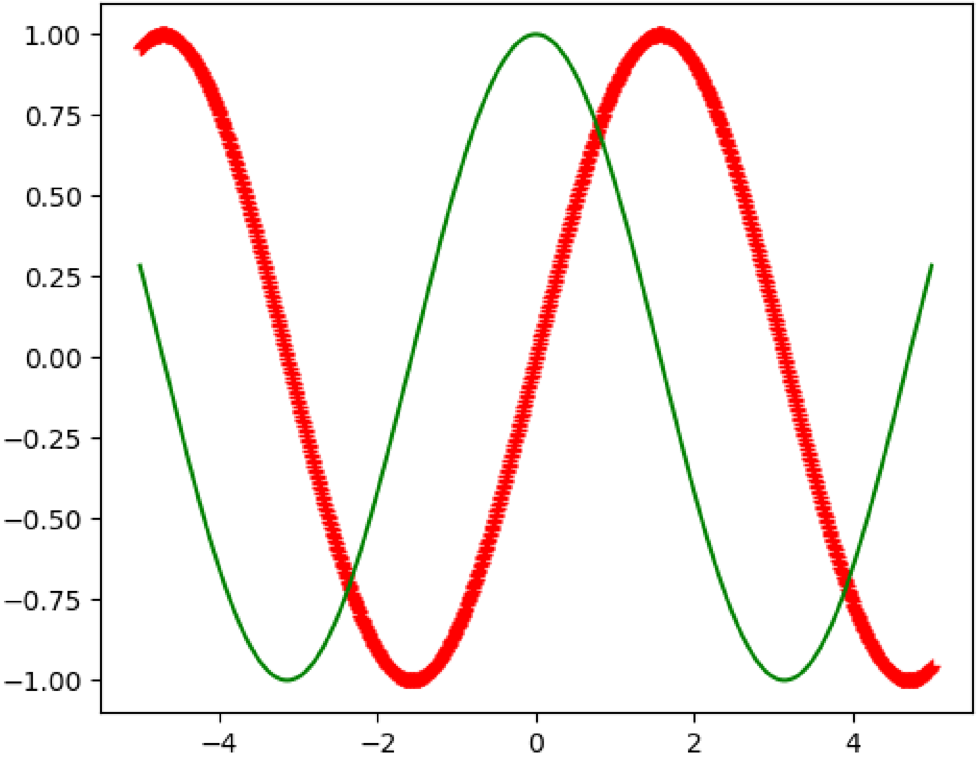
Output:



EX 2:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 1000)
plt.plot(x, np.sin(x), "r+")
plt.plot(x, np.cos(x), "g-")
plt.show()
```

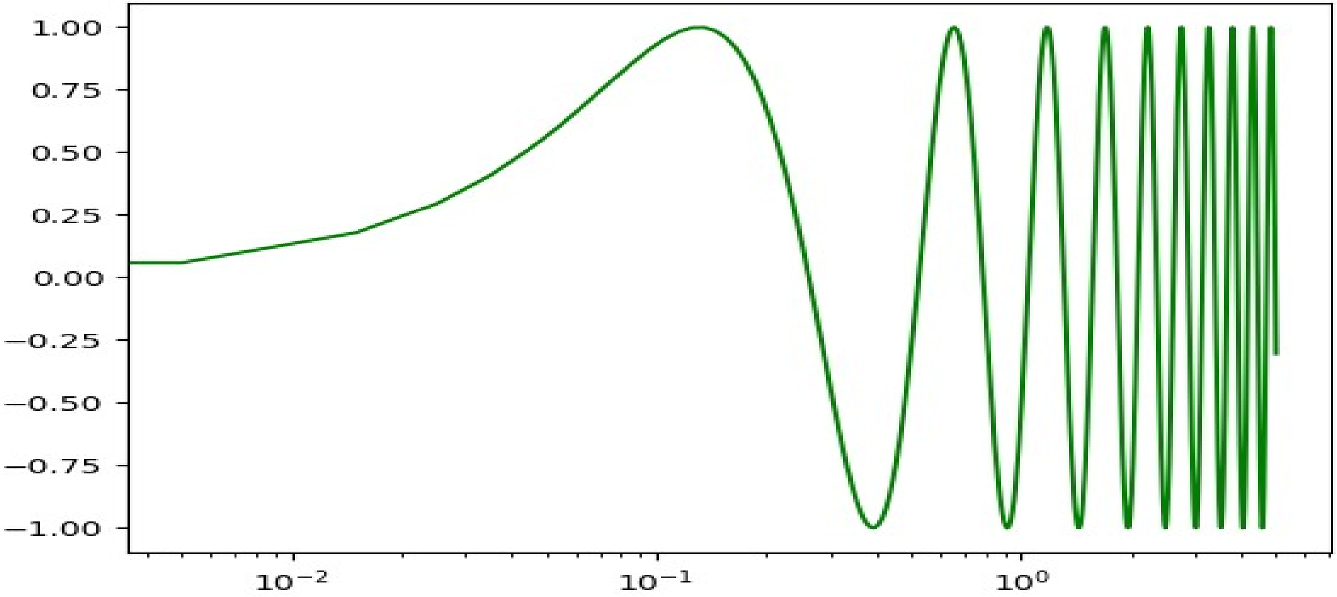
Output



Ex 3: semilogx(**args*, ***kwargs*)Plot curves with logarithmic x-axis scale

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 1000)
plt.semilogx(x, np.sin(12*x), "g-")
plt.show()
```

