<u>GROUP4 MEMBERS:</u>

MANZI DAVID	221022262
IRIHOSE ERIC	221001221
MUGISHA REMY	221008577
MASEZERANO ESTHER SAFINA	221008118
UWASE MARIE CLAIRE	221021988
UWIMANA ALINE	221008389
UMURERWA GISELE	221010561
IRUMVA GEDEON	221013355

COMPUTER GRAPHICS ASSIGNIMENT

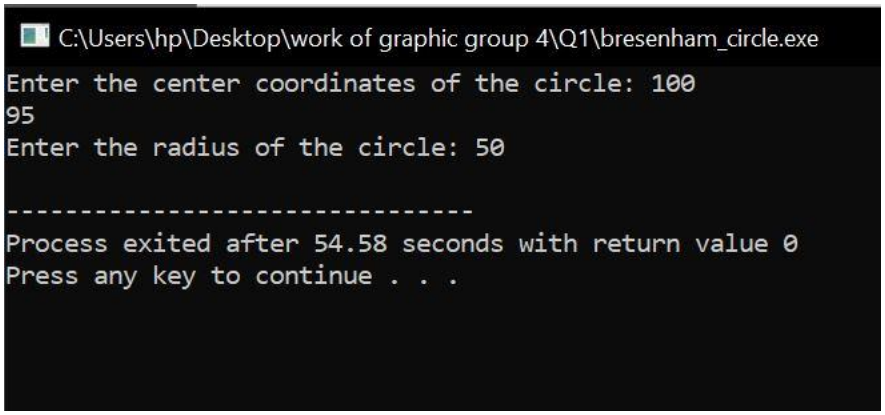
Question 1.

A.

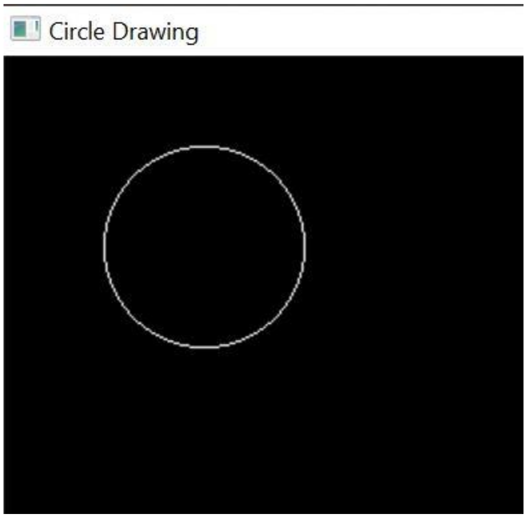
```
#include <iostream>
#include <graphics.h>
using namespace std;

class Circle {
public:
   int centerX, centerY, radius;

   void input() {
      cout << "Enter the center coordinates of the circle: ";
      cin >> centerX >> centerY;
      cout << "Enter the radius of the circle: ";
      cin >> radius;
   }

   void draw() {
        int x = 0, y = radius;
        int d = 3 - 2 * radius;

        initwindow(800, 600, "Circle Drawing");

        while (x <= y) {
            putpixel(centerX + x, centerY + y, WHITE);
            putpixel(centerX + y, centerY + x, WHITE);
            putpixel(centerX - y, centerY + x, WHITE);
            putpixel(centerX - x, centerY + y, WHITE);
            putpixel(centerX - x, centerY - y, WHITE);
            putpixel(centerX - y, centerY - x, WHITE);
            putpixel(centerX + y, centerY - x, WHITE);
            putpixel(centerX + x, centerY - y, WHITE);

            if (d <= 0) {
                d = d + 4 * x + 6;
            } else {
                d = d + 4 * (x - y) + 10;
                y--;
            }
            x++;
        }

        delay(10000);
        closegraph();
    }
};
```

```
int main() {
    Circle circle;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);


    circle.input();
    circle.draw();


    delay(5000);
    return 0;
}
```

Output



```
C:\Users\hp\Desktop\work of graphic group 4\Q1\bresenham_circle.exe
Enter the center coordinates of the circle: 100
95
Enter the radius of the circle: 50

--------------------------------
Process exited after 54.58 seconds with return value 0
Press any key to continue . . .
```

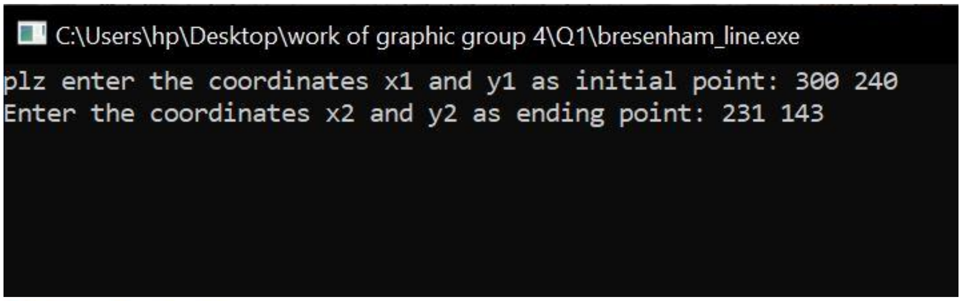


B. Bresenhm line

```
#include <iostream>
#include <graphics.h>
#include <math.h>
using namespace std;

void drawBresenhamLine(int x1, int y1, int x2, int y2, int color) {
    int x, y, dx, dy, dx1, dy1, px, py, xe, ye, i;

    dx = x2 - x1;
    dy = y2 - y1;
    dx1 = fabs(dx);
    dy1 = fabs(dy);
    px = 2 * dy1 - dx1;
    py = 2 * dx1 - dy1;

    initwindow(800, 600, "Bresenham Line Drawing");

    if (dy1 <= dx1) {
        if (dx >= 0) {
            x = x1;
            y = y1;
            xe = x2;
        } else {
            x = x2;
            y = y2;
            xe = x1;
        }
        putpixel(x, y, color);
        for (i = 0; x < xe; i++) {
            x = x + 1;
            if (px < 0) {
                px = px + 2 * dy1;
            } else {
                if ((dx < 0 && dy < 0) || (dx > 0 && dy > 0)) {
                    y = y + 1;
                } else {
                    y = y - 1;
                }
                px = px + 2 * (dy1 - dx1);
            }
            delay(0);
            putpixel(x, y, color);
        }
    } else {
        if (dy >= 0) {
            x = x1;
            y = y1;
            ye = y2;
        } else {
            x = x2;
            y = y2;
            ye = y1;
        }

        putpixel(x, y, color);
```

