Computer Graphics Assignment-4

2D and 3D Transformations

Name-Ashish Goyal

Id-2016ucp1100

Batch-A (1, 2)

2D:

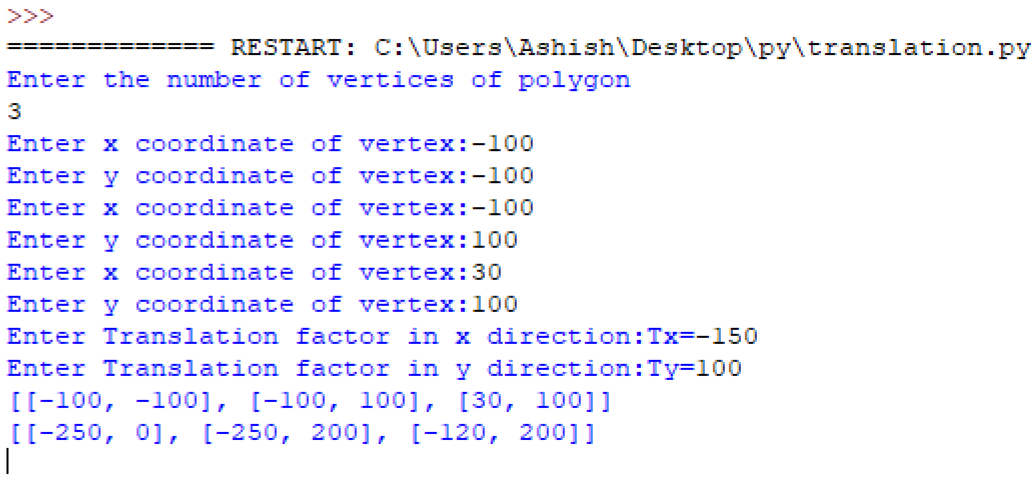
Translation

Code:

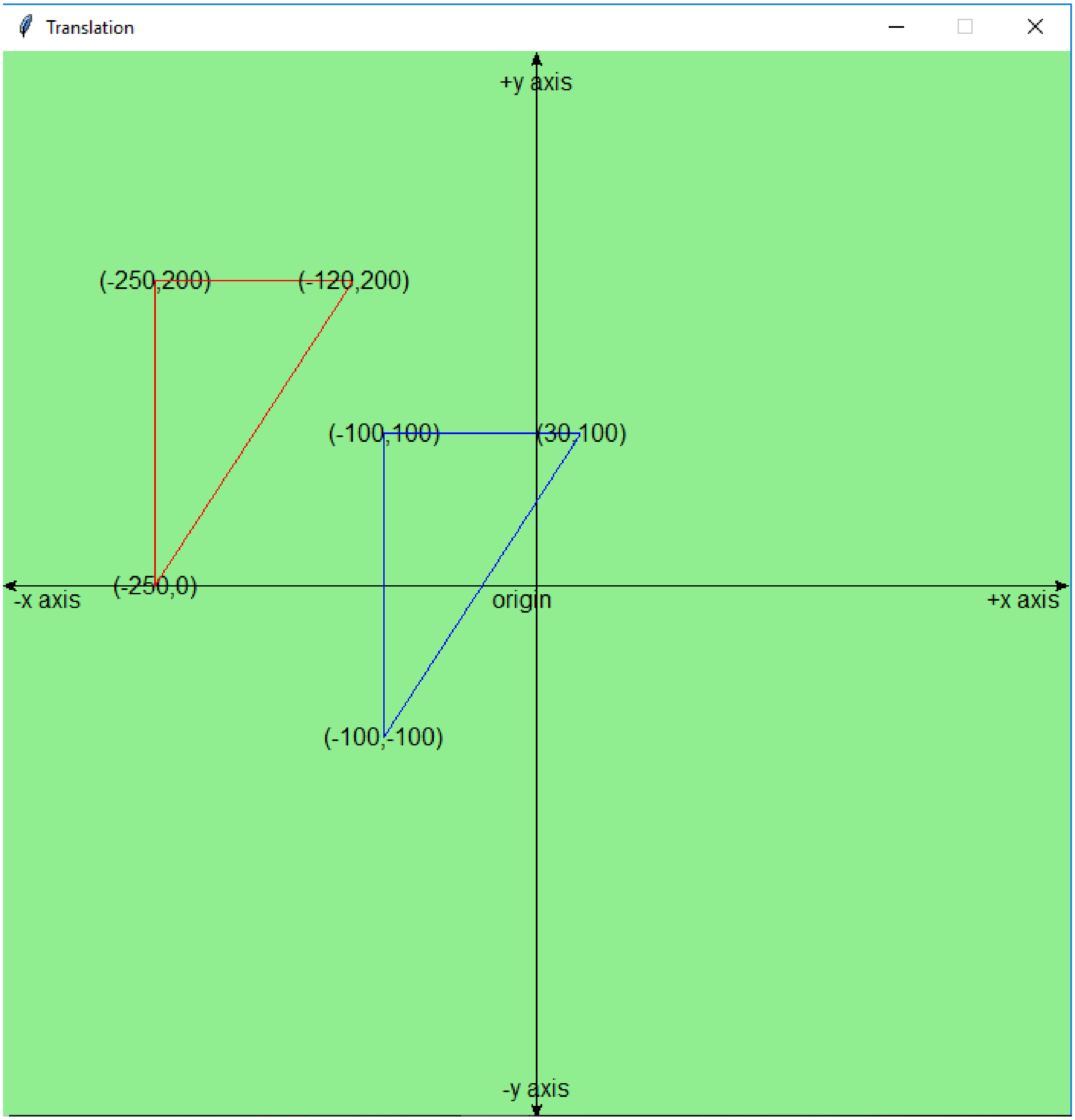
```
from graphics import *



coordinate = []
def translation(tx1,ty1):
	print(coordinate)
	for i in range(c):
		coordinate[i][0]=coordinate[i][0]+tx1
		coordinate[i][1]=coordinate[i][1]+ty1

	print(coordinate)
print("Enter the number of vertices of polygon")
c=int(input())

for i in range(c):
    x = int(input("Enter x coordinate of vertex:"))
    y = int(input("Enter y coordinate of vertex:"))
    point=[]
    point.append(x)
    point.append(y)
    coordinate.append(point)

tx = int(input("Enter Translation factor in x direction:Tx="))
ty = int(input("Enter Translation factor in y direction:Ty="))

win_obj=GraphWin("Translation",700,700) #set viewport size  700,700 are device coordinates
win_obj.setBackground("Light Green")
win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set
x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis
y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis

x_axis.setOutline("Black")
y_axis.setOutline("Black")
x_axis.setArrow('both')
y_axis.setArrow('both')
x_axis.draw(win_obj)
```

```
y_axis.draw(win_obj)


info_x=Text(Point(320,-10),"+x axis")
info_x.draw(win_obj)
info_nx=Text(Point(-320,-10),"-x axis")
info_nx.draw(win_obj)
info_y=Text(Point(0,330),"+y axis")
info_y.draw(win_obj)
info_ny=Text(Point(0,-330),"-y axis")
info_ny.draw(win_obj)


origin=Text(Point(-10,-10),"origin")
origin.draw(win_obj)




#previos
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setOutline("blue")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setOutline("blue")
line.draw(win_obj)


translation(tx,ty)

#after translation
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))

    line.setOutline("red")

    line.draw(win_obj)

x0 = coordinate[0][0]

y0 = coordinate[0][1]

x1 = coordinate[c-1][0]

y1 = coordinate[c-1][1]

#Point(x0,y0).draw(win_obj)

display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")

display.draw(win_obj)

line=Line(Point(x0,y0),Point(x1,y1))

line.setOutline("red")

line.draw(win_obj)



win_obj.getMouse()

win_obj.close()
```

Example1:



```
>>> 
============= RESTART: C:\Users\Ashish\Desktop\py\translation.py
Enter the number of vertices of polygon
3
Enter x coordinate of vertex:-100
Enter y coordinate of vertex:-100
Enter x coordinate of vertex:-100
Enter y coordinate of vertex:100
Enter x coordinate of vertex:30
Enter y coordinate of vertex:100
Enter Translation factor in x direction:Tx=-150
Enter Translation factor in y direction:Ty=100
[[-100, -100], [-100, 100], [30, 100]]
[[-250, 0], [-250, 200], [-120, 200]]
|
```



Scaling

Code:

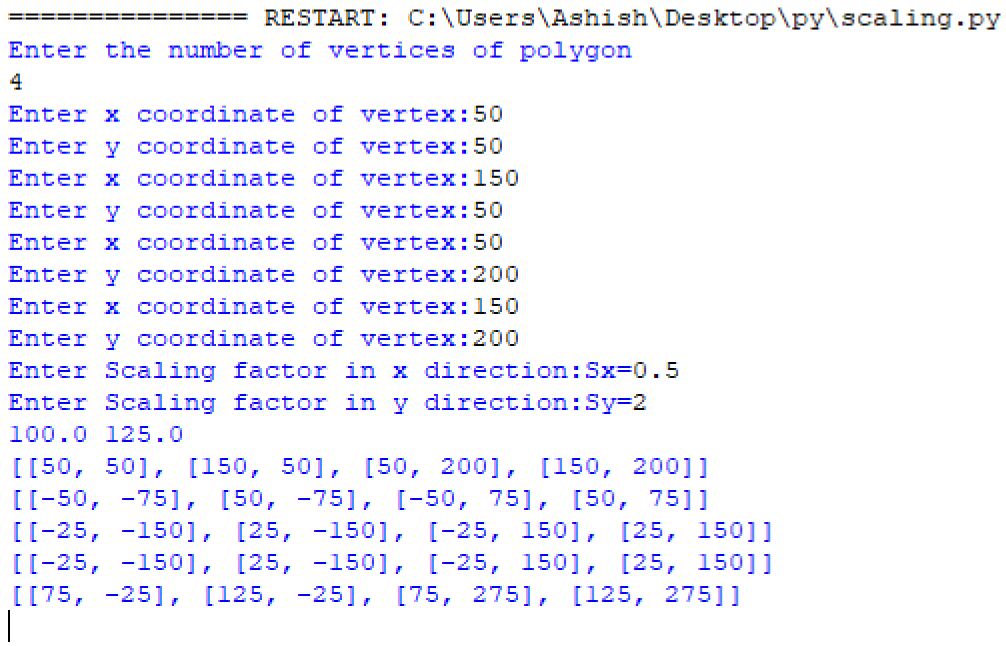
```
from graphics import *
```

```
def getcenter():
        global cx,cy
        for i in range(c):
                cx += coordinate[i][0]
                cy += coordinate[i][1]
        cx = cx/c
        cy = cy/c
def translation(tx1,ty1):
        print(coordinate)
        for i in range(c):
                coordinate[i][0]=(int)(coordinate[i][0]+tx1)
                coordinate[i][1]=(int)(coordinate[i][1]+ty1)

        print(coordinate)

def scaling(sx1,sy1):
        for i in range(c):
                coordinate[i][0]=(int)(coordinate[i][0]*sx1)
                coordinate[i][1]=(int)(coordinate[i][1]*sy1)
        print(coordinate)

print("Enter the number of vertices of polygon")
c=int(input())
coordinate = []
for i in range(c):
    x = int(input("Enter x coordinate of vertex:"))
    y = int(input("Enter y coordinate of vertex:"))
    point=[]
    point.append(x)
    point.append(y)
    coordinate.append(point)

sx = float(input("Enter Scaling factor in x direction:Sx="))
sy = float(input("Enter Scaling factor in y direction:Sy="))

win_obj=GraphWin("Scaling",700,700) #set viewport size  700,700 are device coordinates
win_obj.setBackground("Light Green")
win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set
x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis
y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis

x_axis.setOutline("Black")
y_axis.setOutline("Black")
x_axis.setArrow('both')
y_axis.setArrow('both')
x_axis.draw(win_obj)
y_axis.draw(win_obj)

info_x=Text(Point(320,-10),"+x axis")
info_x.draw(win_obj)
```