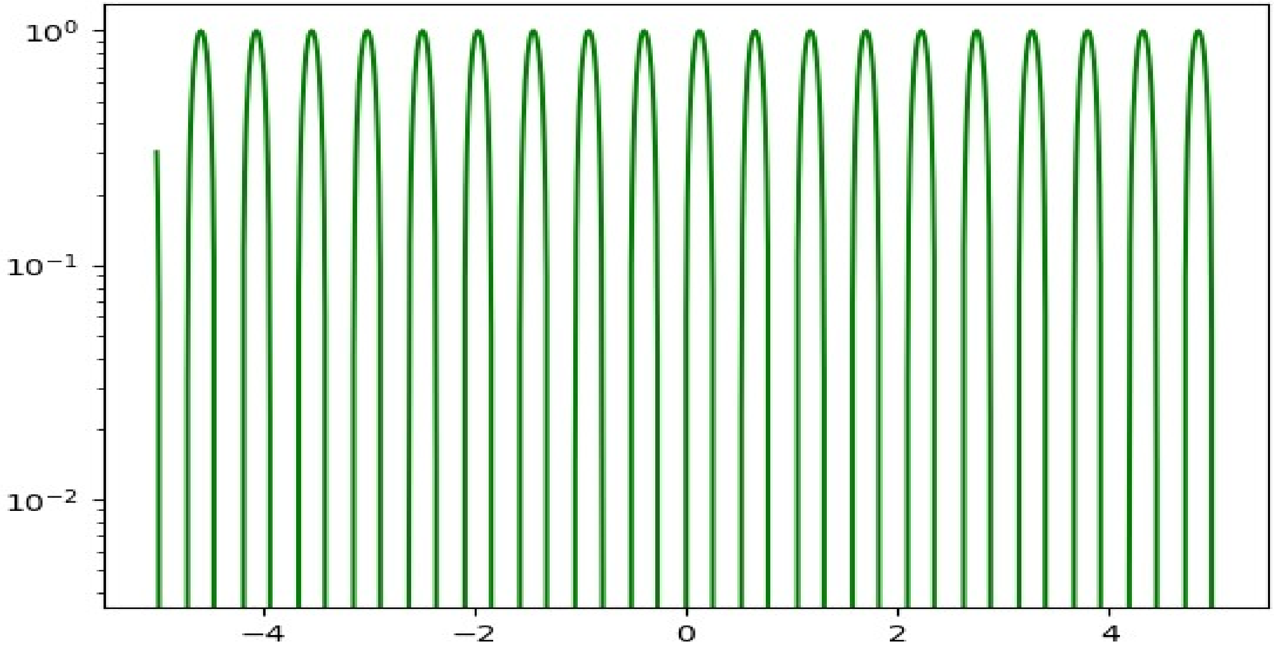
Output



Ex 4 semilogy(**args*, ***kwargs*)Plot curves with logarithmic y-axis scale

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 1000)
plt.semilogy(x, np.sin(12*x), "g-")
plt.show()
```

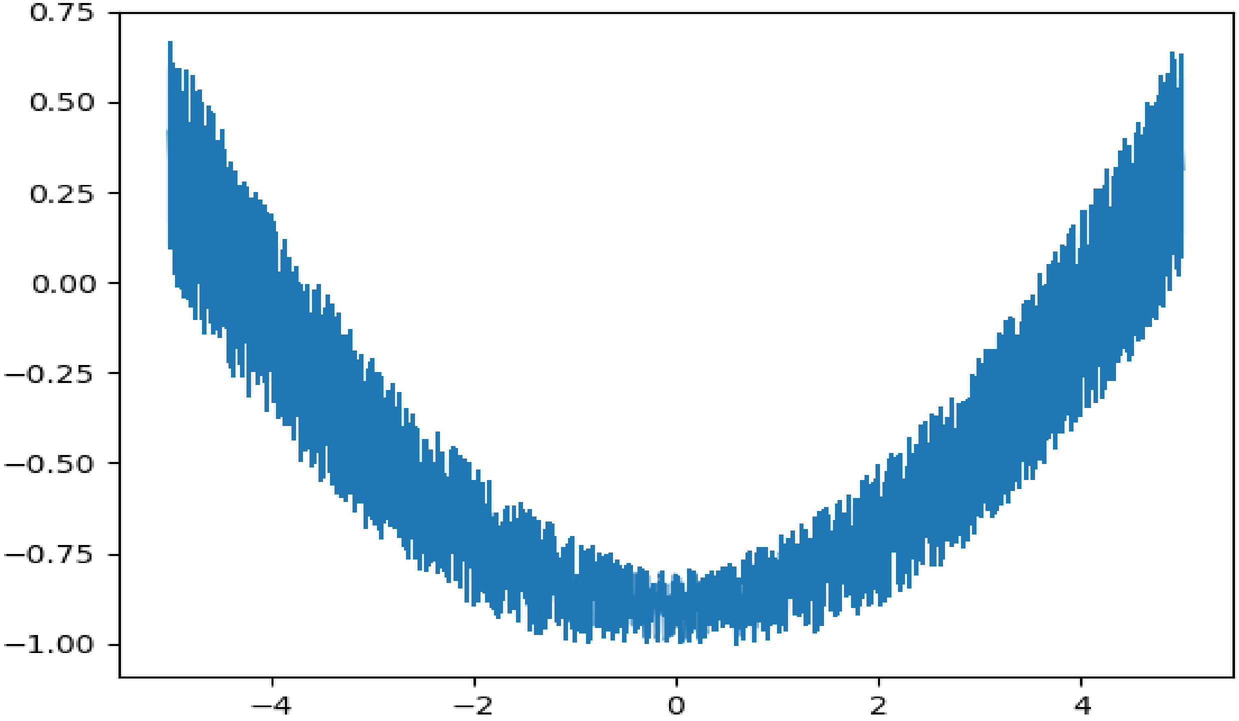
Output



Ex 5:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 1000)
plt.errorbar(x, -1+x**2/20+.2*np.random.rand(len(x)), x/20)
plt.show()
```

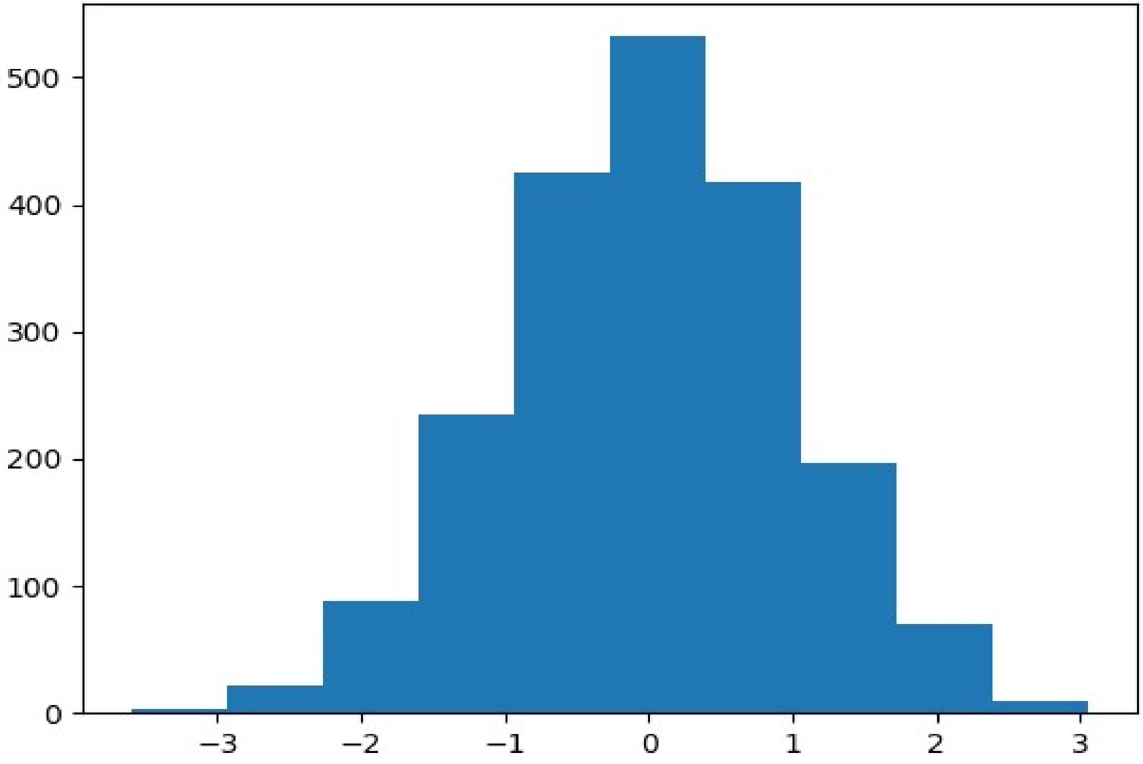
Output



Ex 6 Plot 1-D histogram

```
import matplotlib.pyplot as plt
from numpy.random import normal
data = normal(0, 1, (2000, ))
plt.hist(data)
plt.show()
```