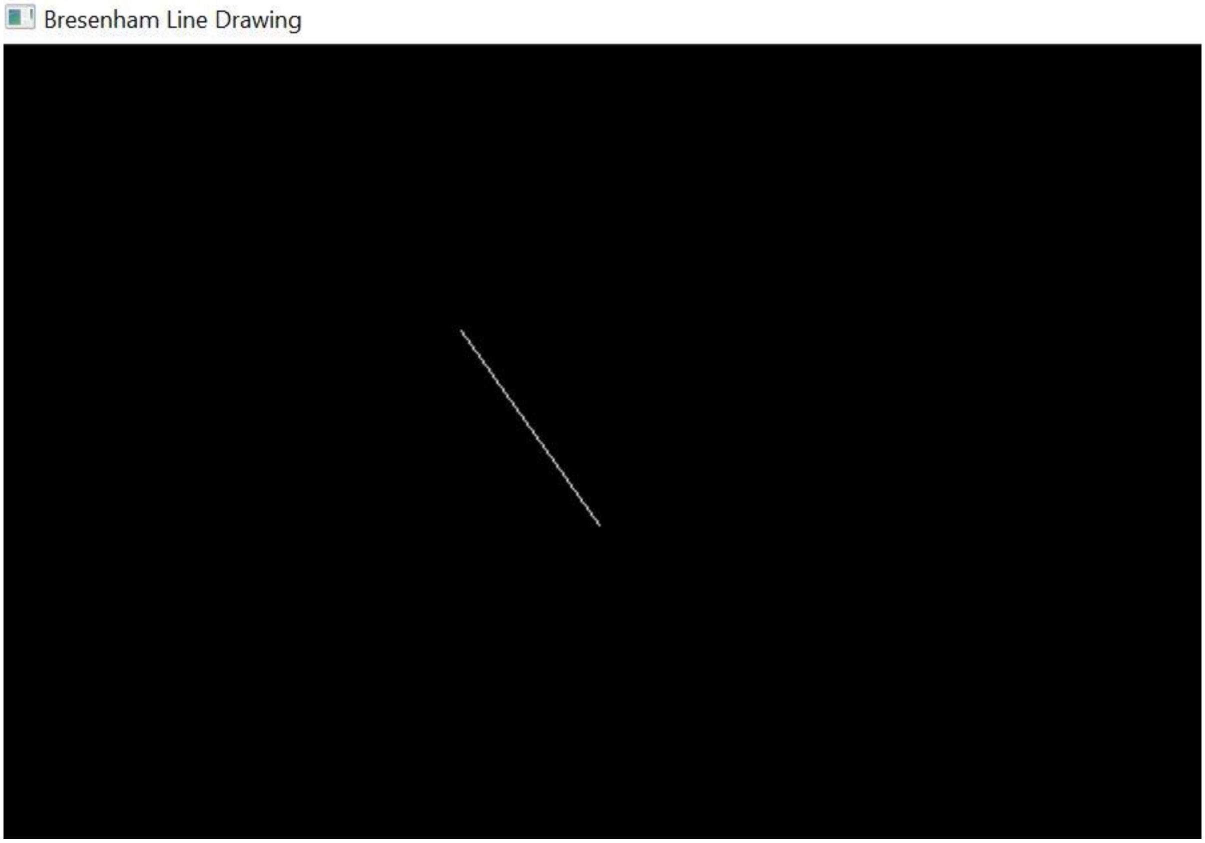
```
        for (i = 0; y < ye; i++) {
            y = y + 1;
            if (py <= 0) {
                py = py + 2 * dx1;
            } else {
                if ((dx < 0 && dy < 0) || (dx > 0 && dy > 0)) {
                    x = x + 1;
                } else {
                    x = x - 1;
                }
                py = py + 2 * (dx1 - dy1);
            }
            delay(0);
            putpixel(x, y, color);
        }
    }

    delay(10000);
    closegraph();
}

int main() {
    int x1, x2, y1, y2;
    initwindow(800, 600, "Bresenham Line Drawing");

    cout << "plz enter the coordinates x1 and y1 as initial point: ";
    cin >> x1 >> y1;
    cout << "Enter the coordinates x2 and y2 as ending point: ";
    cin >> x2 >> y2;
    drawBresenhamLine(x1, y1, x2, y2, WHITE);


    getch();
    closegraph();
    return 0;
}
```

