Ex. No.:10	**Graphics Design and Image Processing**	**Register Number:** **URK24CS1189**
03.04.25		

10 A. Create a program to load an image and demonstrate the following operations on it

Aim: -

To perform basic image processing operations (resizing, rotating, mirroring, flipping, cropping, converting to grayscale, thresholding, and blurring) on an input image.

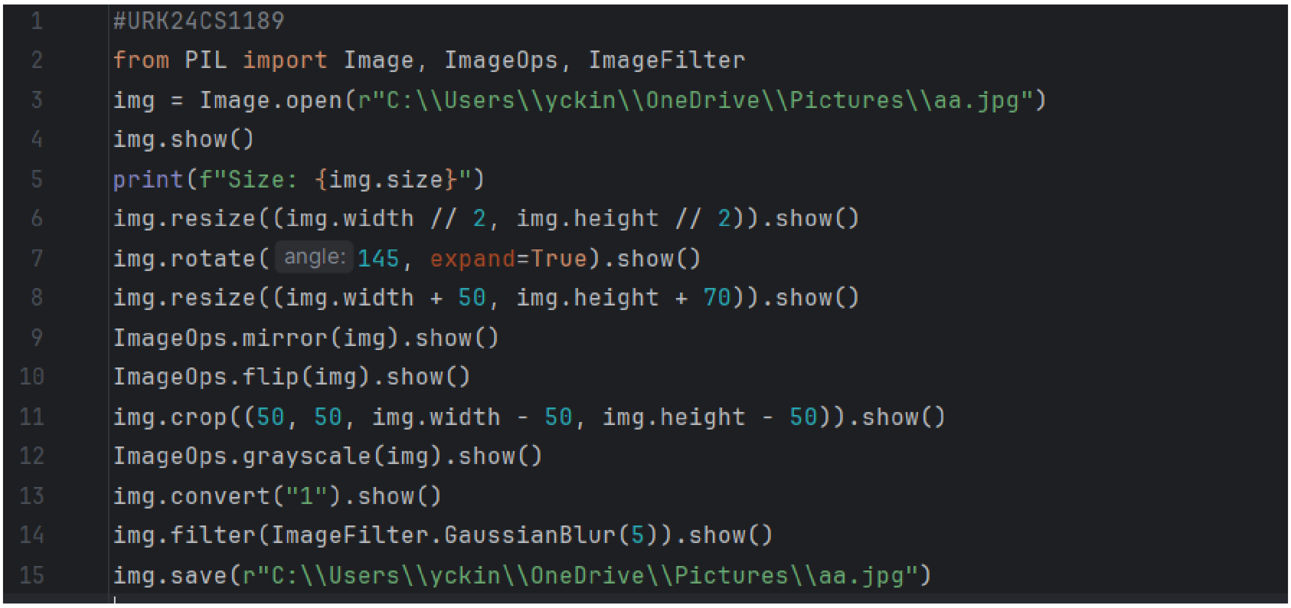
Algorithm: -

Step 1: Open an image file using PIL.
Step 2: Display the original image and print its size.
Step 3: Perform and display the following operations:

- Resize (reduce dimensions by half)
- Rotate by 145 degrees
- Enlarge dimensions
- Mirror (horizontal flip)
- Vertical flip
- Crop edges
- Convert to grayscale
- Apply black-and-white threshold
- Apply Gaussian blur

Step 4: Save the modified image (overwriting the original).