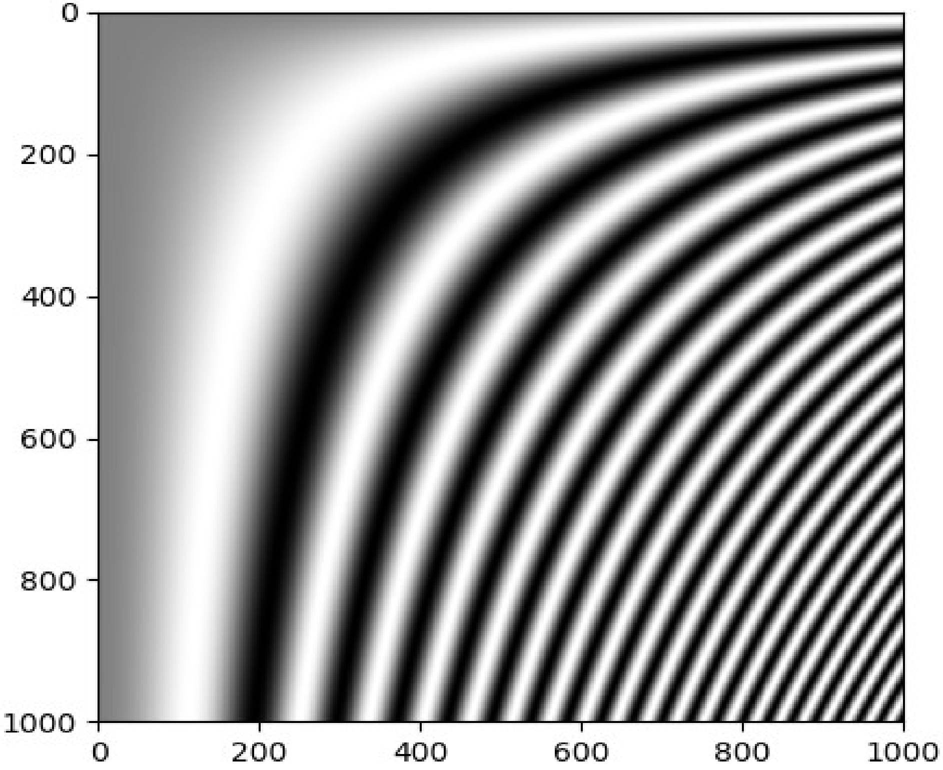
Output



Ex 7

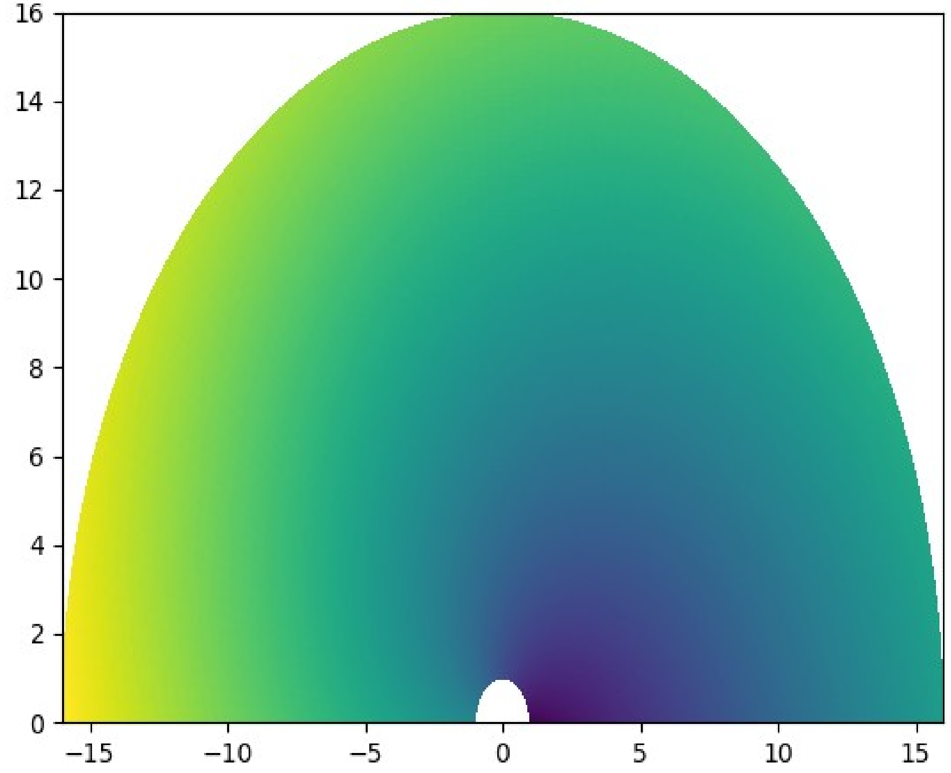
```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5, 5, 1000)
img = np.fromfunction(lambda x, y: np.sin((x/200.)*(y/200.)**2), (1000, 1000))
plt.gray()
plt.imshow(img)
plt.show()
```

Output



Ex 8 Create a pseudocolor plot of a 2-D array

```
import matplotlib.pyplot as plt
import numpy as np
r = np.linspace(1., 16, 100)
th = np.linspace(0., np.pi, 100)
R, TH = np.meshgrid(r, th)
X = R*np.cos(TH)
Y = R*np.sin(TH)
Z = 4*TH+R
plt.pcolor(X, Y, Z)
plt.show()
```



scipy

- SciPy is a free and open-source Python library used for scientific computing and technical computing.
- SciPy contains modules for optimization, linear algebra, integration, special functions, FFT, signal and image processing, solvers and other tasks common in science and engineering.
- SciPy builds on the Numpy array object and is part of the Numpy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries.

scipy is composed of task-specific sub-modules

scipy.cluster	Vector quantization / Kmeans
scipy.constants	Physical and mathematical constants
scipy.fft	Fourier transform
scipy.integrate	Integration routines
scipy.interpolate	Interpolation
scipy.io	Data input and output
scipy.linalg	Linear algebra routines
scipy.ndimage	n-dimensional image package
scipy.odr	Orthogonal distance regression
scipy.optimize	Optimization
scipy.signal	Signal processing
scipy.sparse	Sparse matrices
scipy.spatial	Spatial data structures and algorithms
scipy.special	Any special mathematical functions
scipy.stats	Statistics

Special Functions:

SciPy provides a number of special functions that are used in mathematical physics such as elliptic, convenience functions, gamma, beta, etc.

Example:

```
from scipy.special import *
cb = cbrt([8, 125])
print(cb)
com = comb(5, 2)
print(com)
per = perm(5, 2)
print(per)
c = sindg(90)
print(c)
d = cosdg(45)
print(d)
```

```
[2. 5.]
10.0
20.0
1.0
0.7071067811865475
```

Integration Functions:

SciPy provides a number of functions to solve integrals.

Ex:

```
from scipy import integrate
a = lambda y, x: x*y**2
b = lambda x: 1
c = lambda x: -1
integrate.dblquad(a, 0, 2, b, c)
```

Optimization Functions:

The scipy.optimize provides a number of commonly used optimization algorithms

Ex:

```
import numpy as np
from scipy.optimize import rosen
a = 1.2 * np.arange(5)
rosen(a)
```

Output:

```
7371.0399999999945
```

Interpolation Functions:

Interpolation refers to constructing new data points within a set of known data points.

Ex:

```
from scipy import interpolate
import numpy as np
x = np.arange(5, 20)
y = np.exp(x/3.0)
f = interpolate.interp1d(x, y)
```

Fourier Transform Functions:

Fourier analysis is a method that deals with expressing a function as a sum of periodic components and recovering the signal from those components. The *fft* functions can be used to return the discrete Fourier transform of a real or complex sequence.

Ex:

```
from scipy.fftpack import fft, ifft
x = np.array([0,1,2,3])
y = fft(x)
print(y)
```

io module SciPy has many modules, classes, and functions available to read data from and write data to a variety of file formats.