Output of Bresenhma line



C:\Users\hp\Desktop\work of graphic group 4\Q1\bresenham_line.exe

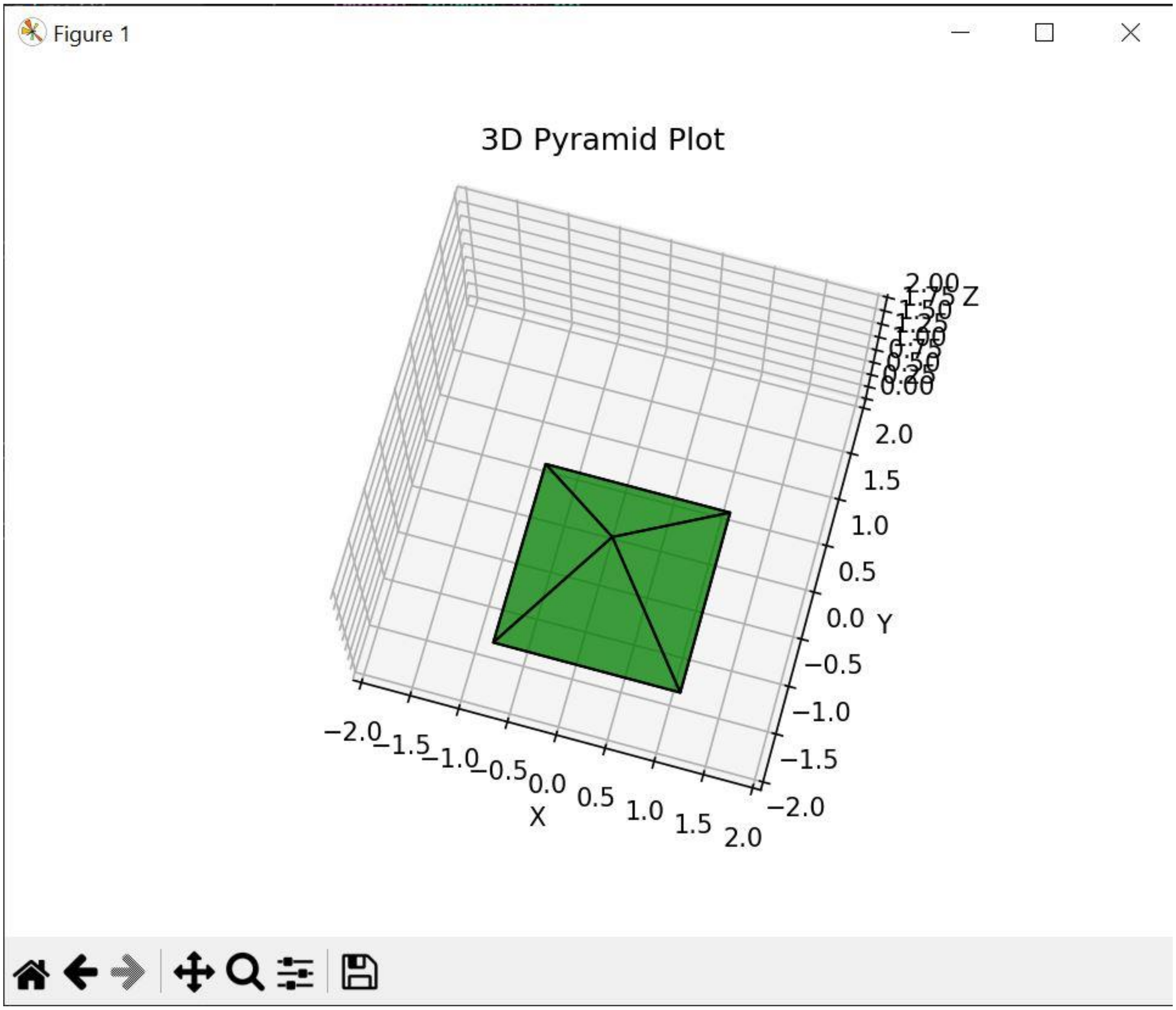
```
plz enter the coordinates x1 and y1 as initial point: 300 240
Enter the coordinates x2 and y2 as ending point: 231 143
```



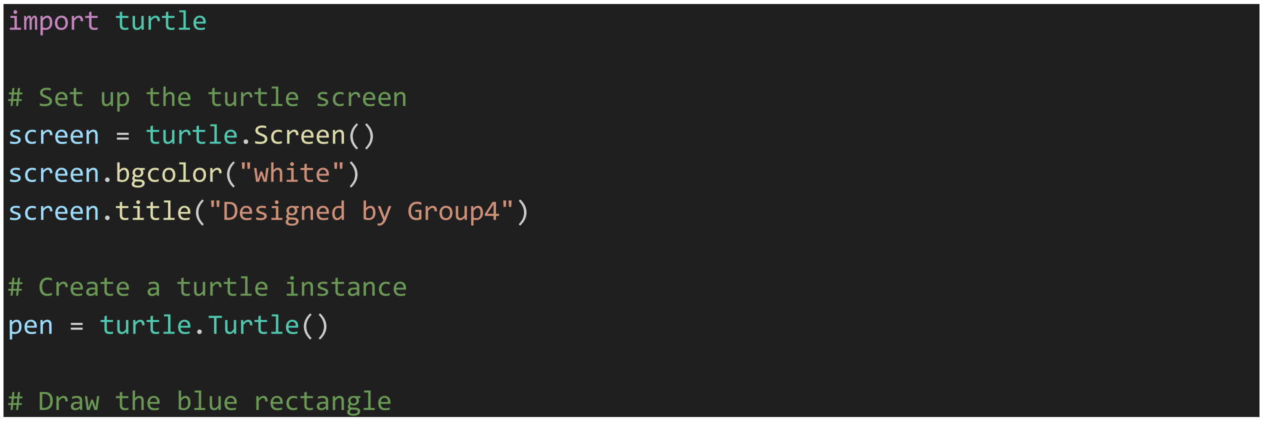
Question 2. modeling

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

# Create a figure and a 3D axis
fig = plt.figure()
axis = fig.add_subplot(111, projection='3d')

# Define pyramid vertices
vertices = np.array([
    [0, 0, 0],
    [1, -1, -1],
    [-1, -1, -1],
    [-1, 1, -1],
    [1, 1, -1],
])