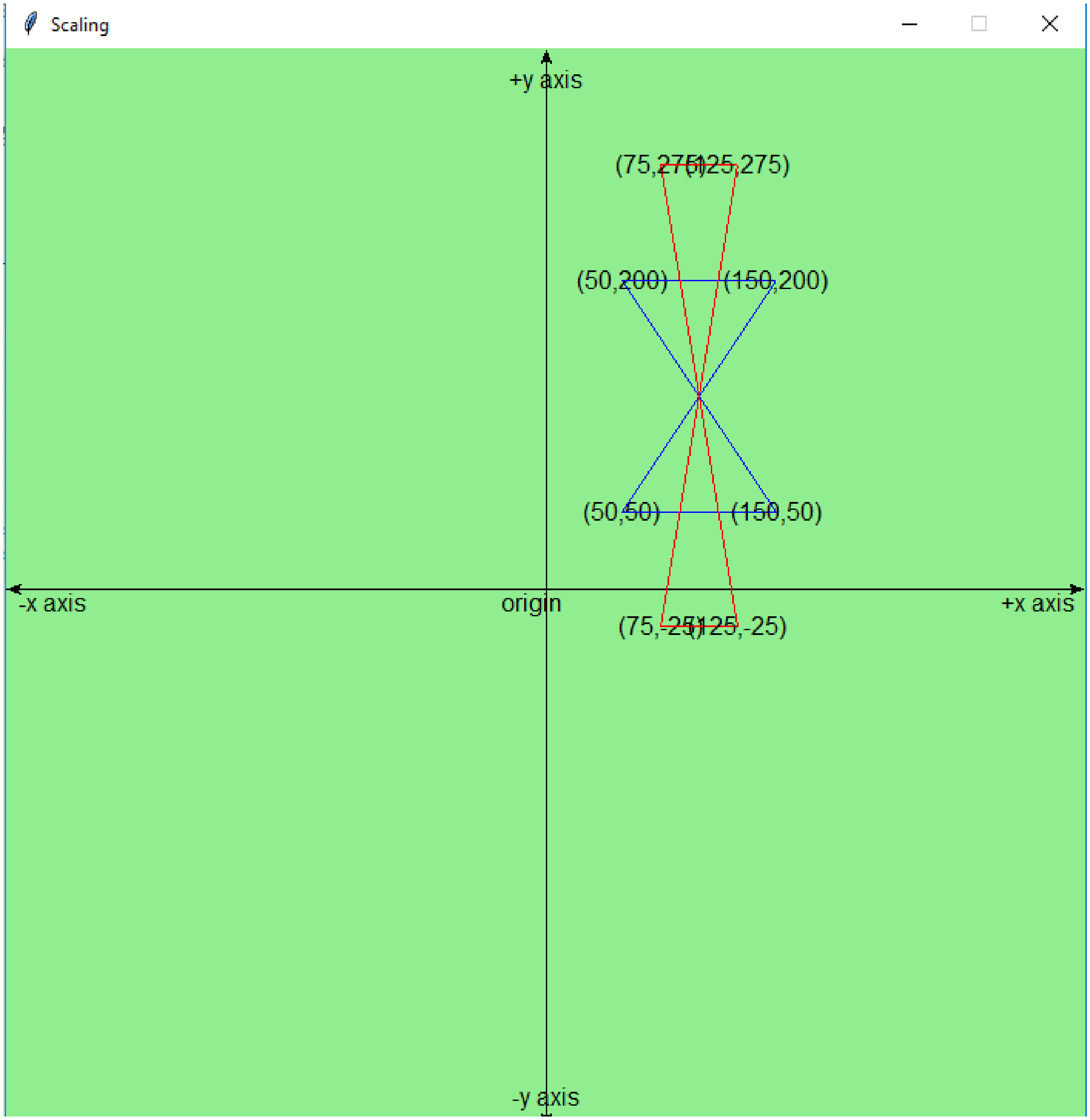
```
info_nx=Text(Point(-320,-10),"-x axis")

info_nx.draw(win_obj)

info_y=Text(Point(0,330),"+y axis")

info_y.draw(win_obj)

info_ny=Text(Point(0,-330),"-y axis")

info_ny.draw(win_obj)


origin=Text(Point(-10,-10),"origin")

origin.draw(win_obj)



#previous

for i in range(c-1):

    x0 = coordinate[i][0]

    y0 = coordinate[i][1]

    x1 = coordinate[i+1][0]

    y1 = coordinate[i+1][1]

    #Point(x0,y0).draw(win_obj)

    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")

    display.draw(win_obj)

    line=Line(Point(x0,y0),Point(x1,y1))

    line.setOutline("blue")

    line.draw(win_obj)

x0 = coordinate[0][0]

y0 = coordinate[0][1]
```

```
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setOutline("blue")
line.draw(win_obj)

cx = 0  #for x coordinate of center
cy = 0
getcenter()
print(cx,cy)
translation(-cx,-cy)
scaling(sx,sy)
translation(cx,cy)

#after scaling
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setOutline("red")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setOutline("red")
line.draw(win_obj)


win_obj.getMouse()
win_obj.close()
```

Example2:



```
=============== RESTART: C:\Users\Ashish\Desktop\py\scaling.py
Enter the number of vertices of polygon
4
Enter x coordinate of vertex:50
Enter y coordinate of vertex:50
Enter x coordinate of vertex:150
Enter y coordinate of vertex:50
Enter x coordinate of vertex:50
Enter y coordinate of vertex:200
Enter x coordinate of vertex:150
Enter y coordinate of vertex:200
Enter Scaling factor in x direction:Sx=0.5
Enter Scaling factor in y direction:Sy=2
100.0 125.0
[[50, 50], [150, 50], [50, 200], [150, 200]]
[[-50, -75], [50, -75], [-50, 75], [50, 75]]
[[-25, -150], [25, -150], [-25, 150], [25, 150]]
[[-25, -150], [25, -150], [-25, 150], [25, 150]]
[[75, -25], [125, -25], [75, 275], [125, 275]]
```



Rotation

Code:

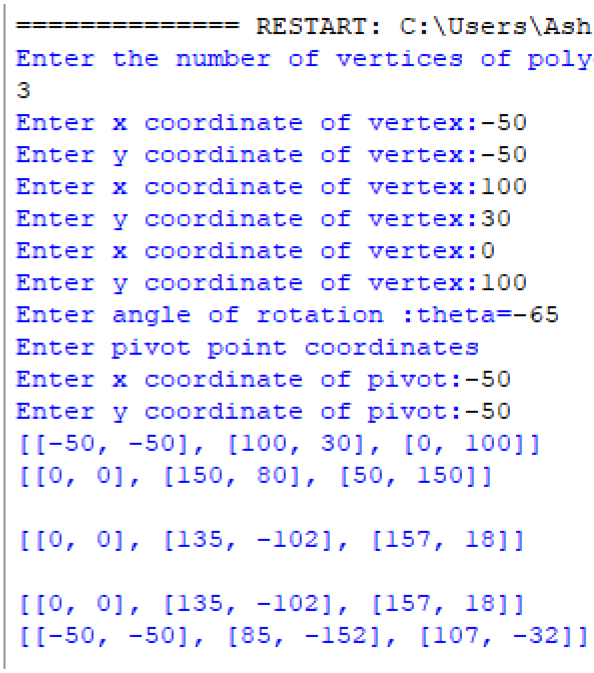
```
from graphics import *

import math
```

```
coordinate = []
def translation(tx1,ty1):
    print(coordinate)
    for i in range(c):
        coordinate[i][0]=(int)(coordinate[i][0]+tx1)
        coordinate[i][1]=(int)(coordinate[i][1]+ty1)

    print(coordinate)
    print()
def rotation(theta):
    for i in range(c):
        x0 =coordinate[i][0]
        y0=coordinate[i][1]
        coordinate[i][0]=(int)(x0*math.cos(math.radians(theta))-
y0*math.sin(math.radians(theta)))

    coordinate[i][1]=(int)(x0*math.sin(math.radians(theta))+y0*math.cos(math.radians(theta
)))

    print(coordinate)
    print()

print("Enter the number of vertices of polygon")
c=int(input())
```