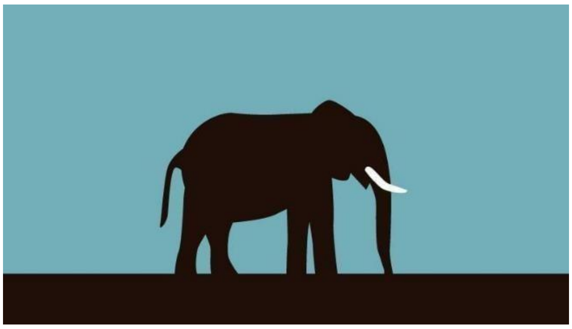
Program: -



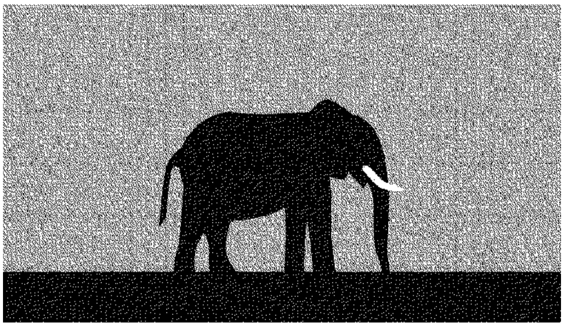
```
#URK24CS1189
from PIL import Image, ImageOps, ImageFilter
img = Image.open(r"C:\\Users\\yckin\\OneDrive\\Pictures\\aa.jpg")
img.show()
print(f"Size: {img.size}")
img.resize((img.width // 2, img.height // 2)).show()
img.rotate( angle: 145, expand=True).show()
img.resize((img.width + 50, img.height + 70)).show()
ImageOps.mirror(img).show()
ImageOps.flip(img).show()
img.crop((50, 50, img.width - 50, img.height - 50)).show()
ImageOps.grayscale(img).show()
img.convert("1").show()
img.filter(ImageFilter.GaussianBlur(5)).show()
img.save(r"C:\\Users\\yckin\\OneDrive\\Pictures\\aa.jpg")
```

Output: -

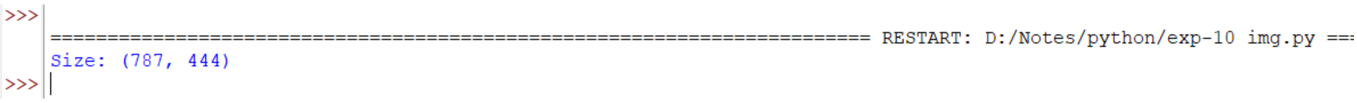
a) Display the image



b) Plot the image in console window

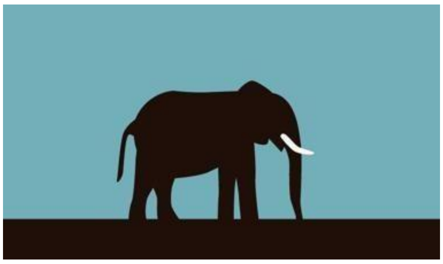


c) Display the image size (width and height)

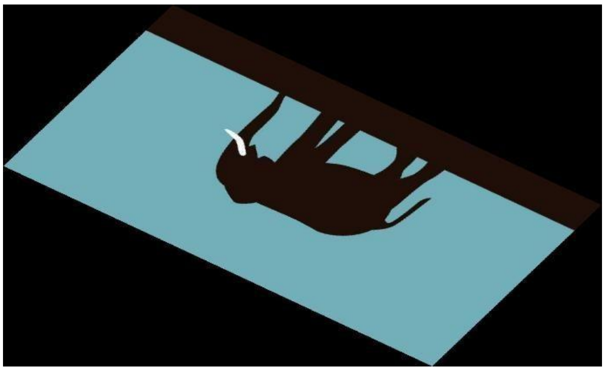


```
>>>
========================================================================= RESTART: D:/Notes/python/exp-10 img.py ==
Size: (787, 444)
>>> |
```

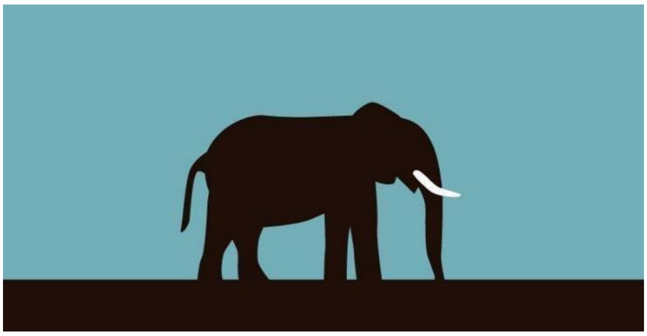
d) Reduce the Image size of its half size