EX

```
import numpy as np
from scipy import io as spio
a = np.ones((3, 3))
spio.savemat('file.mat', {'a': a})
data = spio.loadmat('file.mat')
print(data['a'])
```

Output

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

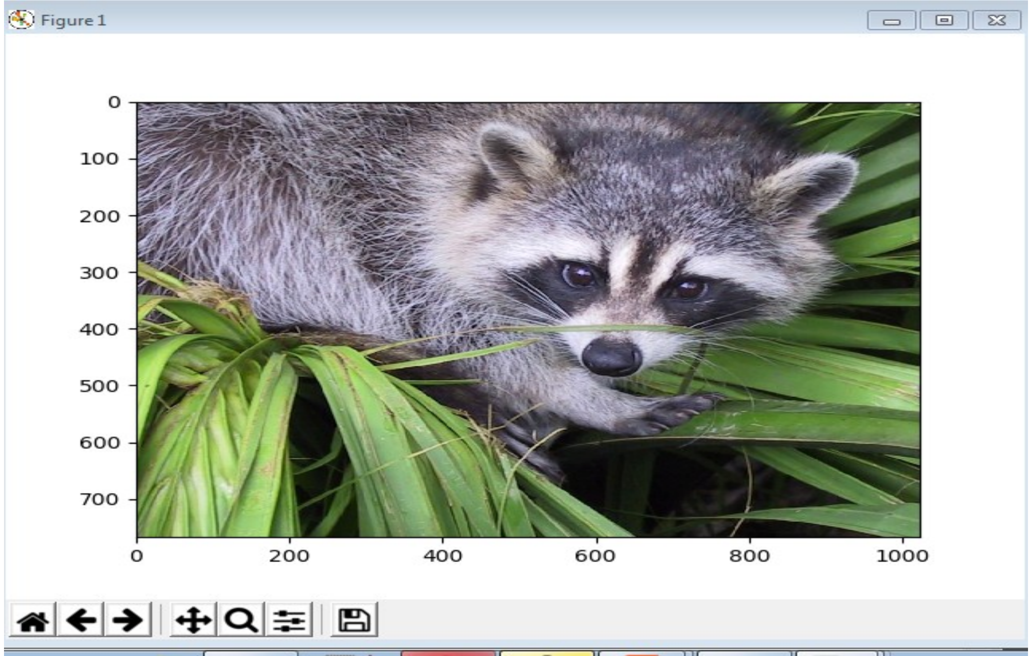
scipy.ndimage is a submodule of SciPy which is mostly used for performing an image related operation

EX

```
import scipy
from scipy import misc
import matplotlib.pyplot as plt
face=scipy.misc.face()
print(face.shape)
print(face.max())
print(face.dtype)
plt.gray()
plt.imshow(face)
plt.show()
```

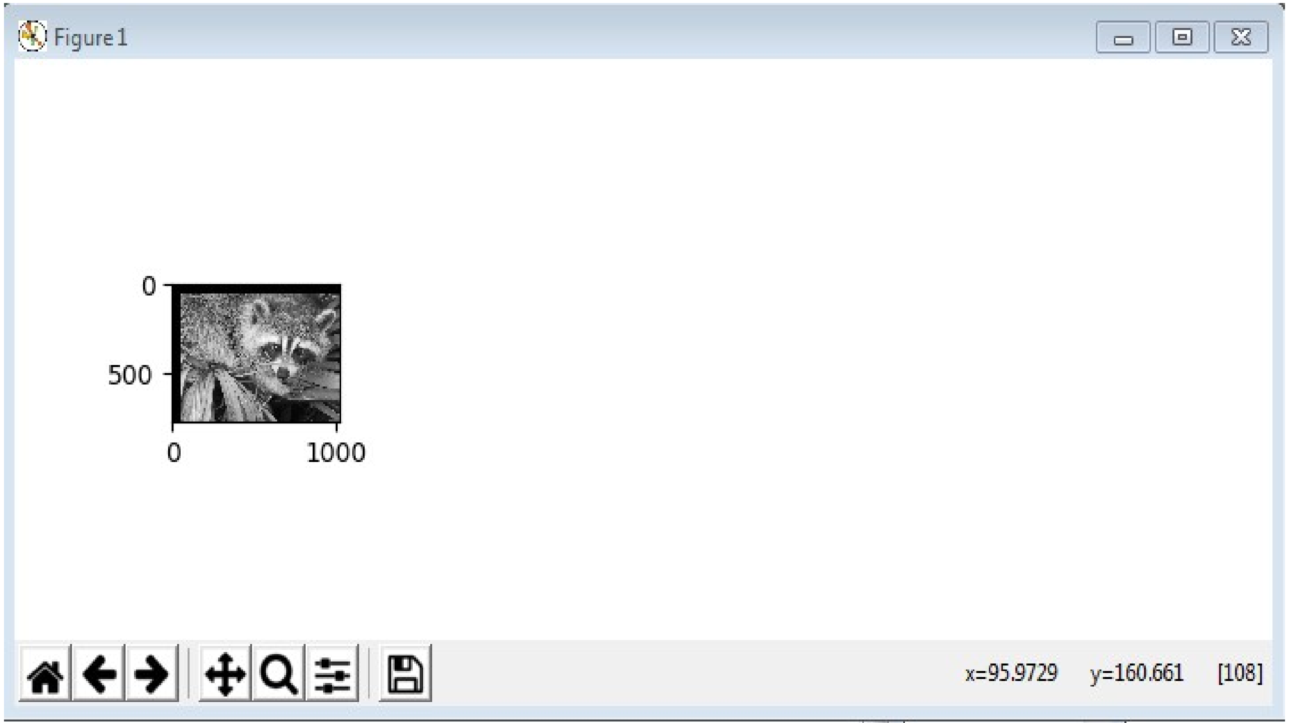
Output

```
(768, 1024, 3)
255
uint8
```



```
import scipy
from scipy import ndimage,misc
import matplotlib.pyplot as plt
face=misc.face(gray=True)
shifted_face = ndimage.shift(face, (50, 50))
plt.figure(figsize=(15, 3))
plt.subplot(151)
plt.imshow(shifted_face,cmap=plt.cm.gray)
plt.show()
```

Output



GUI Programming

Introduction

- A person who interacts with a software or application is called a user.
- Two ways for a user to interact with any application.

❑ Character User Interface

The user gives some commands to perform the work

❑ Graphical User Interface

The user interacts with an application through Graphics or pictures or images.

Advantages of GUI

- User friendly
- It adds attraction and beauty to any application by adding pictures, colors, menus, animations etc.
- possible to simulate the real life objects.
- It helps to create graphical components like push button, radio button, check button etc.

Tkinter programming

- Python offers Tkinter module to create graphics programs
- Tkinter represents toolkit interface for GUI
- We can enable the interface by using the classes of Tk module of TCL/TK language(Tool command Language).
- TCL is a dynamic programming language suitable for web and desktop applications, networking, administration, testing etc.

The following steps are involved in basic GUI programs

- We should create the root window. The root window is the top level window provides rectangular space on the screen where we display text, colors, images, components etc.
- In the root window , the space allocation is done by Frame or canvas. These are child windows of root window.
- We use canvas for displaying drawings like lines, arcs, circles, shapes etc.
- The frame is used for displaying components like push buttons, check buttons, menus etc. These are called widgets.

Root window

- To display the graphical output , the space is allocated to any GUI program called top level window or root window.
- We can reach this root window by creating an object to the class.
- The root window will have a title bar that contains minimize, resize and close options.