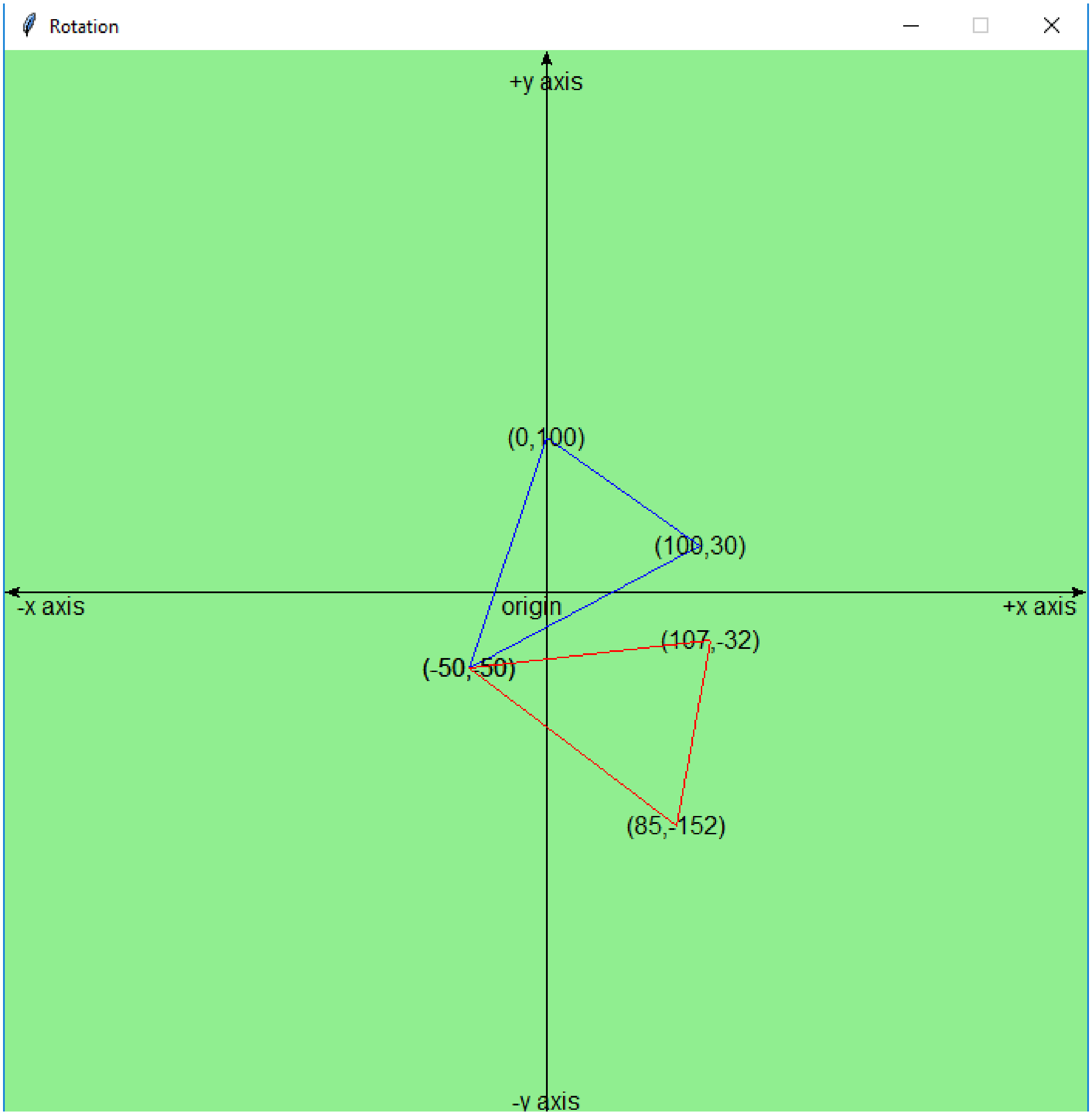
```
for i in range(c):
    x = int(input("Enter x coordinate of vertex:"))
    y = int(input("Enter y coordinate of vertex:"))
    point=[]
    point.append(x)
    point.append(y)
    coordinate.append(point)



theta= float(input("Enter angle of rotation :theta="))


print("Enter pivot point coordinates")
a = int(input("Enter x coordinate of pivot:"))
b = int(input("Enter y coordinate of pivot:"))


win_obj=GraphWin("Rotation",700,700) #set viewport size  700,700 are device coordinates
win_obj.setBackground("Light Green")
win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set
x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis
y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis


x_axis.setOutline("Black")
y_axis.setOutline("Black")
x_axis.setArrow('both')
y_axis.setArrow('both')
```

```
x_axis.draw(win_obj)

y_axis.draw(win_obj)


info_x=Text(Point(320,-10),"+x axis")

info_x.draw(win_obj)

info_nx=Text(Point(-320,-10),"-x axis")

info_nx.draw(win_obj)

info_y=Text(Point(0,330),"+y axis")

info_y.draw(win_obj)

info_ny=Text(Point(0,-330),"-y axis")

info_ny.draw(win_obj)


origin=Text(Point(-10,-10),"origin")

origin.draw(win_obj)



#previous

for i in range(c-1):

    x0 = coordinate[i][0]

    y0 = coordinate[i][1]

    x1 = coordinate[i+1][0]

    y1 = coordinate[i+1][1]

    #Point(x0,y0).draw(win_obj)

    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")

    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setOutline("blue")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setOutline("blue")
line.draw(win_obj)



translation(-1*a,-1*b)
rotation(theta)
translation(a,b)

#after rotation
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setOutline("red")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setOutline("red")
line.draw(win_obj)

win_obj.getMouse()
win_obj.close()
```

Example3:



```
=============== RESTART: C:\Users\Ashish\Desktop\py\rotation.py
Enter the number of vertices of polygon
3
Enter x coordinate of vertex:-50
Enter y coordinate of vertex:-50
Enter x coordinate of vertex:100
Enter y coordinate of vertex:30
Enter x coordinate of vertex:0
Enter y coordinate of vertex:100
Enter angle of rotation :theta=-65
Enter pivot point coordinates
Enter x coordinate of pivot:-50
Enter y coordinate of pivot:-50
[[-50, -50], [100, 30], [0, 100]]
[[0, 0], [150, 80], [50, 150]]

[[0, 0], [135, -102], [157, 18]]

[[0, 0], [135, -102], [157, 18]]
[[-50, -50], [85, -152], [107, -32]]
```



Shearing

Code:

```
from graphics import *

import math

coordinate = []
```

```
def translation(tx1,ty1):
        print(coordinate)
        for i in range(c):
                coordinate[i][0]=(int)(coordinate[i][0]+tx1)
                coordinate[i][1]=(int)(coordinate[i][1]+ty1)

        print(coordinate)
        print()

def rotation(theta):
        for i in range(c):
                x0 =coordinate[i][0]
                y0=coordinate[i][1]
                coordinate[i][0]=(int)(x0*math.cos(theta)-y0*math.sin(theta))
                coordinate[i][1]=(int)(x0*math.sin(theta)+y0*math.cos(theta))

        print(coordinate)
        print()


def shear(shx1,shy1):
      print(coordinate)
      for i in range(c):

           temp_x=coordinate[i][0]
```

```
                temp_y=coordinate[i][1]
                coordinate[i][0]=(int)(temp_x+shx1*temp_y)
                coordinate[i][1]=(int)(temp_y+shy1*temp_x)

        print(coordinate)



print("Enter the number of vertices of polygon")
c=int(input())

for i in range(c):
    x = int(input("Enter x coordinate of vertex:"))
    y = int(input("Enter y coordinate of vertex:"))
    point=[]
    point.append(x)
    point.append(y)
    coordinate.append(point)

print("enter 1 for shear in x-axis and 2 for shear in y axis and 3 for arbitrary line")
temp=(int)(input())



if temp==1:
            shx = float(input("Enter shear factor in x direction:shx="))
            shy=0
```

```
elif temp==2:
        shx=0
        shy = float(input("Enter shear factor in y direction:shy="))



win_obj=GraphWin("Shear",700,700) #set viewport size  700,700 are device coordinates
win_obj.setBackground("Light Green")
win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set
x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis
y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis

x_axis.setOutline("Black")
y_axis.setOutline("Black")
x_axis.setArrow('both')
y_axis.setArrow('both')
x_axis.draw(win_obj)
y_axis.draw(win_obj)

info_x=Text(Point(320,-10),"+x axis")
info_x.draw(win_obj)
info_nx=Text(Point(-320,-10),"-x axis")
info_nx.draw(win_obj)
info_y=Text(Point(0,330),"+y axis")
info_y.draw(win_obj)
info_ny=Text(Point(0,-330),"-y axis")
```

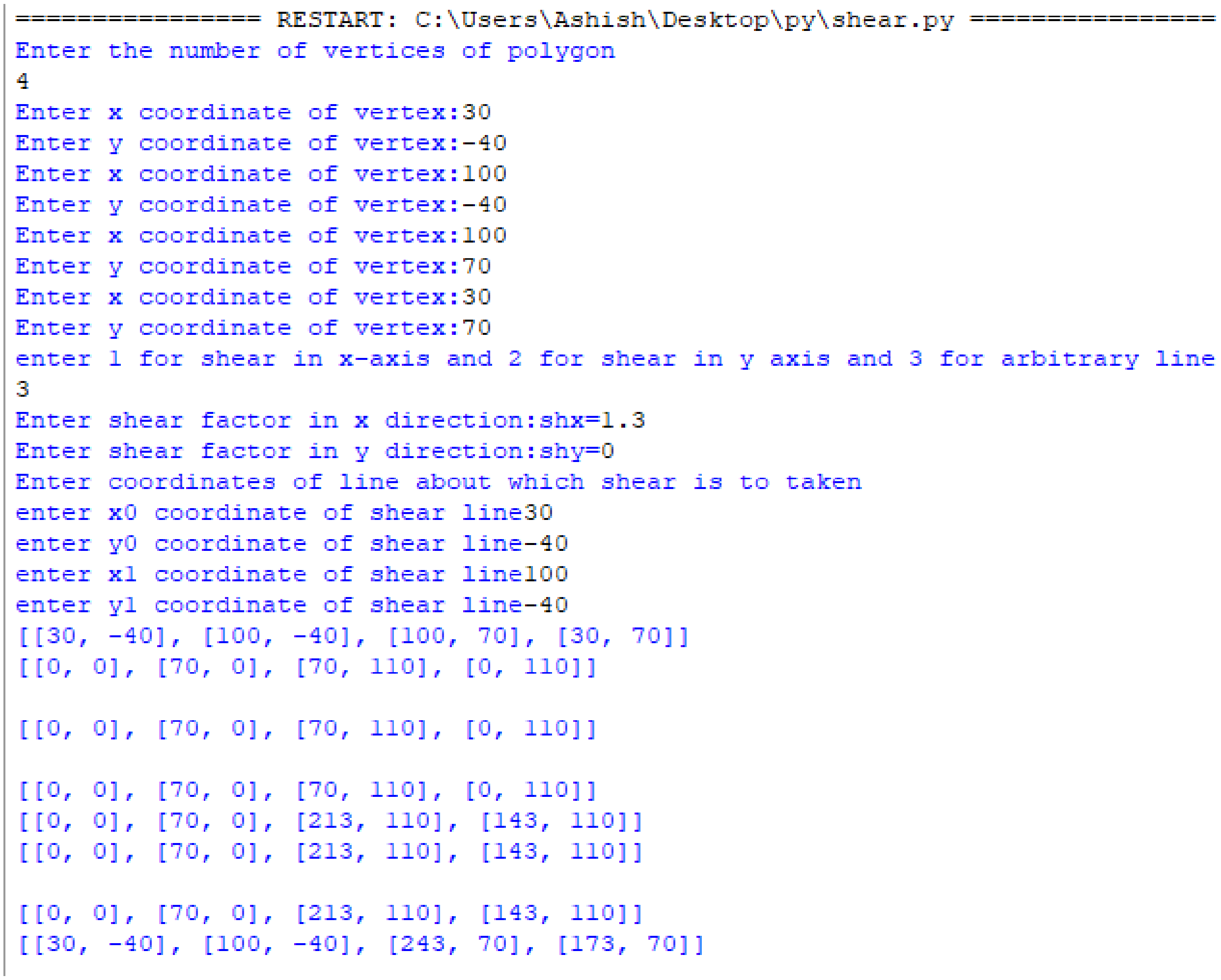
```
info_ny.draw(win_obj)

origin=Text(Point(-10,-10),"origin")
origin.draw(win_obj)


#previos
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setFill("blue")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
```

```
line=Line(Point(x0,y0),Point(x1,y1))
line.setFill("blue")
line.draw(win_obj)


if temp==1:
      shear(shx,shy)
elif temp==2:
      shear(shx,shy)
elif temp==3:
    shx = float(input("Enter shear factor in x direction:shx="))
    shy = float(input("Enter shear factor in y direction:shy="))
    print("Enter coordinates of line about which shear is to taken")
    a0=float(input("enter x0 coordinate of shear line"))
    b0=float(input("enter y0 coordinate of shear line"))
    a1=float(input("enter x1 coordinate of shear line"))
    b1=float(input("enter y1 coordinate of shear line"))


    if a1-a0 ==0:
        theta = (90*math.pi)/180
    elif b1-b0==0:
        theta=0
    else:
        theta=math.atan((b1-b0)/(a1-a0))
    translation(-a0,-b0)
```