# Define faces using vertex indices
faces = [
    [vertices[0], vertices[1], vertices[2]],
    [vertices[0], vertices[2], vertices[3]],
    [vertices[0], vertices[3], vertices[4]],
    [vertices[0], vertices[4], vertices[1]],
    [vertices[1], vertices[2], vertices[3], vertices[4]]
]

# Create a Poly3DCollection from the faces
pyramid = Poly3DCollection(faces, facecolors='green', edgecolors='black',
alpha=0.5)
axis.add_collection3d(pyramid)

# Set plot limits and labels
axis.set_xlim([-2, 2])
axis.set_ylim([-2, 2])
axis.set_zlim([0, 2])
axis.set_xlabel('X')
axis.set_ylabel('Y')
axis.set_zlabel('Z')
axis.set_title('3D Pyramid Plot')

# Show the plot
plt.show()
```

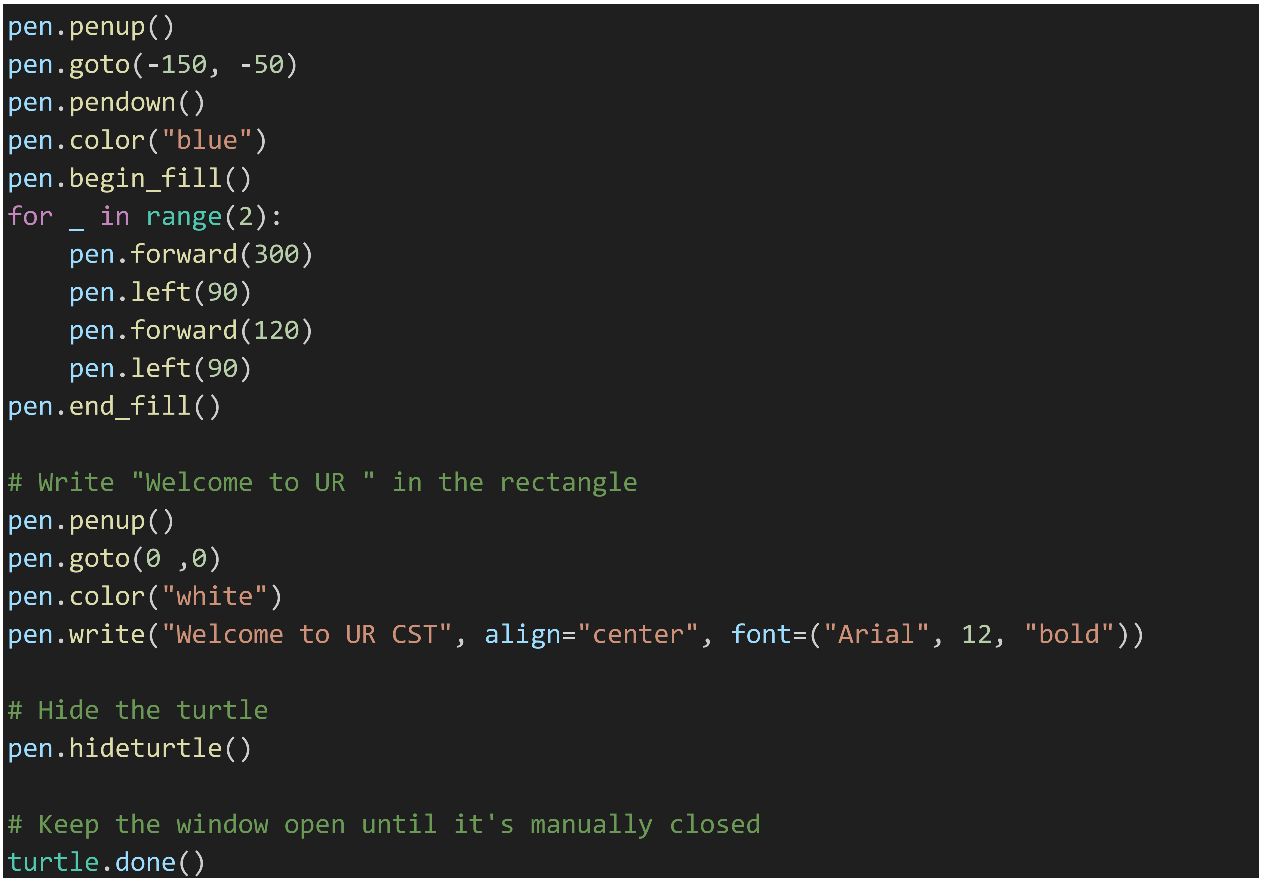
Output



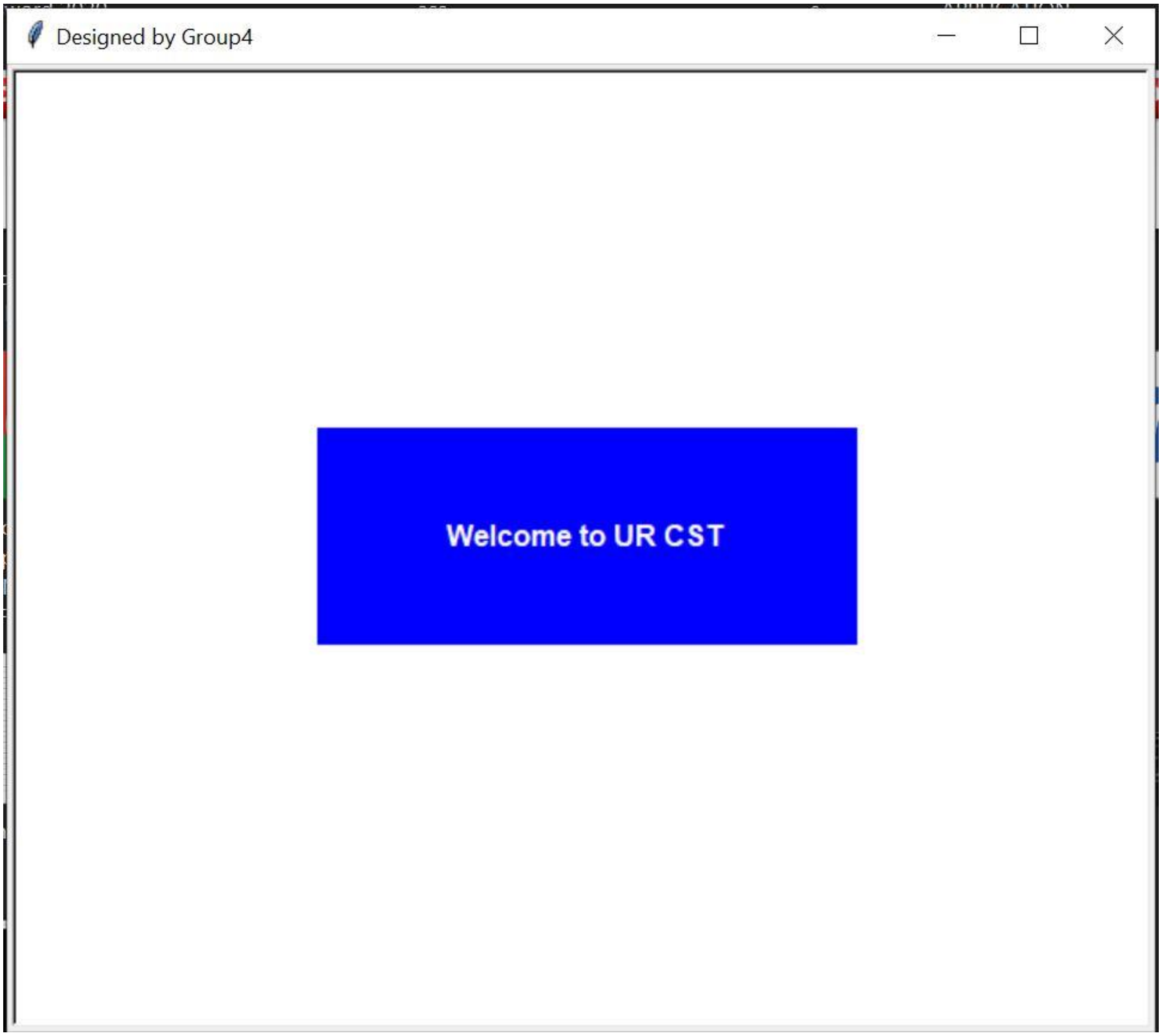
Question 3.a