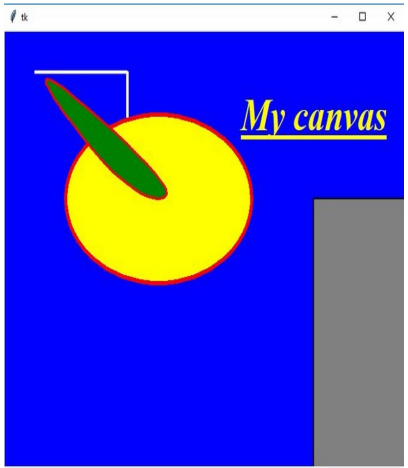
EX

```
# creating root window

from tkinter import *

root = Tk()

root.title(“my window”)

root.geometry(“400x300”)

root.mainloop()
```

Canvas: This is a container which is used to draw shapes like lines, curves ,arcs and circles

c=Canvas(master , option=value…..)

```
from tkinter import *
w1=Tk()
c=Canvas(w1,bg="blue",height=700,width=1200,cursor="pencil")
id=c.create_line(50,50,200,50,200,150,width=4,fill="white")
id=c.create_oval(100,100,400,300,width=5,fill="yellow",outline="red",activefill="gree
n")
id=c.create_polygon(10,10,200,200,300,200,width=3,fill="green",outline="red",
            smooth=1,activefill="lightblue")
id=c.create_rectangle(500,200,700,600,width=2,fill="gray",outline="black",
             activefill="yellow")
fnt=("Times",40,"bold italic underline")
id=c.create_text(500,100,text="My canvas",font=fnt,fill="yellow",
          activefill="green")
c.pack()
w1.mainloop()
```



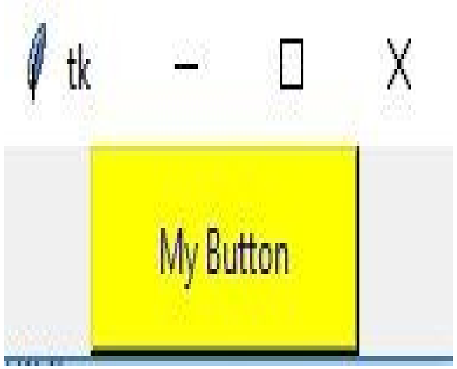
Frame: This is a container which is used to display widgets like buttons,checkbuttons or menu

f=Frame(master,option=value,-----)

Button: It is used to display buttons in your application

b=Button(master, text="submit", command=function,option=value,…….)

```
from tkinter import *
def Clickme(self):
    print("you have clicked me")
w=Tk()
f=Frame(w,height=200,width=300).pack()
b=Button(f,text="My Button",width=15,height=2,bg="yellow",fg="blue",
 activebackground="green",activeforeground="red", command=Clickme())
w.mainloop()
```



Widgets: A widget is aGUI component that is displayed on the screen and can perform a task as desired by the user. We create widgets as objects.

1. Button
2. Label
3. Checkbutton
4. Radiobutton
5. Entry
6. Spinbox
7. Listbox
8. Text
9. Message
10. Scrollbar
11. Menu

Layout Management: once we create widgets or components, we can arrange them in the frame in a particular manner. Arranging the widgets in the frame is called layout management. There are three types of layout managers.

1.Pack layout manager: It uses pack() method which associate a widget with its parent component. While using the pack() method, we can mention the position of the widget using fill or side options. The fill option can take the values X, Y,BOTH,NONE. The value X represents that the widget should occupy the frame horizontally, The value Y represents that the widget should occupy the frame vertically. Both represents that the widget should occupy the frame horizontally. The option side which is used to place the widgets side by side.' side ' can take the values LEFT,RIGHT,TOP or BOTTOM. The default value is TOP

```
b1.pack(side=Left)
b1.pack(fill=x)
```

2.Grid layout manager :It uses the grid() method to arrange the widgets in a two dimensional table that contains rows and columns. The position of a widget is defined by a row and column number

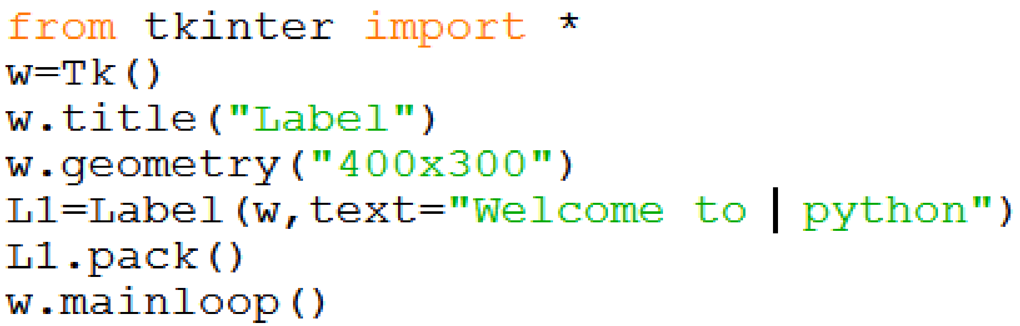
```
b.grid(row=0,column=0)
```

3.Place layout manager: It uses the place() method to arrange the widgets.The position of the widget is defined by x and y coordinates

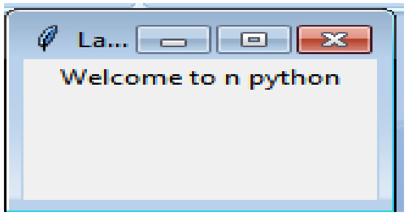
```
b1.place(x=10,y=80)
```

Label :The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

l1=Label(master , text=“ hello", option=value,….)

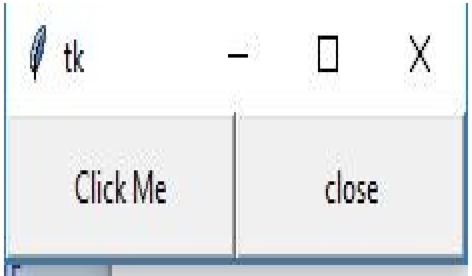


```
from tkinter import *
w=Tk()
w.title("Label")
w.geometry("400x300")
L1=Label(w,text="Welcome to python")
L1.pack()
w.mainloop()
```



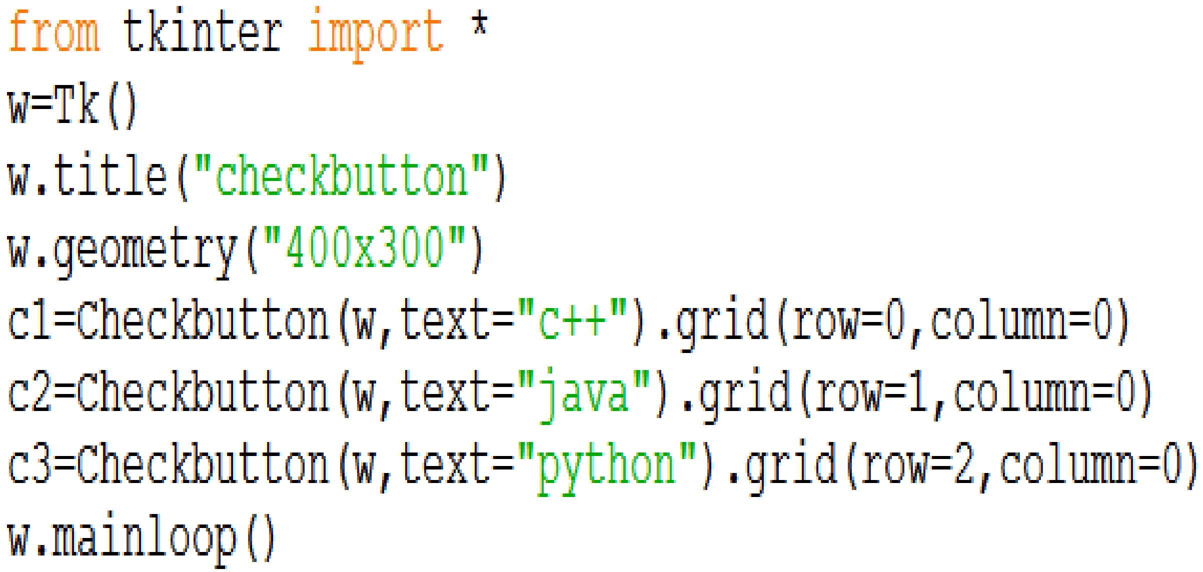
Welcome to n python

```
from tkinter import *
class MyButtons:
   def __init__(self,w):
      self.b1=Button(w,text="ClickMe“,width=15,height=2,command=self.buttonClick)
      self.b2=Button(w,text="close",width=15,height=2,command=w.destroy)
      self.b1.grid(row=0,column=1)
      self.b2.grid(row=0,column=2)
   def buttonClick(self):
      self.lbl=Label(w,text="welcometopython",width=20,height=2,
                        font=("Courier",-30,"bold underline"),fg="blue")
      self.lbl.grid(row=2,column=0)
w=Tk()
mb=MyButtons(w)
w.mainloop()
```