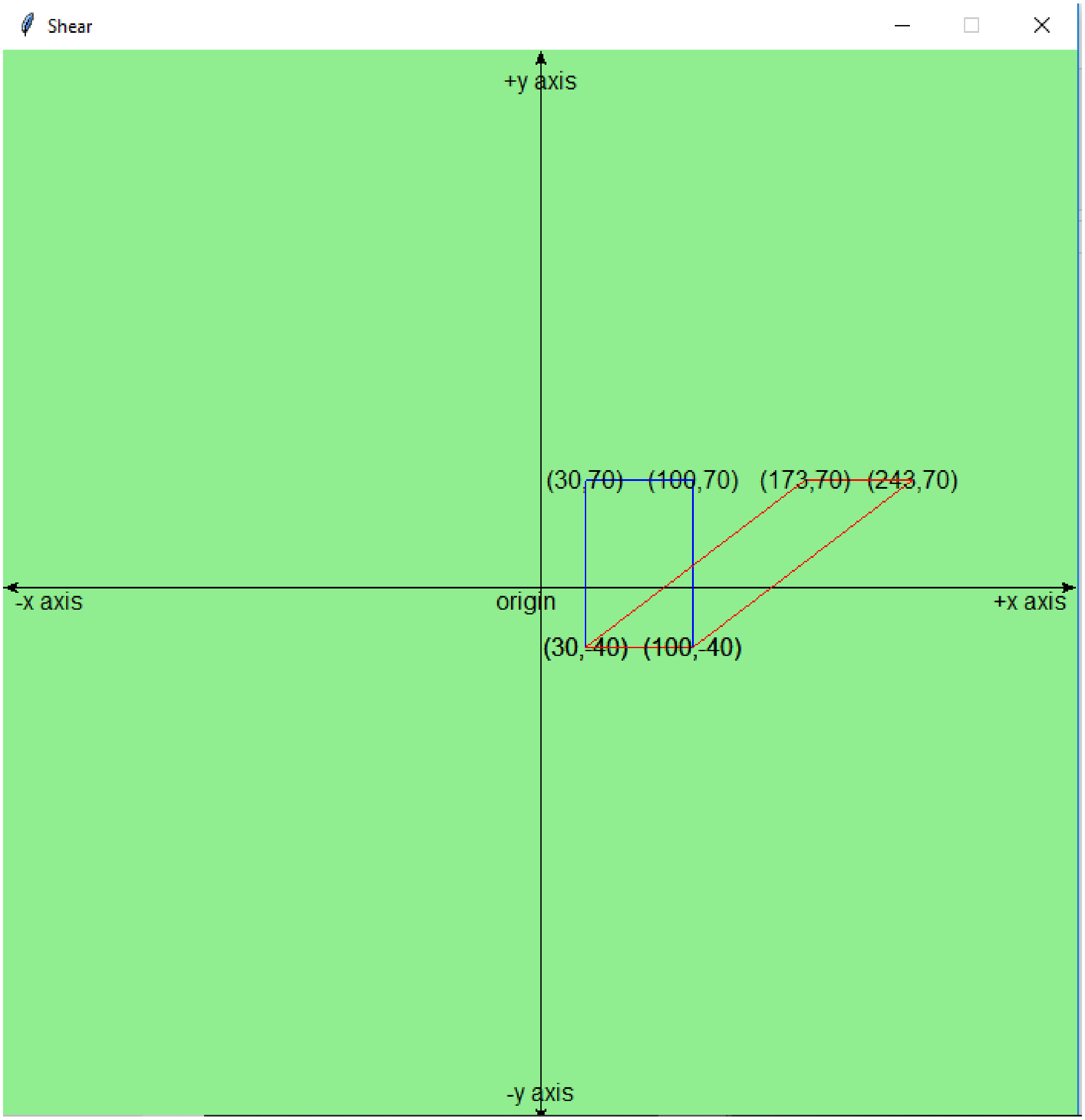
```
        if theta ==0:   #shear parallel to x axis
                rotation(-theta)
                shear(shx,0)
                rotation(theta)
        else:
                theta=math.radians(90)-theta
                rotation(theta)
                shear(0,shy)
                rotation(-theta)

        translation(a0,b0)


#after shear
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setFill("red")
    line.draw(win_obj)
```

```
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setFill("red")
line.draw(win_obj)

win_obj.getMouse()
win_obj.close()
```

Example4:



```
================ RESTART: C:\Users\Ashish\Desktop\py\shear.py ================
Enter the number of vertices of polygon
4
Enter x coordinate of vertex:30
Enter y coordinate of vertex:-40
Enter x coordinate of vertex:100
Enter y coordinate of vertex:-40
Enter x coordinate of vertex:100
Enter y coordinate of vertex:70
Enter x coordinate of vertex:30
Enter y coordinate of vertex:70
enter 1 for shear in x-axis and 2 for shear in y axis and 3 for arbitrary line
3
Enter shear factor in x direction:shx=1.3
Enter shear factor in y direction:shy=0
Enter coordinates of line about which shear is to taken
enter x0 coordinate of shear line30
enter y0 coordinate of shear line-40
enter x1 coordinate of shear line100
enter y1 coordinate of shear line-40
[[30, -40], [100, -40], [100, 70], [30, 70]]
[[0, 0], [70, 0], [70, 110], [0, 110]]

[[0, 0], [70, 0], [70, 110], [0, 110]]

[[0, 0], [70, 0], [70, 110], [0, 110]]
[[0, 0], [70, 0], [213, 110], [143, 110]]
[[0, 0], [70, 0], [213, 110], [143, 110]]

[[0, 0], [70, 0], [213, 110], [143, 110]]
[[30, -40], [100, -40], [243, 70], [173, 70]]
```



Reflection

Code:

```
from graphics import *

import math
```

```
coordinate = []
def translation(tx1,ty1):
    print(coordinate)
    for i in range(c):
        coordinate[i][0]=(int)(coordinate[i][0]+tx1)
        coordinate[i][1]=(int)(coordinate[i][1]+ty1)

    print(coordinate)
    print()

def rotation(theta):
    for i in range(c):
        x0 =coordinate[i][0]
        y0=coordinate[i][1]
        coordinate[i][0]=(int)(x0*math.cos(theta)-y0*math.sin(theta))
        coordinate[i][1]=(int)(x0*math.sin(theta)+y0*math.cos(theta))

    print(coordinate)
    print()

def ref_x():
    print(coordinate)
    for i in range(c):
        coordinate[i][0]=coordinate[i][0]
        coordinate[i][1]=coordinate[i][1]*-1
        print(coordinate)


def ref_y():
        print(coordinate)
        for i in range(c):
                coordinate[i][0]=coordinate[i][0]*-1
                coordinate[i][1]=coordinate[i][1]

        print(coordinate)


def ref_O():
        print(coordinate)
        for i in range(c):
                coordinate[i][0]=coordinate[i][0]*-1
                coordinate[i][1]=coordinate[i][1]*-1

        print(coordinate)


print("Enter the number of vertices of polygon")
c=int(input())


for i in range(c):
    x = int(input("Enter x coordinate of vertex:"))
    y = int(input("Enter y coordinate of vertex:"))
    point=[]
    point.append(x)
    point.append(y)
    coordinate.append(point)



print("enter 1:reflect in x :: 2:reflect in y :: 3:reflect in origin :: 4:arbitrary")
temp=(int)(input())

win_obj=GraphWin("reflect",700,700) #set viewport size  700,700 are device coordinates
win_obj.setBackground("Light Green")
win_obj.setCoords(-350,-350,350,350) #set window  use coordinates are set
x_axis=Line(Point(-350,0),Point(350,0))   #obj for x axis
y_axis=Line(Point(0,-350),Point(0,350))    #obj for y axis

x_axis.setOutline("Black")
y_axis.setOutline("Black")
x_axis.setArrow('both')
y_axis.setArrow('both')
x_axis.draw(win_obj)
y_axis.draw(win_obj)

info_x=Text(Point(320,-10),"+x axis")
info_x.draw(win_obj)
```