```
import turtle

# Set up the turtle screen
screen = turtle.Screen()
screen.bgcolor("white")
screen.title("Designed by Group4")

# Create a turtle instance
pen = turtle.Turtle()

# Draw the blue rectangle
```

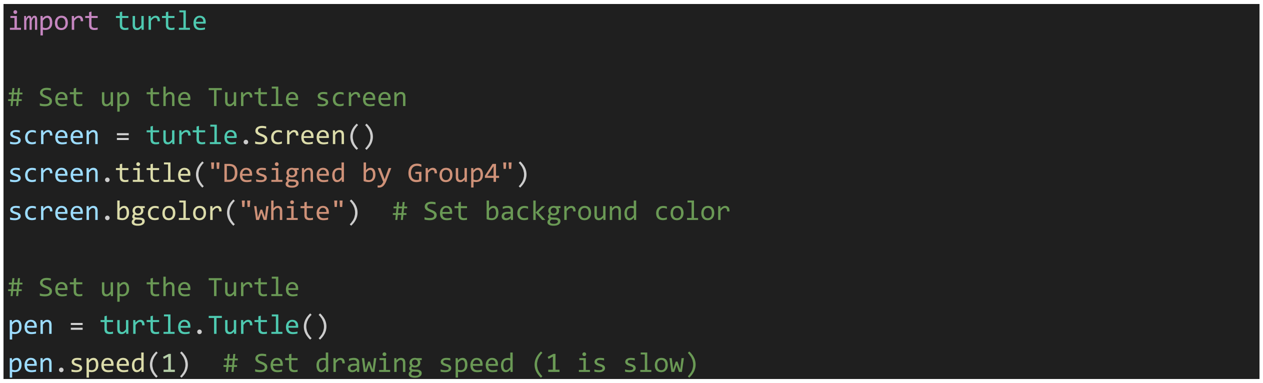


```
pen.penup()
pen.goto(-150, -50)
pen.pendown()
pen.color("blue")
pen.begin_fill()
for _ in range(2):
    pen.forward(300)
    pen.left(90)
    pen.forward(120)
    pen.left(90)
pen.end_fill()

# Write "Welcome to UR " in the rectangle
pen.penup()
pen.goto(0 ,0)
pen.color("white")
pen.write("Welcome to UR CST", align="center", font=("Arial", 12, "bold"))