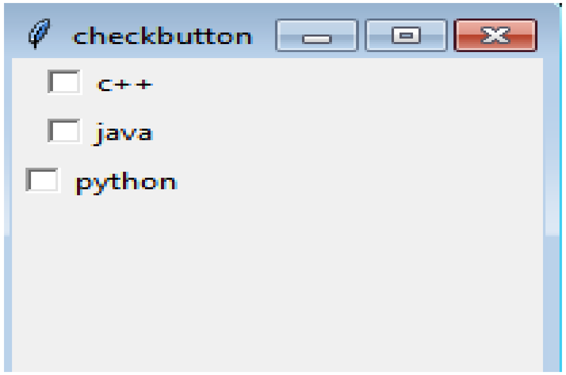


Checkbutton: The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

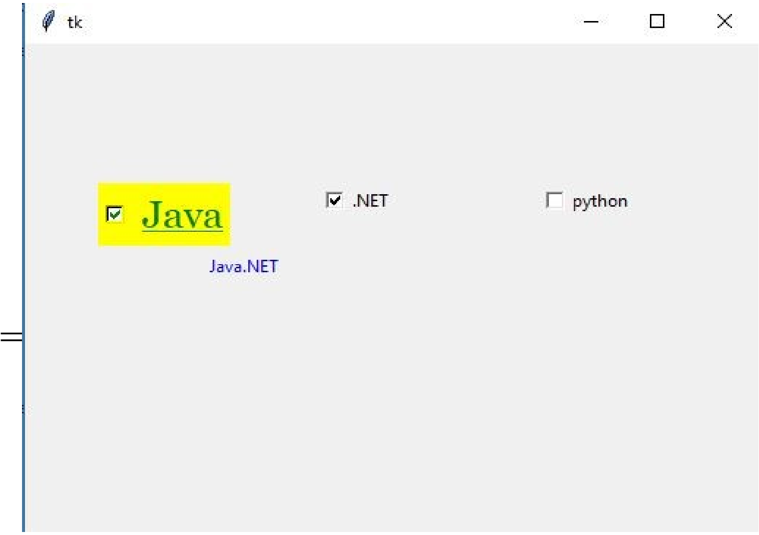
c= Checkbutton(master , text=‘submit ’, variable=var1,command=function , option=value……)



```
from tkinter import *
w=Tk()
w.title("checkbutton")
w.geometry("400x300")
c1=Checkbutton(w,text="c++").grid(row=0,column=0)
c2=Checkbutton(w,text="java").grid(row=1,column=0)
c3=Checkbutton(w,text="python").grid(row=2,column=0)
w.mainloop()
```



```
from tkinter import *
class Mycheck:
    def __init__(self,w):
        self.var1=IntVar()
        self.var2=IntVar()
        self.var3=IntVar()
        self.c1=Checkbutton(w, text="Java“, variable=self.var1,command= self.display)
        self.c2=Checkbutton(w,text=".NET“,variable=self.var2,command=  self.display)
        self.c3=Checkbutton(w,text="python",variable=self.var3,command= self.display)
        self.c1.place(x=50,y=100)
        self.c2.place(x=200,y=100)
        self.c3.place(x=350,y=100)
    def display(self):
        x=self.var1.get()
        y=self.var2.get()
        z=self.var3.get()
        str=''
        if x==1:
            str+="Java"
        if y==1:
            str+=".NET"
        if z==1:
            str+="python"
        lbl=Label(text=str,fg="blue").place(x=50,y=150,width=
w=Tk()
mb=Mycheck(w)
w.mainloop()
```



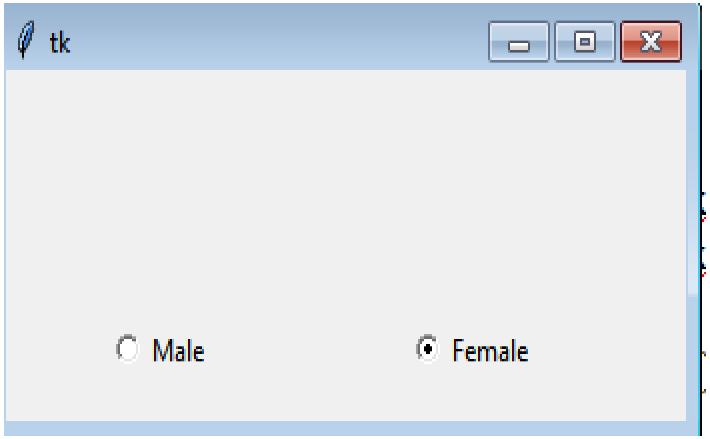
☑ Java ☑ .NET □ python

Java.NET

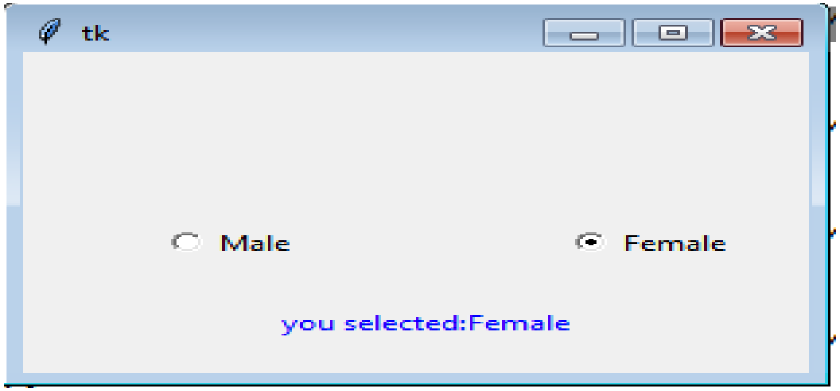
Radiobutton :The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

r= Radiobutton(master, text=“option1”,variable=z , value=1,command=fun,…..)

```
from tkinter import *
w=Tk()
def dis():
    print(x.get())
x=IntVar()
r1= Radiobutton(w,text="Male", variable=x,value=1,command=dis)
r2= Radiobutton(w,text="Female",variable=x,value=2,command=dis)
r1.place(x=50,y=100)
r2.place(x=200,y=100)
w.mainloop()
```



```
from tkinter import *
class Myradio:
    def __init__(self,w):
        self.var=IntVar()
        self.r1= Radiobutton(w,text="Male“, variable=self.var ,value=1,command=  self.display)
        self.r2= Radiobutton(w,text="Female“ , variable=self.var ,value=2,command=self.display)
        self.r1.place(x=50,y=100)
        self.r2.place(x=200,y=100)
    def display(self):
        x=self.var.get()
        str=''
        if x==1:
            str+="you selected:Male"
        if x==2:
            str+="you selected:Female"
        lbl=Label(text=str,fg="blue").place(x=50,y=150,width=200,height=20)
w=Tk()
mb=Myradio(w)
w.mainloop()
```



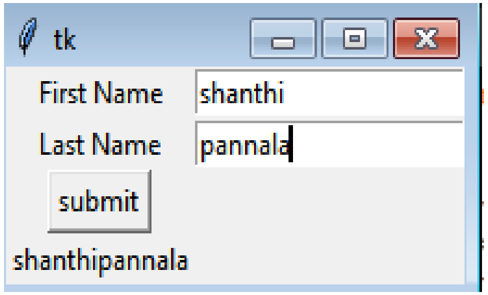
○ Male ● Female

you selected:Female

Entry :The Entry widget is used to display a single-line text field for accepting values from a user.