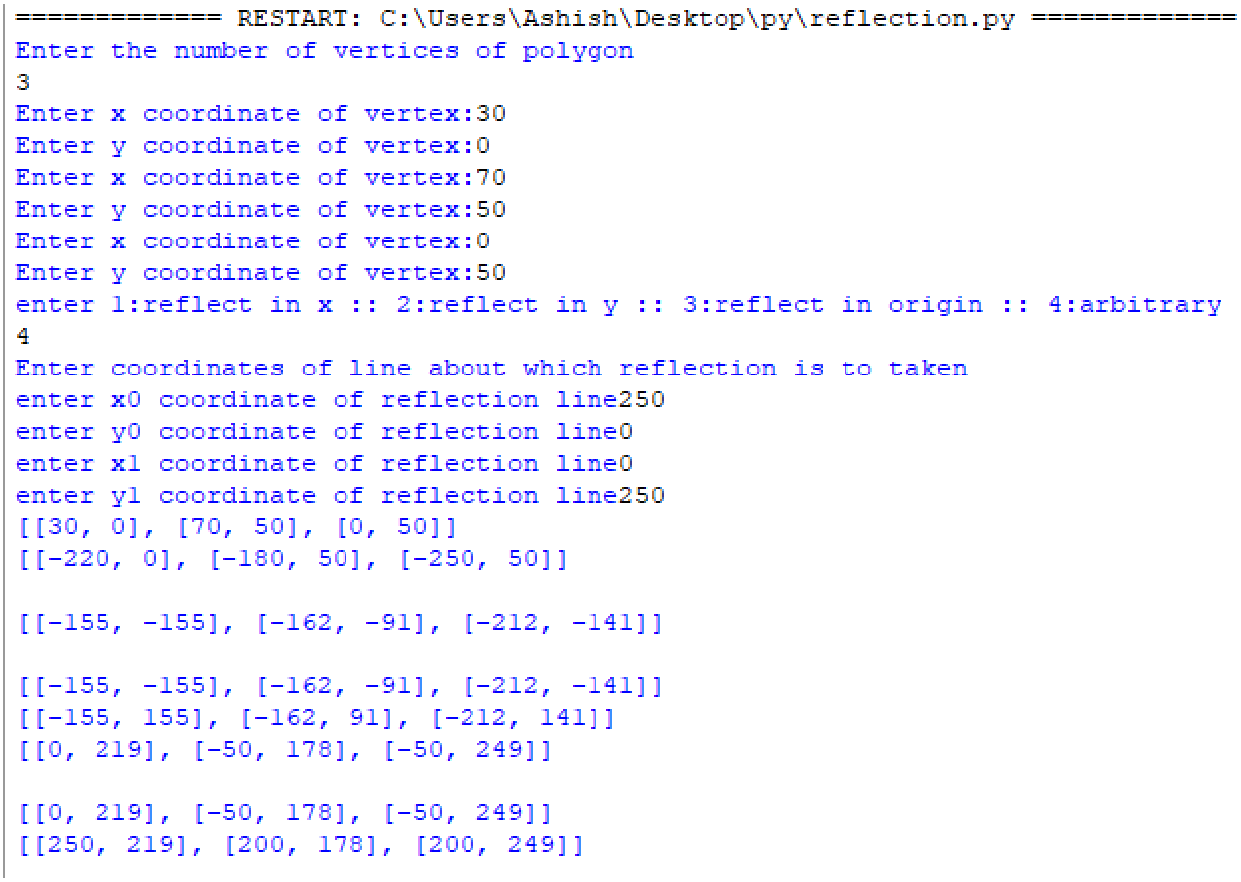
```
info_nx=Text(Point(-320,-10),"-x axis")
info_nx.draw(win_obj)
info_y=Text(Point(0,330),"+y axis")
info_y.draw(win_obj)
info_ny=Text(Point(0,-330),"-y axis")
info_ny.draw(win_obj)

origin=Text(Point(-10,-10),"origin")
origin.draw(win_obj)



#previos
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setFill("blue")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
```

```
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
#Point(x0,y0).draw(win_obj)
display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")
display.draw(win_obj)
line=Line(Point(x0,y0),Point(x1,y1))
line.setFill("blue")
line.draw(win_obj)


if temp==1:
        ref_x()
elif temp==2:
        ref_y()
elif temp==3:
        ref_O()
elif temp==4:
     print("Enter coordinates of line about which reflection is to taken")
     a0=float(input("enter x0 coordinate of reflection line"))
     b0=float(input("enter y0 coordinate of reflection line"))
     a1=float(input("enter x1 coordinate of reflection line"))
     b1=float(input("enter y1 coordinate of reflection line"))
     line=Line(Point(a0,b0),Point(a1,b1))
     line.setFill("black")
     line.draw(win_obj)
     if a1-a0 ==0:
```

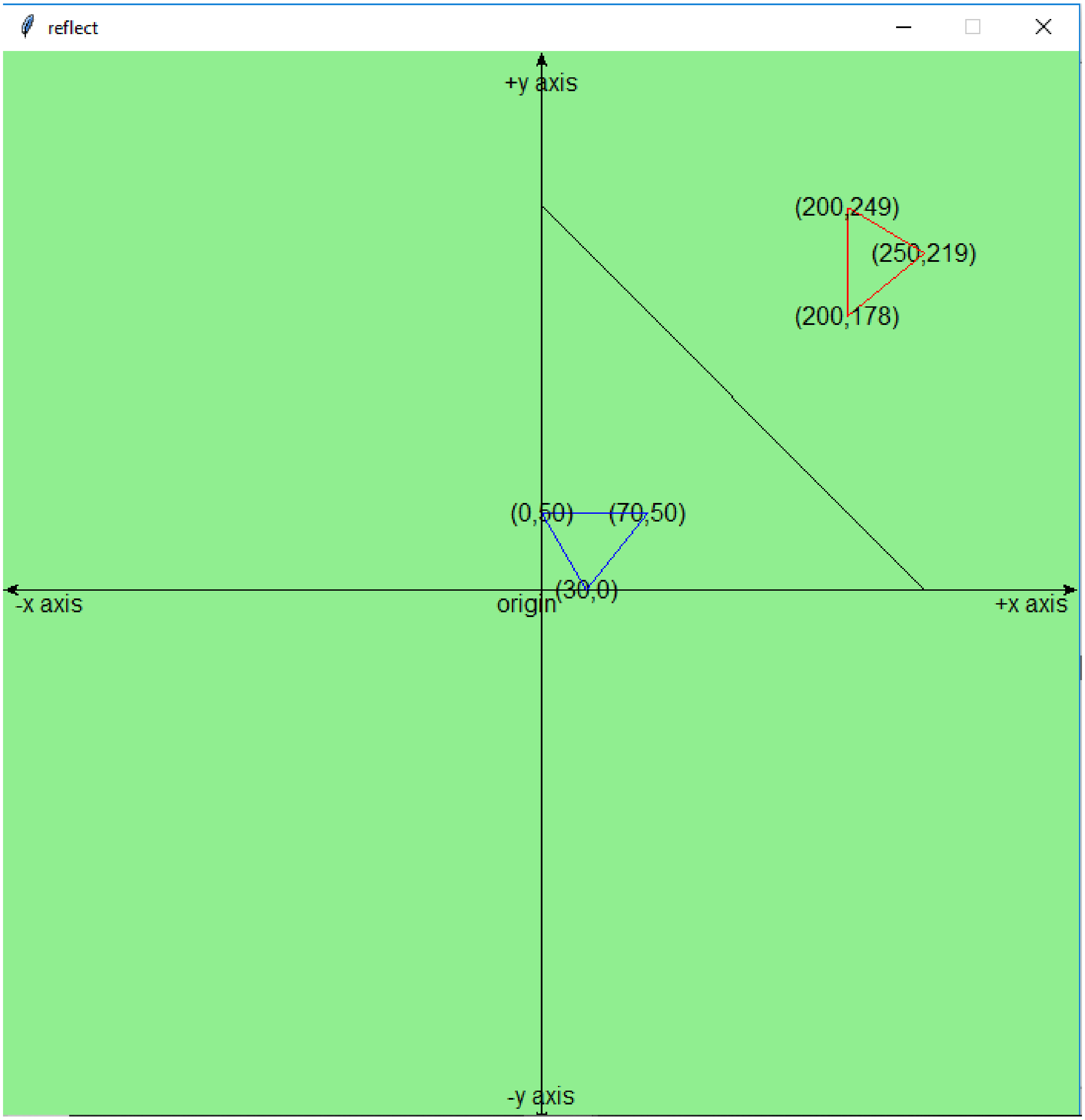
```
                theta = (90*math.pi)/180
        else:
                theta=math.atan((b1-b0)/(a1-a0))
        translation(-a0,-b0)
        rotation(-theta)
        ref_x()
        rotation(theta)
        translation(a0,b0)

#after reflect
for i in range(c-1):
    x0 = coordinate[i][0]
    y0 = coordinate[i][1]
    x1 = coordinate[i+1][0]
    y1 = coordinate[i+1][1]
    #Point(x0,y0).draw(win_obj)
    display = Text(Point(x0,y0),"("+str(x0)+","+str(y0)+")")
    display.draw(win_obj)
    line=Line(Point(x0,y0),Point(x1,y1))
    line.setFill("red")
    line.draw(win_obj)
x0 = coordinate[0][0]
y0 = coordinate[0][1]
x1 = coordinate[c-1][0]
y1 = coordinate[c-1][1]
```

```
#Point(x0,y0).draw(win_obj)

display = Text(Point(x1,y1),"("+str(x1)+","+str(y1)+")")

display.draw(win_obj)

line=Line(Point(x0,y0),Point(x1,y1))

line.setFill("red")

line.draw(win_obj)


win_obj.getMouse()

win_obj.close()
```

Example5:



```
============= RESTART: C:\Users\Ashish\Desktop\py\reflection.py =============
Enter the number of vertices of polygon
3
Enter x coordinate of vertex:30
Enter y coordinate of vertex:0
Enter x coordinate of vertex:70
Enter y coordinate of vertex:50
Enter x coordinate of vertex:0
Enter y coordinate of vertex:50
enter 1:reflect in x :: 2:reflect in y :: 3:reflect in origin :: 4:arbitrary
4
Enter coordinates of line about which reflection is to taken
enter x0 coordinate of reflection line250
enter y0 coordinate of reflection line0
enter x1 coordinate of reflection line0
enter y1 coordinate of reflection line250
[[30, 0], [70, 50], [0, 50]]
[[-220, 0], [-180, 50], [-250, 50]]

[[-155, -155], [-162, -91], [-212, -141]]

[[-155, -155], [-162, -91], [-212, -141]]
[[-155, 155], [-162, 91], [-212, 141]]
[[0, 219], [-50, 178], [-50, 249]]

[[0, 219], [-50, 178], [-50, 249]]
[[250, 219], [200, 178], [200, 249]]
```


reflect
+y axis
(200,249)
(250,219)
(200,178)
(0,50)
(70,50)
(30,0)
-x axis
origin
+x axis
-y axis

3D:

Single Code for translation, scaling, rotation in 3D:-

```
import numpy as np

import math

import time

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D



def Trans(q,tx,ty,tz,centerx,centery,centerz):

    shift=q.tolist()

    for i in range(len(shift)):

        shift[i][0]=shift[i][0]-centerx

        shift[i][1]=shift[i][1]-centery

        shift[i][2]=shift[i][2]-centerz

    q=np.array(shift)

    t=np.array([[1,0,0,0],[0,1,0,0],[0,0,1,0],[tx,ty,tz,1]])

    tr=q.dot(t)

    l=tr.tolist()

    for i in range(len(l)):

        l[i][0]=l[i][0]+centerx

        l[i][1]=l[i][1]+centery

        l[i][2]=l[i][2]+centerz
    tr=np.array(l)
    return tr

def Scale(q,sx,sy,sz,centerx,centery,centerz):
    shift=q.tolist()
    for i in range(len(shift)):
        shift[i][0]=shift[i][0]-centerx
        shift[i][1]=shift[i][1]-centery
        shift[i][2]=shift[i][2]-centerz
    q=np.array(shift)
    s=np.array([[sx,0,0,0],[0,sy,0,0],[0,0,sz,0],[0,0,0,1]])
    sc=q.dot(s)
    l=sc.tolist()
    for i in range(len(l)):
        l[i][0]=l[i][0]+centerx
        l[i][1]=l[i][1]+centery
        l[i][2]=l[i][2]+centerz
    sc=np.array(l)
    return sc

def Rotate(q,theta,centerx,centery,centerz):
    shift=q.tolist()
    ch=int(input("enter 1: x-axis 2: y-axis 3: z-axis "))
    for i in range(len(shift)):
        shift[i][0]=shift[i][0]-centerx
        shift[i][1]=shift[i][1]-centery
        shift[i][2]=shift[i][2]-centerz
    q=np.array(shift)
    if(ch == 3):
        s=np.array([[math.cos(theta),math.sin(theta),0,0],[(-1*math.sin(theta)),math.cos(theta),0,0],[0,0,1,0],[0,0,0,1]])
    elif(ch == 1):
        s=np.array([[1,0,0,0],[0,math.cos(theta),math.sin(theta),0],[0,(-1*math.sin(theta)),math.cos(theta),0],[0,0,0,1]])
    elif(ch == 2):
        s=np.array([[math.cos(theta),0,(-1*math.sin(theta)),0],[0,1,0,0],[(math.sin(theta)),0,math.cos(theta),0],[0,0,0,1]])
    else:
        print("you enter incorrect input");
    sc=q.dot(s)
    l=sc.tolist()
    for i in range(len(l)):
        l[i][0]=l[i][0]+centerx
        l[i][1]=l[i][1]+centery
        l[i][2]=l[i][2]+centerz
    sc=np.array(l)
    return sc
```

```
def RotateAA(q,theta,centerx,centery,centerz,aax1,aay1,aaz1,A,B,C):

        V=math.sqrt((B*B+C*C))

        L=math.sqrt((A*A+B*B+C*C))

        shift=q.tolist()

        c=C/V

        s=B/V

        c1=A/L

        s1=V/L

        for i in range(len(shift)):

                shift[i][0]=shift[i][0]-centerx

                shift[i][1]=shift[i][1]-centery

                shift[i][2]=shift[i][2]-centerz

        q=np.array(shift)

        ss=np.array([[1,0,0,(-1*aax1)],[0,1,0,(-1*aay1)],[0,0,1,(-1*aaz1)],[0,0,0,1]])
#Translation to pass from origin

        sc=q.dot(ss)

        ss=np.array([[1,0,0,0],[0,c,s,0],[0,(-1*s),c,0],[0,0,0,1]]) #To bring in x-z plane

        sc=sc.dot(ss)

        ss=np.array([[s1,0,c1,0],[0,1,0,0],[(-1*c1),0,s1,0],[0,0,0,1]]) #To make it z-axis

        sc=sc.dot(ss)

        ss=np.array([[math.cos(theta),math.sin(theta),0,0],[(-
1*math.sin(theta)),math.cos(theta),0,0],[0,0,1,0],[0,0,0,1]])#Actual Rotation
```

```
    sc=sc.dot(ss)

    ss=np.array([[s1,0,(-1*c1),0],[0,1,0,0],[(c1),0,s1,0],[0,0,0,1]]) #Inverse of making it z-axis

    sc=sc.dot(ss)

    ss=np.array([[1,0,0,0],[0,c,(-1*s),0],[0,(s),c,0],[0,0,0,1]]) #Inverse of bringing in x-z plane

    sc=sc.dot(ss)

    ss=np.array([[1,0,0,(aax1)],[0,1,0,(aay1)],[0,0,1,(aaz1)],[0,0,0,1]]) # Inverse of Translation

    sc=sc.dot(ss)

    l=sc.tolist()

    for i in range(len(l)):

        l[i][0]=l[i][0]+centerx

        l[i][1]=l[i][1]+centery

        l[i][2]=l[i][2]+centerz

    sc=np.array(l)

    n = len(polygon)

    for i in range(0,n-1):

        ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1], polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])

    ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1], polygon[0][1]],[polygon[n-1][2], polygon[0][2]])

    plt.show()

    return sc
```

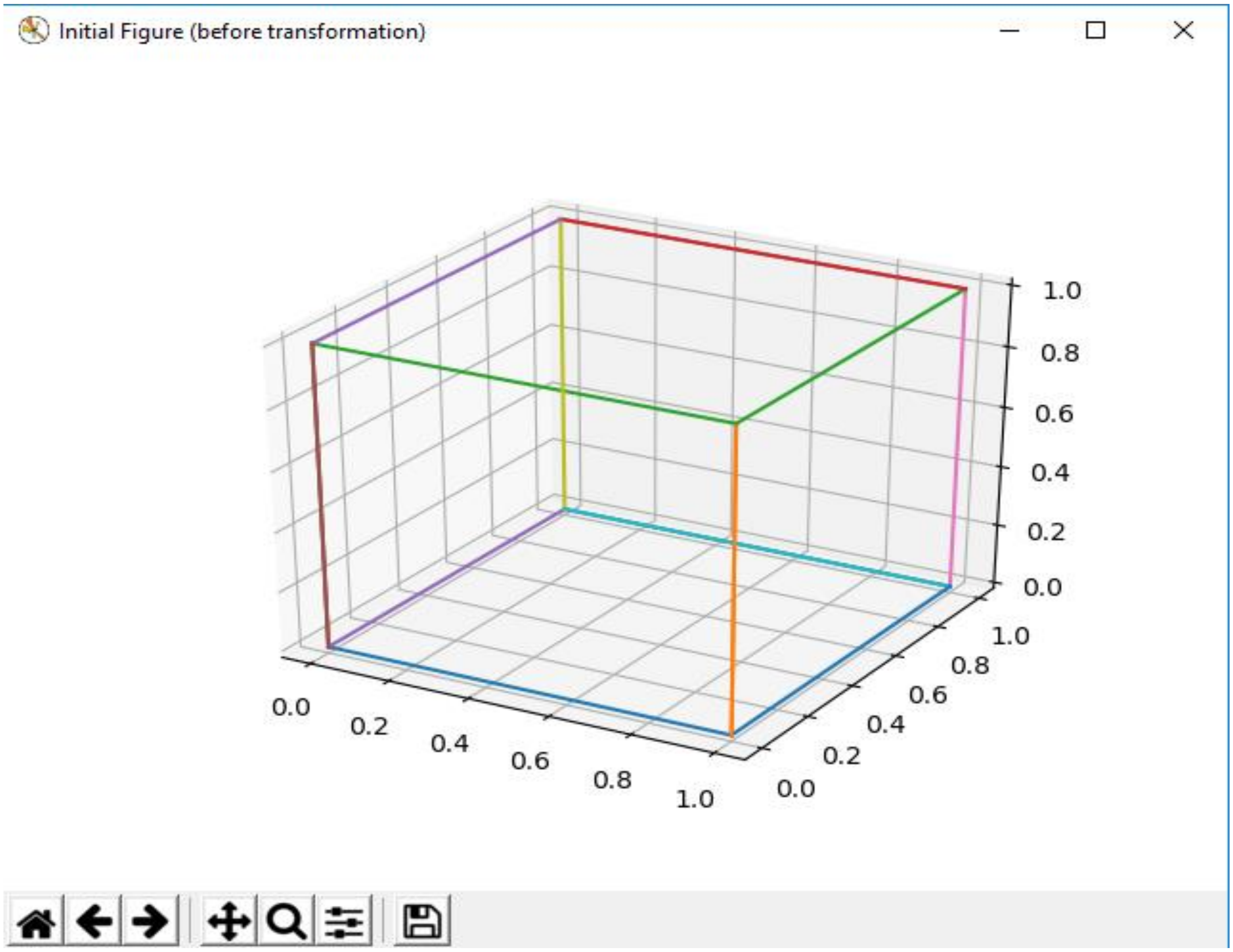
```
polygon =
[[0,0,0,1],[1,0,0,1],[1,0,1,1],[0,0,1,1],[0,0,0,1],[0,1,0,1],[1,1,0,1],[1,1,1,1],[0,1,1,1],[0,1,0,1],[1,1,
0,1],[1,0,0,1],[1,0,1,1],[1,1,1,1],[0,1,1,1],[0,0,1,1]]
n2=8
for kj in range(0,0):
    x11=int(input("x"+str(kj+1)+" "));
    y11=int(input("y"+str(kj+1)+" "));
    z11=int(input("z"+str(kj+1)+" "));
    polygon.append([x11,y11,z11,1])
fig = plt.figure("Initial Figure (before transformation)")
ax = fig.add_subplot(111, projection='3d')
n = len(polygon)
for i in range(0,n-1):
   ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1],
polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])
ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1], polygon[0][1]],[polygon[n-
1][2], polygon[0][2]])
plt.show()

xm=[]
ym=[]
zm=[]
for i in range(len(polygon)):
    xm.append(polygon[i][0])
    ym.append(polygon[i][1])
```

```
        zm.append(polygon[i][2])


centerx=sum(xm[:-1])/n2

centery=sum(ym[:-1])/n2

centerz=sum(zm[:-1])/n2


polyarr=np.array(polygon)

traask=int(input("enter 1 for translation and  0 for not"));

if(traask == 1):

    tx=int(input("enter Tx "))

    ty=int(input("enter Ty "))

    tz=int(input("enter Tz "))

    polyarr=Trans(polyarr,tx,ty,tz,centerx,centery,centerz)

    polygon=polyarr.tolist()

    fig = plt.figure("After Translation")

    ax = fig.add_subplot(111, projection='3d')

    n = len(polygon)

    for i in range(0,n-1):

        ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1], 
polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])

    ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1], 
polygon[0][1]],[polygon[n-1][2], polygon[0][2]])

    plt.show()


scaask=int(input("enter 1 for scaling and 0 for not"));
```

```
if(scaask == 1):
    sx=int(input("enter Sx "))
    sy=int(input("enter Sy "))
    sz=int(input("enter Sz "))
    polyarr=Scale(polyarr,sx,sy,sz,centerx,centery,centerz)
    polygon=polyarr.tolist()
    fig = plt.figure("After Scaling")
    ax = fig.add_subplot(111, projection='3d')
    n = len(polygon)
    for i in range(0,n-1):
        ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1], polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])
    ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1], polygon[0][1]],[polygon[n-1][2], polygon[0][2]])
    plt.show()



rotask=int(input("enter 1 for rotation and 0 for not"));
if(rotask == 1):
    theta=int(input("enter angle of rotation "))
    theta=(theta*math.pi)/180;
    polyarr=Rotate(polyarr,theta,centerx,centery,centerz)
    polygon=polyarr.tolist()
    fig = plt.figure("After Rotation")
    ax = fig.add_subplot(111, projection='3d')
    n = len(polygon)

    for i in range(0,n-1):

        ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1],
polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])

    ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1],
polygon[0][1]],[polygon[n-1][2], polygon[0][2]])

    plt.show()



rotaask=int(input("enter 1 for rotation about arbitrary axis and 0 for not"));

if(rotaask == 1):

    aax1=int(input("enter x1 of arbitray axis "))

    aay1=int(input("enter y1 of arbitray axis "))

    aaz1=int(input("enter z1 of arbitray axis "))

    A=int(input("enter x direction ratio "))

    B=int(input("enter y direction ratio "))

    C=int(input("enter z direction ratio "))

    theta=int(input("enter angle of rotation: "))


    theta=(theta*math.pi)/180

    polyarr=RotateAA(polyarr,theta,centerx,centery,centerz,aax1,aay1,aaz1,A,B,C)

    polygon=polyarr.tolist()

    fig = plt.figure("After Rotation About Axis Passing Through
("+str(aax1)+","+str(aay1)+","+str(aaz1)+") and dirction Ratios
("+str(A)+","+str(B)+","+str(C)+")")
```