# Hide the turtle
pen.hideturtle()

# Keep the window open until it's manually closed
turtle.done()
```



Welcome to UR CST

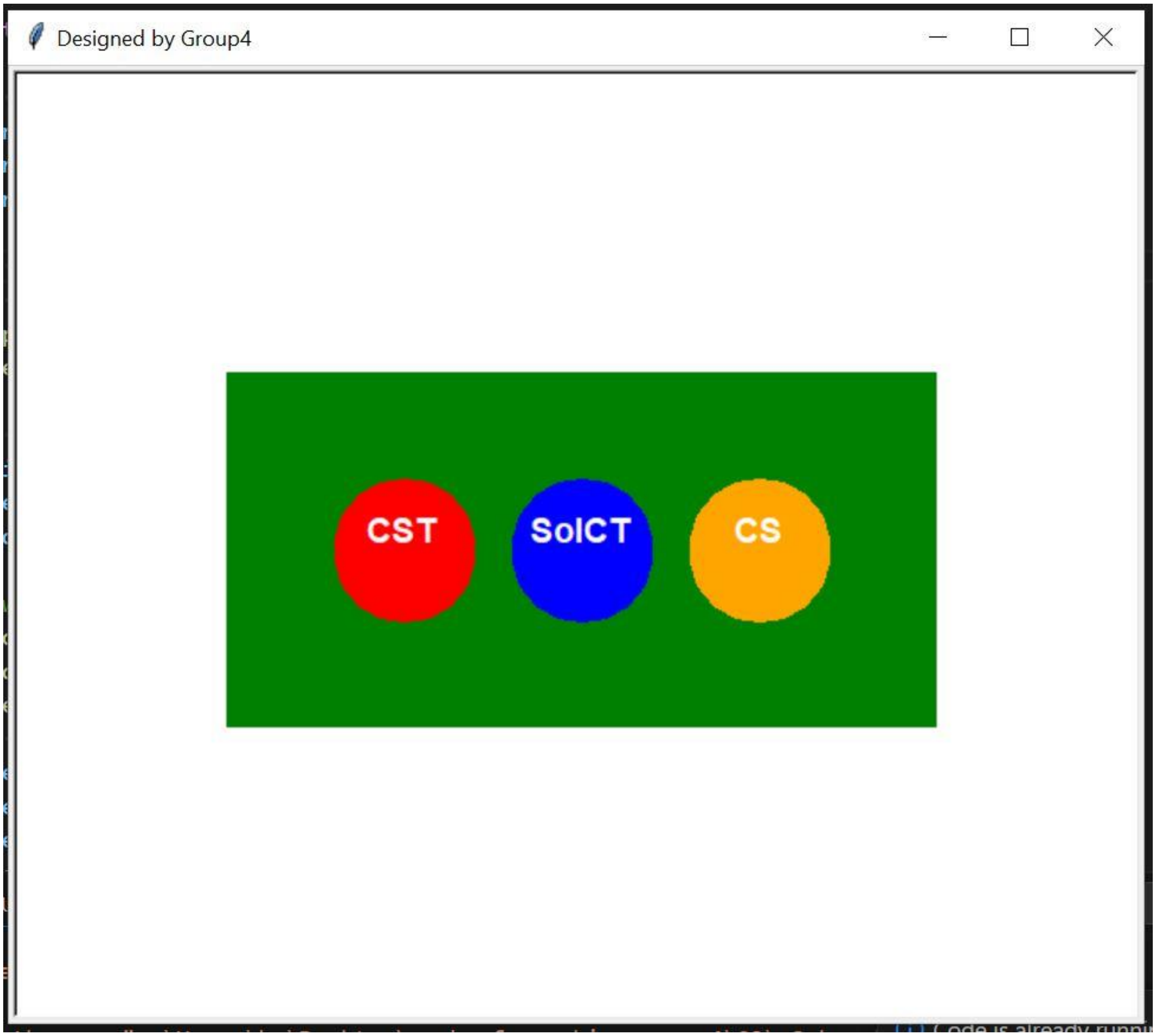
Question 3.b.



```
import turtle

# Set up the Turtle screen
screen = turtle.Screen()
screen.title("Designed by Group4")
screen.bgcolor("white")  # Set background color

# Set up the Turtle
pen = turtle.Turtle()
pen.speed(1)  # Set drawing speed (1 is slow)
pen.penup()

# Set up the rectangular box
box_width = 400
box_height = 200
box_color = "green"

# Draw the rectangular box
pen.goto(-box_width / 2, -box_height / 2)
pen.color(box_color)
pen.begin_fill()
for _ in range(2):
    pen.forward(box_width)
    pen.left(90)
    pen.forward(box_height)
    pen.left(90)
pen.end_fill()

pen.penup()
pen.goto(0, box_height / 2 + 20)  # Position above the box
pen.color("black")  # Set text color
pen.write(" ", align="center", font=("Arial", 14, "bold"))

# List of circle data: (x, y, radius, color, text)
circle_data = [(-100, 0, 40, "red", "CST"),
               (0, 0, 40, "blue", "SoICT"),
               (100, 0, 40, "orange", "CS")]

# Draw the circles using a loop
for x, y, radius, color, text in circle_data:
    pen.penup()
    pen.goto(x, y - radius)
    pen.pendown()
    pen.color(color)
    pen.begin_fill()
    pen.circle(radius)
    pen.end_fill()

    pen.penup()
    pen.goto(x, y)
    pen.color("white")
    pen.write(text, align="center", font=("Arial", 14, "bold"))