e=Entry(master , option=value,….)

```
from tkinter import *
w = Tk()
def dis():
    str1=e1.get() + e2.get()
    Label(text=str1).grid(row=3)
l1=Label(w,text='First Name').grid(row=0,column=0)
l2=Label(w, text='Last Name').grid(row=1,column=0)
e1=Entry(w)
e1.grid(row=0, column=1)
e2=Entry(w)
e2.grid(row=1, column=1)
b=Button(w,text='submit',command=dis).grid(row=2)
mainloop()
```

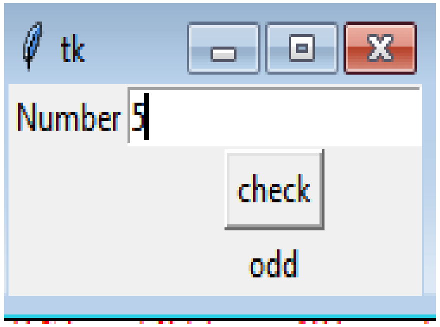


First Name shanthi

Last Name pannala

shanthipannala

```
import tkinter as t
w=t.Tk()
def even():
    n=int(e1.get())
    if(n%2):
        res="odd"
    else:
        res="even"
    l1=t.Label(w,text=res).grid(row=3,column=1)

L1=t.Label(w,text="Number").grid(row=0,column=0)
e1 = t.Entry(w)
e1.grid(row = 0, column = 1)
rb=t.Button(w,text='check',command=even).grid(row=1,column=1)
w.mainloop()
```



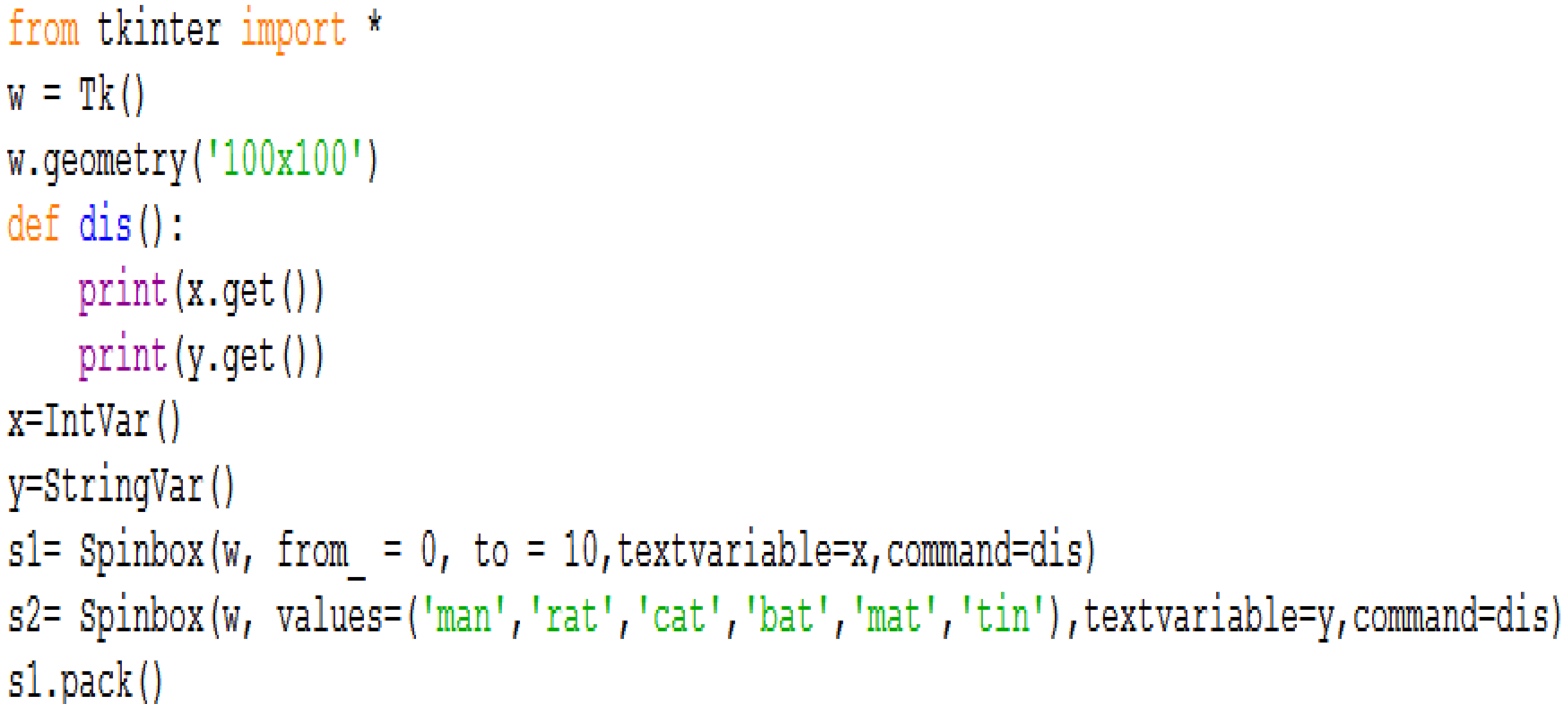
Number 5

odd

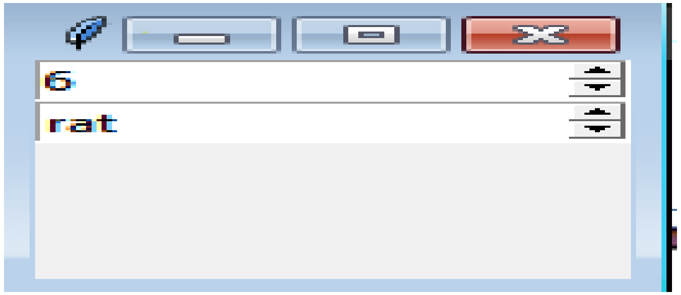
Spin box: It allows the user to select the values from a given set of values. The values may be a range of numbers or a fixed set of strings

S= Spinbox (master , from_=4,to=15,textvariable= var, command=fun,……..)

S= Spinbox (master , values=('a','b','c'),textvariable= var, command=fun,……..)