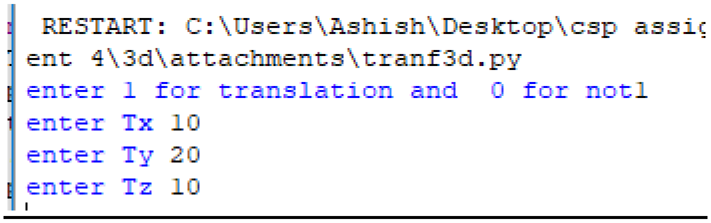
```
    ax = fig.add_subplot(111, projection='3d')


    n = len(polygon)

    for i in range(0,n-1):

        ax.plot([polygon[i][0], polygon[i+1][0]], [polygon[i][1], polygon[i+1][1]],[polygon[i][2], polygon[i+1][2]])

    ax.plot([polygon[n-1][0], polygon[0][0]], [polygon[n-1][1], polygon[0][1]],[polygon[n-1][2], polygon[0][2]])

    plt.show()
```

Before Transformation

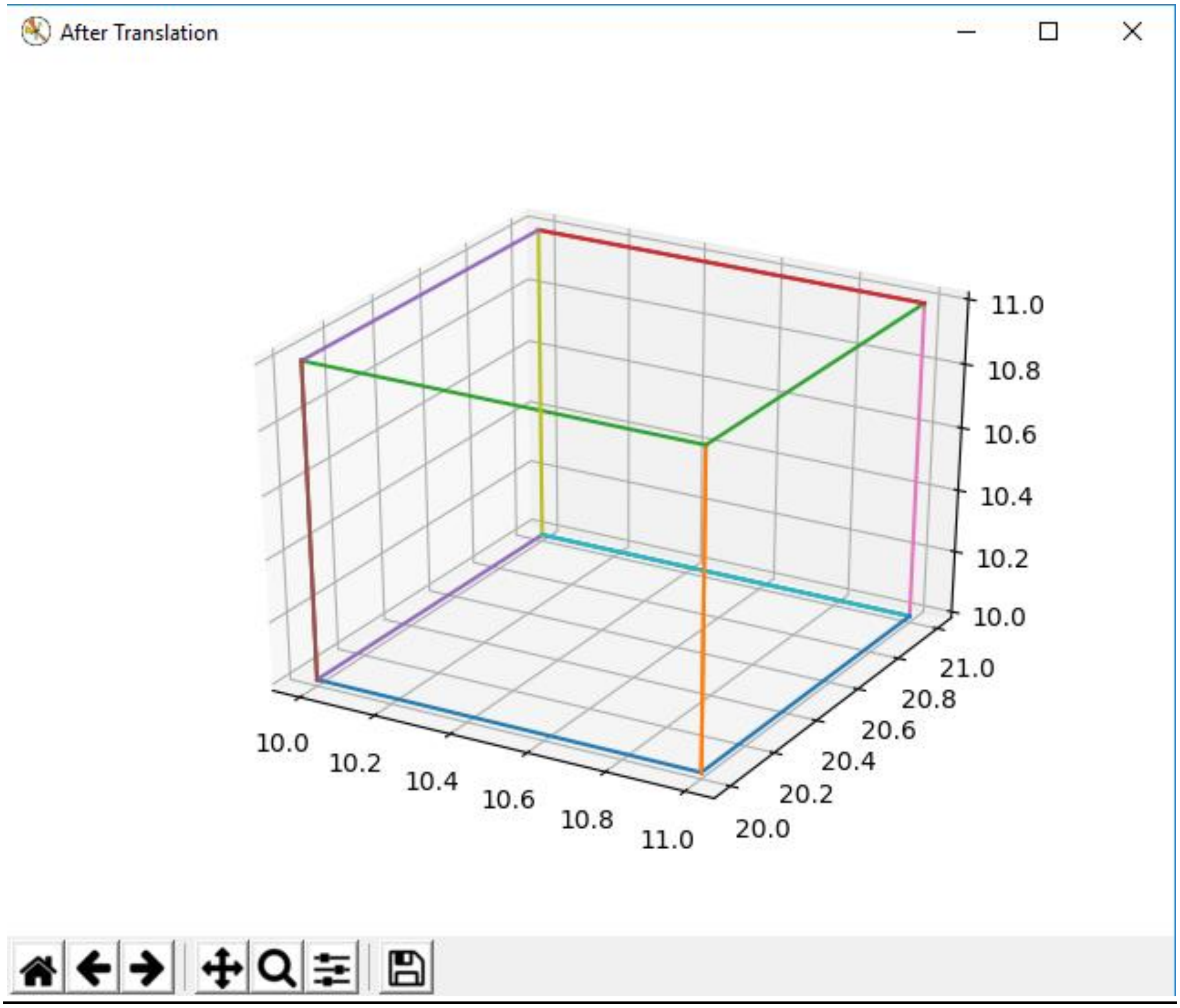


Translation

Example6:

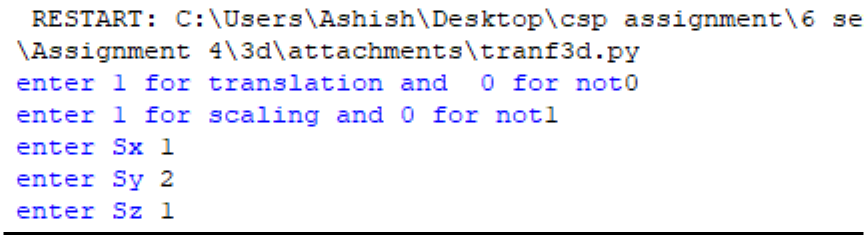


```
 RESTART: C:\Users\Ashish\Desktop\csp assig
ent 4\3d\attachments\tranf3d.py
enter 1 for translation and  0 for notl
enter Tx 10
enter Ty 20
enter Tz 10
```

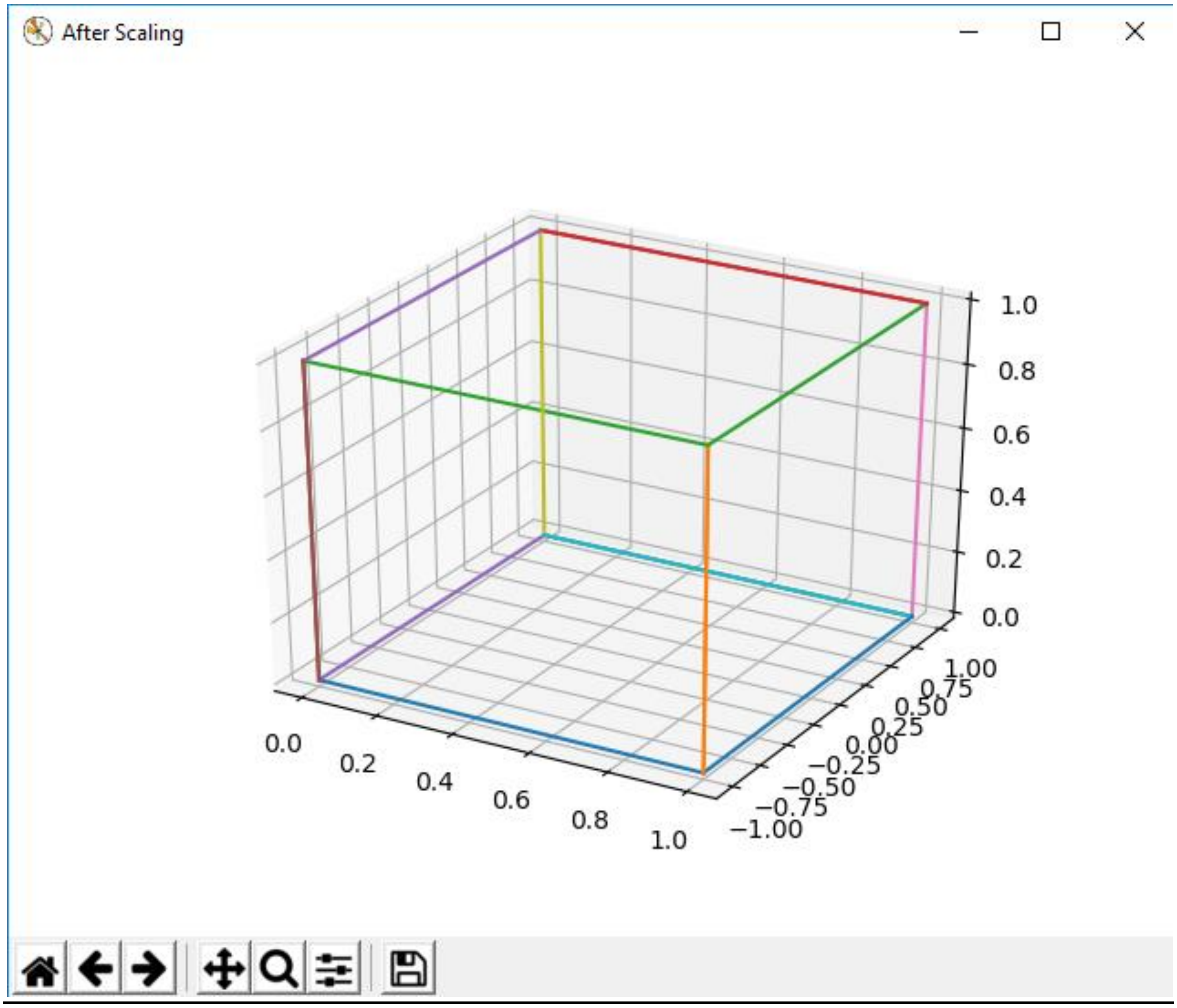


Scaling

Example7:

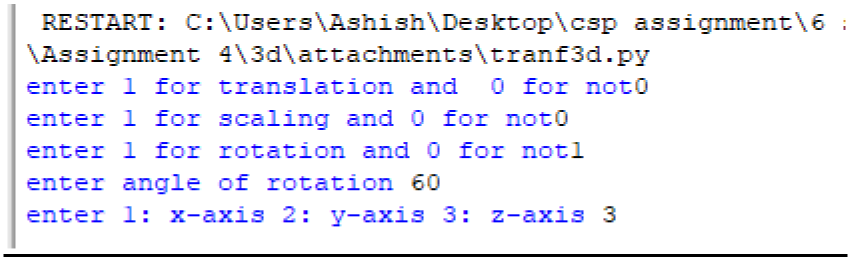


```
 RESTART: C:\Users\Ashish\Desktop\csp assignment\6 se
\Assignment 4\3d\attachments\tranf3d.py
enter 1 for translation and  0 for not0
enter 1 for scaling and 0 for not1
enter Sx 1
enter Sy 2
enter Sz 1
```

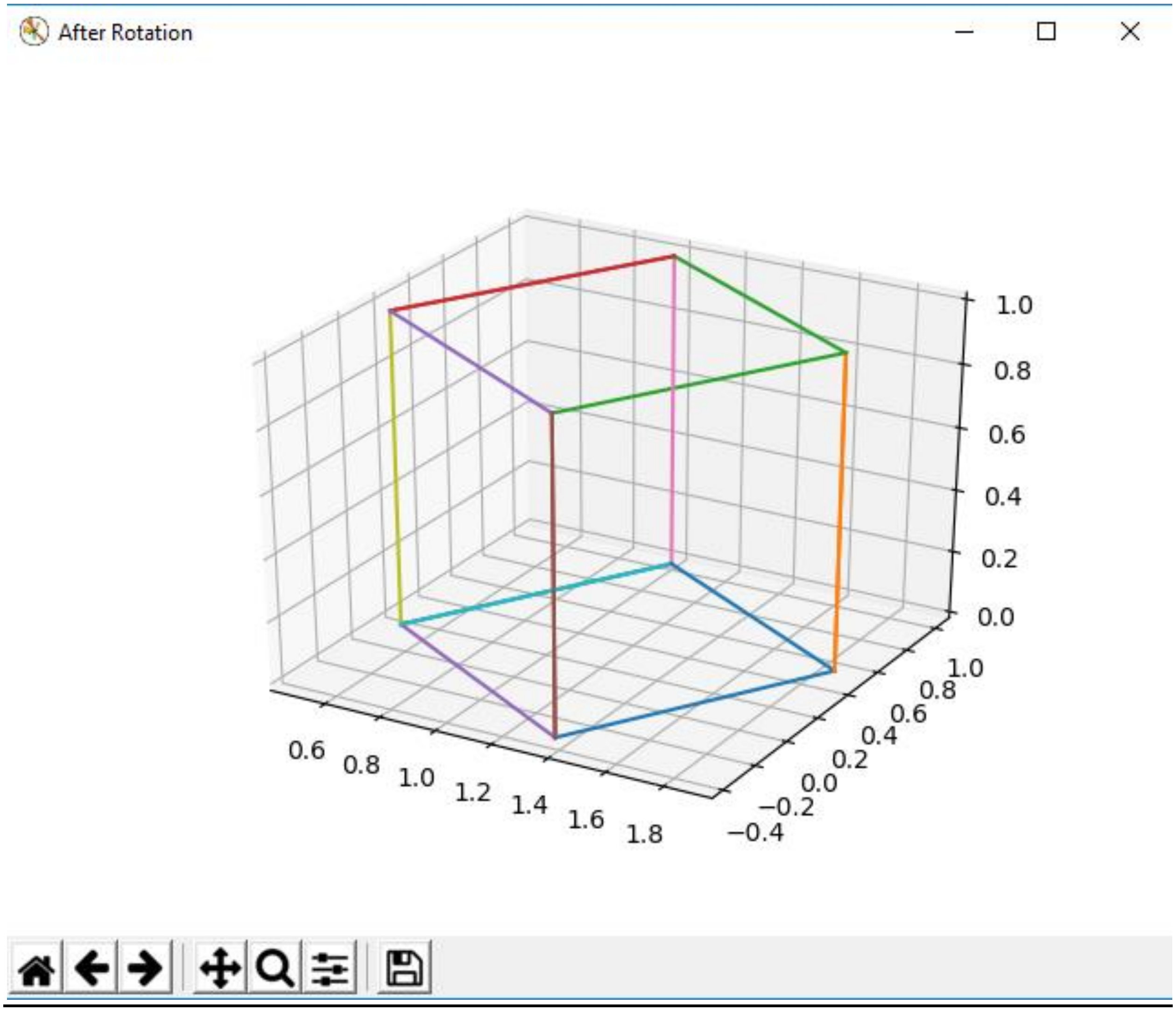


Rotation

Example8:

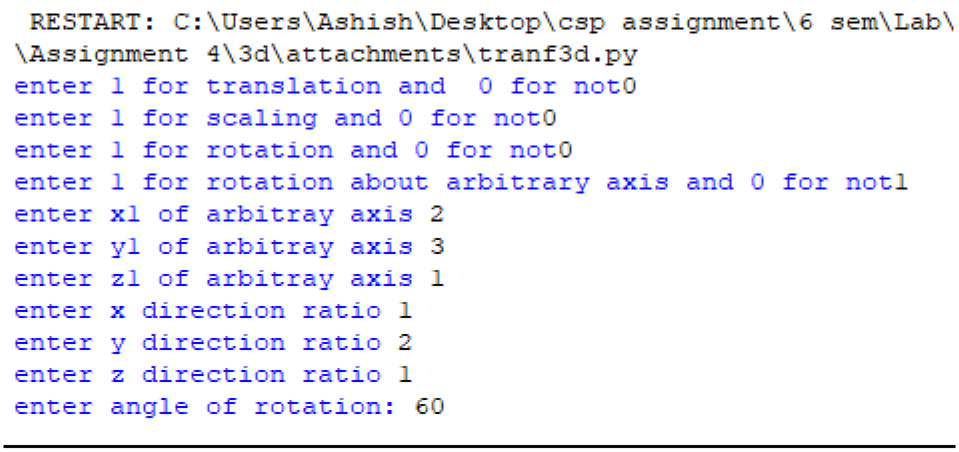


```
 RESTART: C:\Users\Ashish\Desktop\csp assignment\6 
\Assignment 4\3d\attachments\tranf3d.py
enter 1 for translation and  0 for not0
enter 1 for scaling and 0 for not0
enter 1 for rotation and 0 for not1
enter angle of rotation 60
enter 1: x-axis 2: y-axis 3: z-axis 3
```

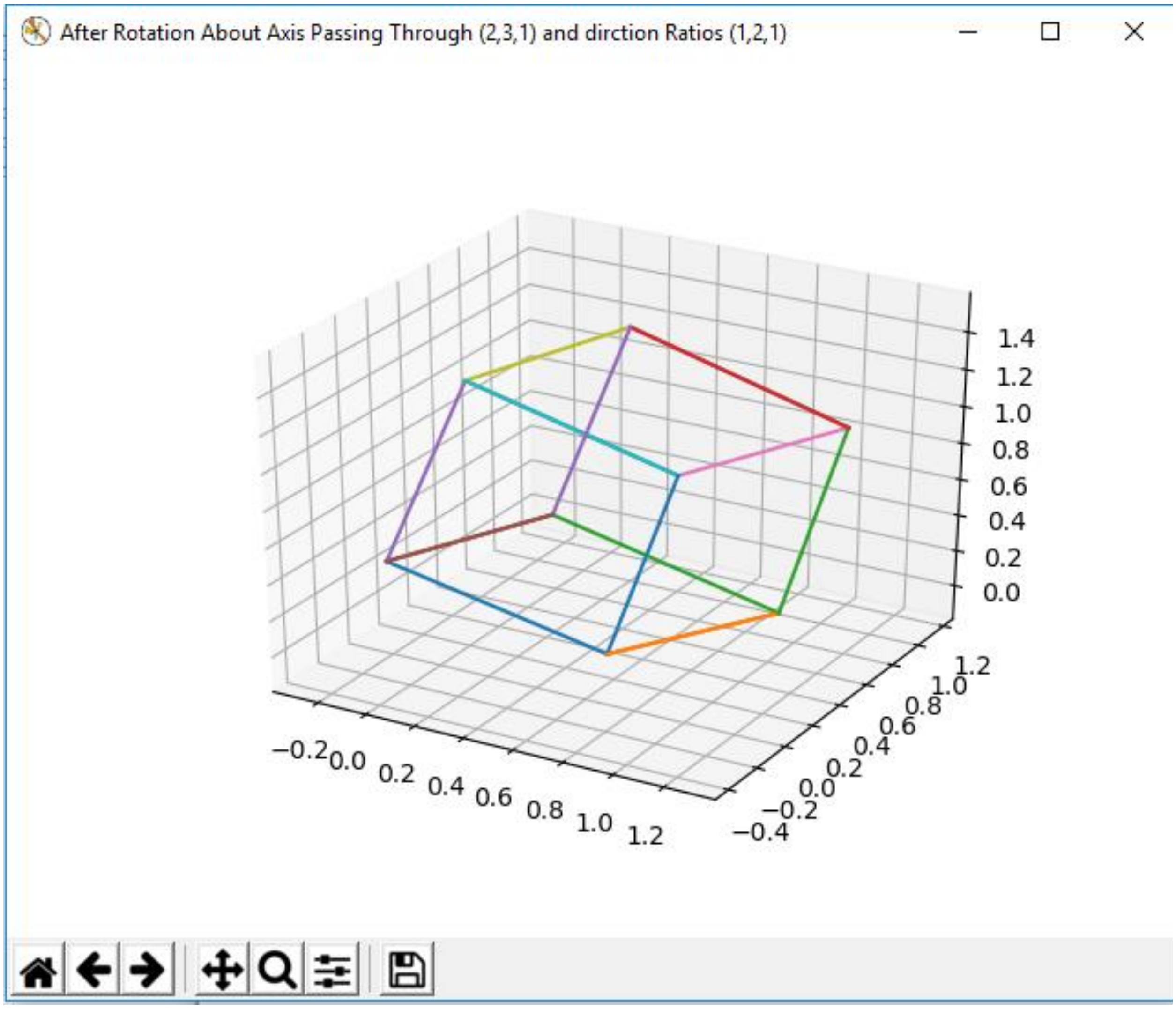


<u>Rotation (About arbitrary line)</u>

Example9:



```
 RESTART: C:\Users\Ashish\Desktop\csp assignment\6 sem\Lab\
\Assignment 4\3d\attachments\tranf3d.py
enter 1 for translation and  0 for not0
enter 1 for scaling and 0 for not0
enter 1 for rotation and 0 for not0
enter 1 for rotation about arbitrary axis and 0 for not1
enter x1 of arbitray axis 2
enter y1 of arbitray axis 3
enter z1 of arbitray axis 1
enter x direction ratio 1
enter y direction ratio 2
enter z direction ratio 1
enter angle of rotation: 60
```



After Rotation About Axis Passing Through (2,3,1) and dirction Ratios (1,2,1)