# Hide the Turtle
pen.hideturtle()
```

```
# Display the result
screen.mainloop()
```

Output:

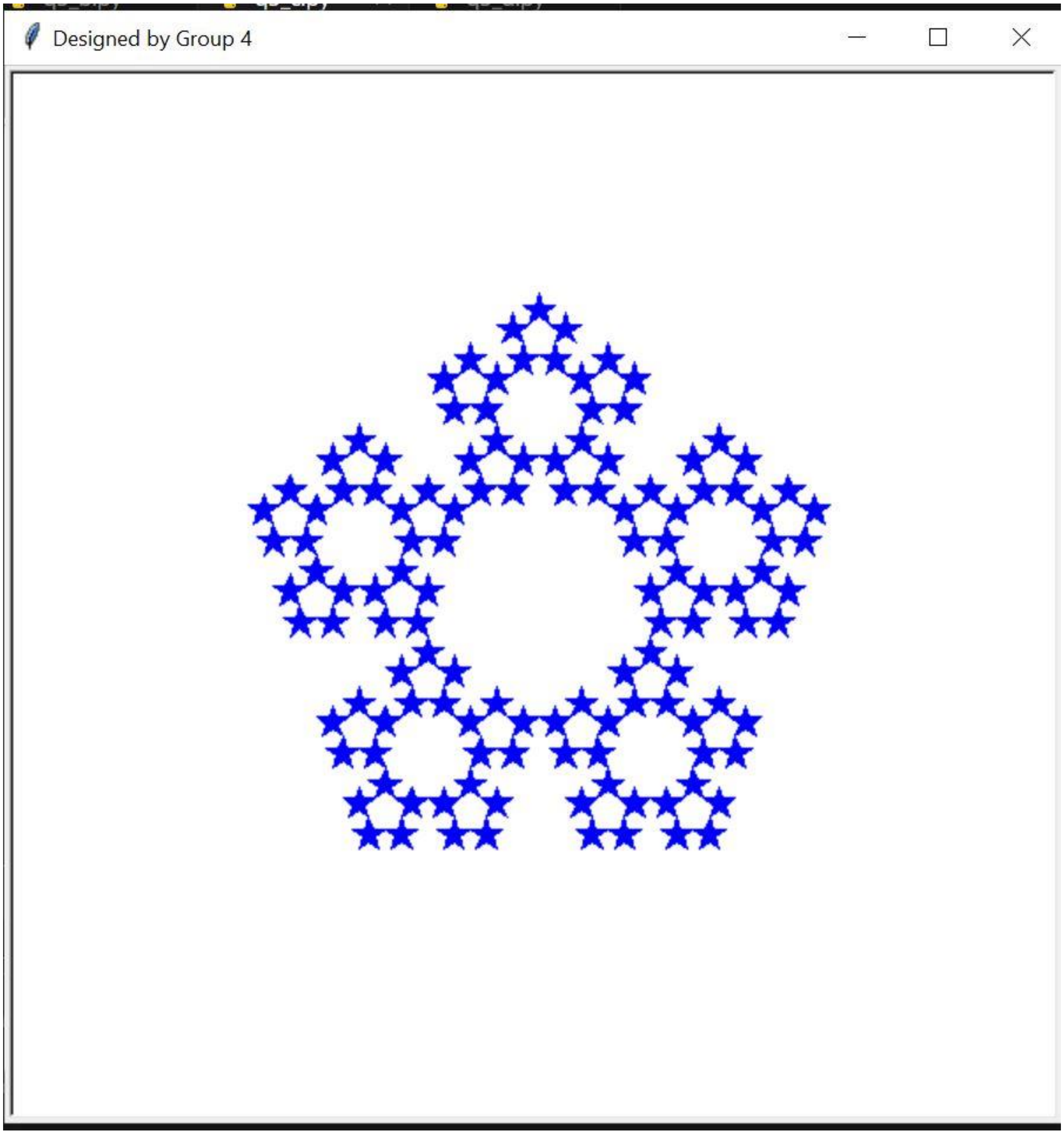


Designed by Group4

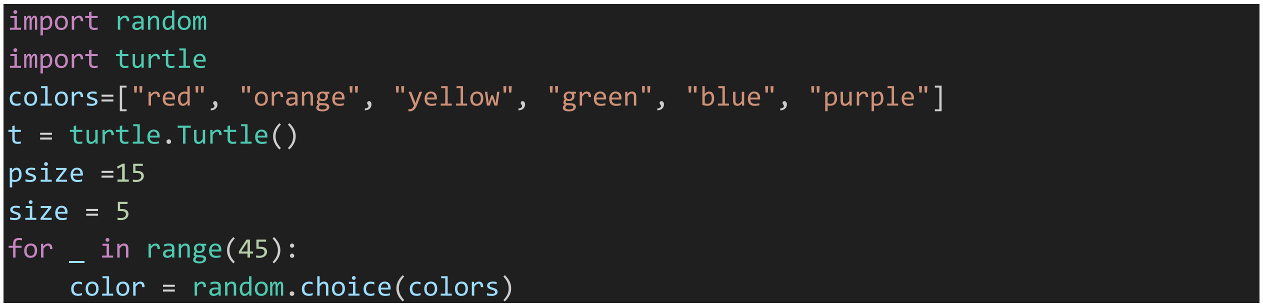
Question 3.c