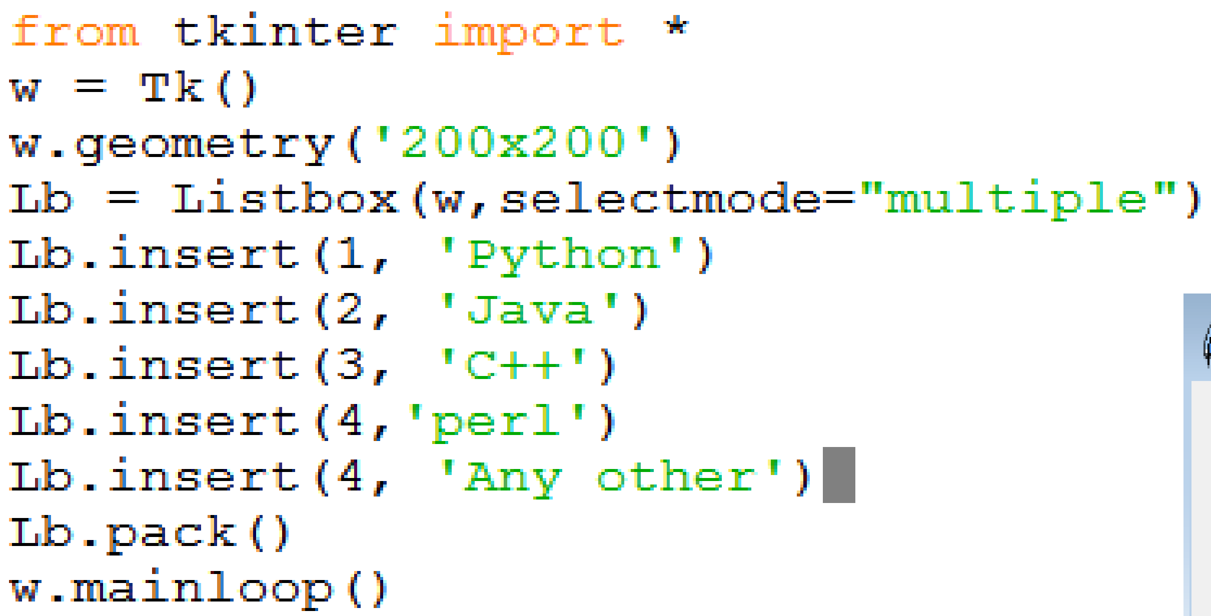
```
from tkinter import *
w = Tk()
w.geometry('100x100')
def dis():
    print(x.get())
    print(y.get())
x=IntVar()
y=StringVar()
s1= Spinbox(w, from_ = 0, to = 10,textvariable=x,command=dis)
s2= Spinbox(w, values=('man','rat','cat','bat','mat','tin'),textvariable=y,command=dis)
s1.pack()
s2.pack()
mainloop()
```



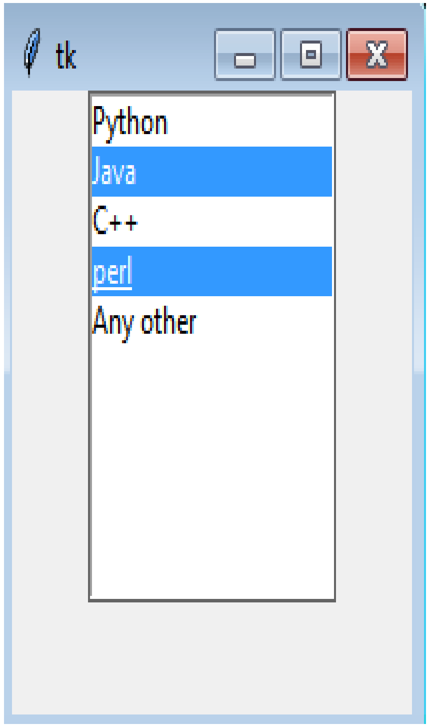
```
6
rat
```

List box:It displays a list of items in a box so that the user can select one or more items

l1= Listbox (master ,selectmode=“multiple”,.. …)



```
from tkinter import *
w = Tk()
w.geometry('200x200')
Lb = Listbox(w,selectmode="multiple")
Lb.insert(1, 'Python')
Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4,'perl')
Lb.insert(4, 'Any other')
Lb.pack()
w.mainloop()
```



```
Python
Java
C++
perl
Any other
```

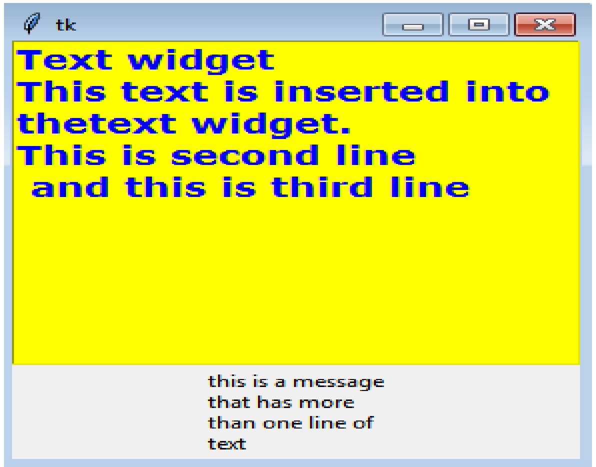
Message: It is used to display multiple lines of text

m=Message(master,text=str1,option=value1,…….)

Text : It is same as a label or message and used to display multiple lines of text in different colors and fonts.

T=Text(master,option=value,………)

```
from tkinter import *
w=Tk()
t=Text(w,width=20,height=10,font=("verdana",14,"bold"),
                    fg="blue",bg="yellow",wrap=WORD)
t.insert(END,"Text widget\nThis text is inserted into thetext widget.\
                      \nThis is second line\n and this is third line\n")
t.pack(sid=TOP)

m=Message(w,text='this is a message that has more than one line of text')
m.pack()
w.mainloop()
```



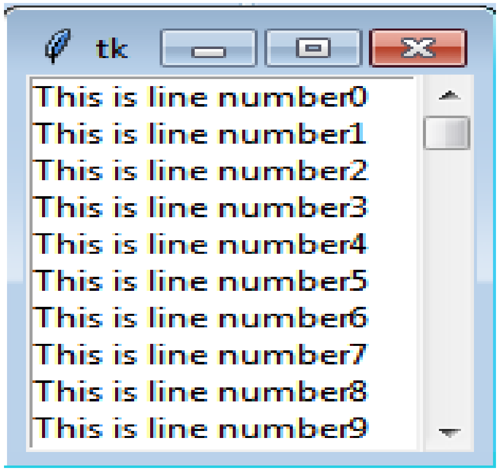
```
Text widget
This text is inserted into
thetext widget.
This is second line
 and this is third line
```

```
this is a message
that has more
than one line of
text
```

Scrollbar: The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

s=Scrollbar(master,option=value,……)

```
from tkinter import *
w = Tk()
s= Scrollbar(w)
s.pack(side=RIGHT,fill=Y)
mylist = Listbox(w, yscrollcommand = s.set )
for line in range(100):
   mylist.insert(END, 'This is line number' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
s.config( command = mylist.yview )
mainloop()
```

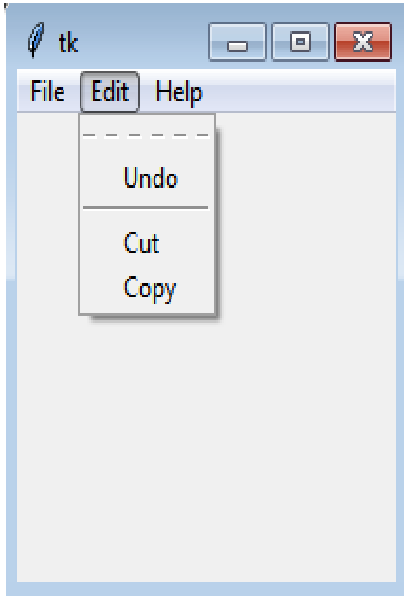


```
This is line number0
This is line number1
This is line number2
This is line number3
This is line number4
This is line number5
This is line number6
This is line number7
This is line number8
This is line number9
```

Menu: It is used to create all kinds of menus used by the application.

m=Menu(master,option=value,……..)

```
from tkinter import *
w = Tk()
menubar = Menu(w)
w.config(menu=menubar)
filemenu = Menu(menubar)
filemenu.add_command(label="New")
filemenu.add_command(label="Open")
filemenu.add_separator()
filemenu.add_command(label="Exit", command=w.quit)
menubar.add_cascade(label="File", menu=filemenu)
editmenu = Menu(menubar)
editmenu.add_command(label="Undo")
editmenu.add_separator()
editmenu.add_command(label="Cut")
editmenu.add_command(label="Copy")
menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu = Menu(menubar)
helpmenu.add_command(label="Help Index")
helpmenu.add_command(label="About...")
menubar.add_cascade(label="Help")
w.mainloop()
```


tk
File
Edit
Help
Undo
Cut
Copy