```
import turtle
import math
# Set up the turtle screen
screen = turtle.Screen()
screen.title("Designed by Group 4")
screen.setup(600, 600)
screen.setworldcoordinates(-1000, -1000, 1000, 1000)
turtle.hideturtle()
turtle.speed(0)
turtle.bgcolor('white')
# Define a function to draw a star
def star(x, y, length, penc, fillc):
    turtle.up()
    turtle.goto(x, y)
    turtle.seth(90)
    turtle.fd(length)
    turtle.seth(180 + 36 / 2)
    L = length * math.sin(36 * math.pi / 180) / math.sin(54 * math.pi / 180)
    turtle.seth(180 + 72)
    turtle.down()
    turtle.fillcolor(fillc)
    turtle.pencolor(penc)
    turtle.begin_fill()
    for _ in range(5):
        turtle.fd(L)
        turtle.right(72)
        turtle.fd(L)
        turtle.left(144)
    turtle.end_fill()
# Define a recursive function to draw a star fractal
def star_fractal(x, y, length, penc, fillc, n):
    if n == 0:
        star(x, y, length, penc, fillc)
        return
    length2 = length / (1 + (math.sin(18 * math.pi / 180) + 1) / math.sin(54 *
math.pi / 180))
    L = length - length2 * math.sin(18 * math.pi / 180) / math.sin(54 * math.pi /
180)
    for i in range(5):
        star_fractal(x + math.cos((90 + i * 72) * math.pi / 180) * (length -
length2),
                     y + math.sin((90 + i * 72) * math.pi / 180) * (length -
length2),
                     length2, penc, fillc, n - 1)
# Draw a star fractal with specified parameters
star_fractal(0, 0, 600, 'blue', 'blue', 3)
screen.update()# Update the screen and finish the drawing
turtle.done()
```

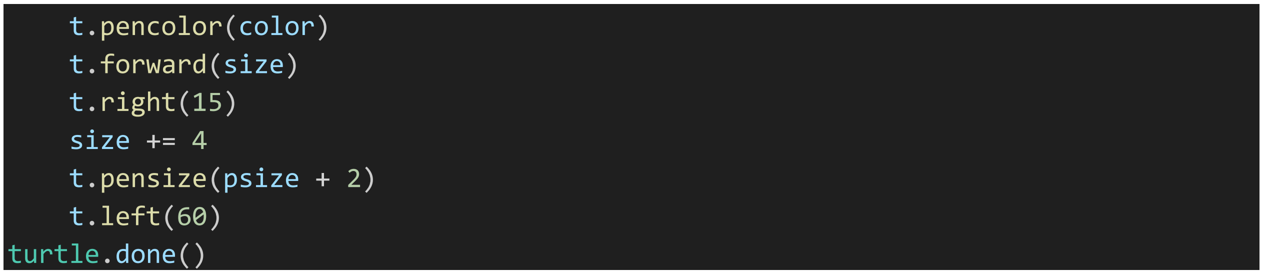
Output:



Question 3.D.

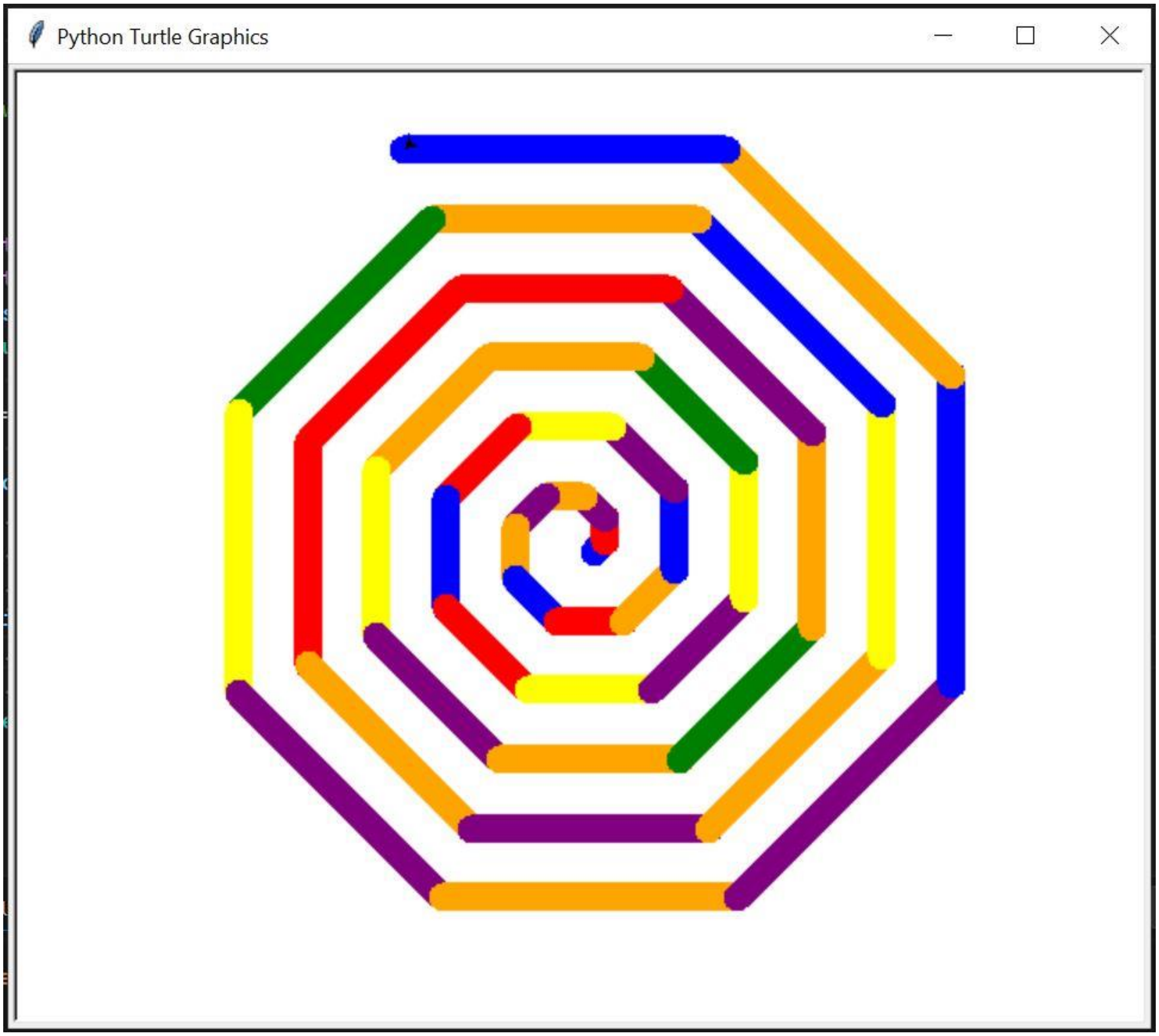


```
import random
import turtle
colors=["red", "orange", "yellow", "green", "blue", "purple"]
t = turtle.Turtle()
psize =15
size = 5
for _ in range(45):
    color = random.choice(colors)
```



```
    t.pencolor(color)
    t.forward(size)
    t.right(15)
    size += 4
    t.pensize(psize + 2)
    t.left(60)
turtle.done()
```

Output:


Python Turtle Graphics