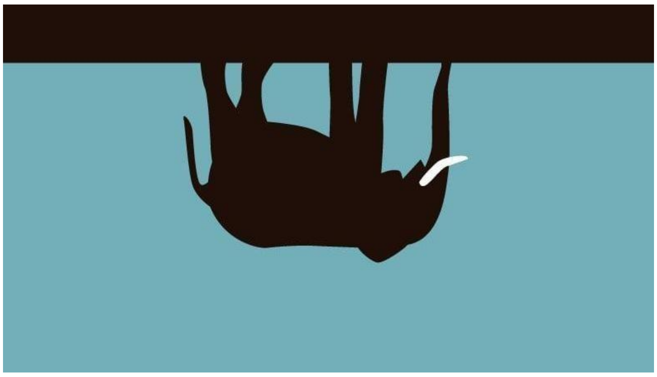
e) Rotate the image 145 degrees



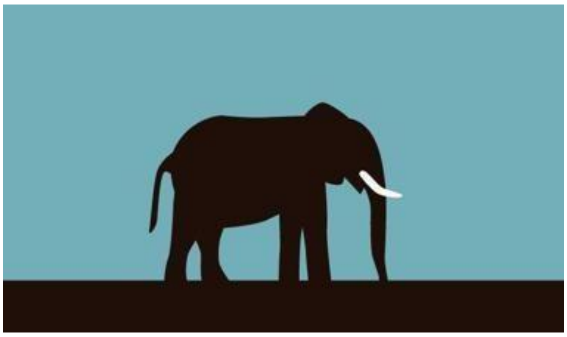
f) Resize the image with 50 units in x direction and 70 units in y direction



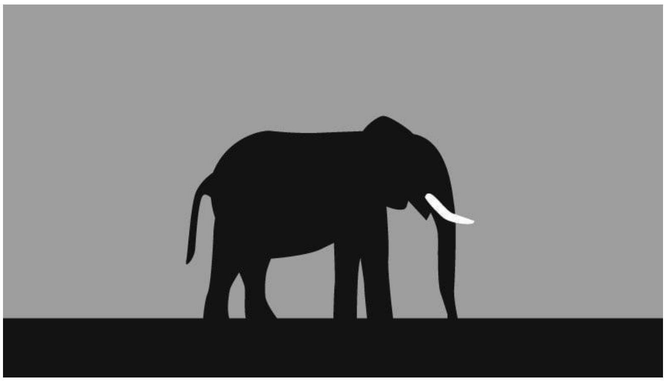
g) Flip the image (Left to Right, Top to Bottom)



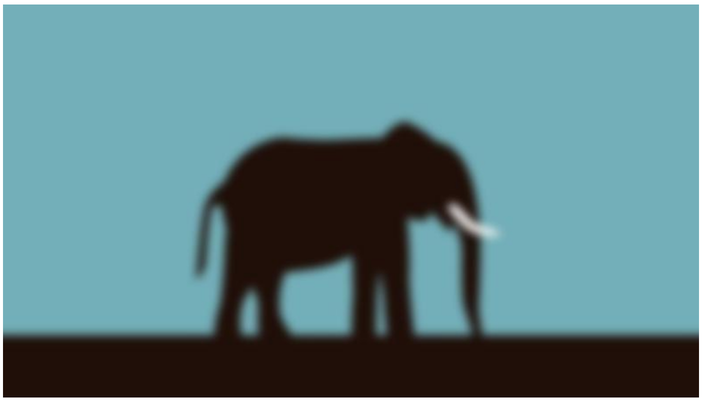
h) Crop the image



i) Change the color image to GrayScale, Black and White



j) Apply blur effect on the image



Result: -

Thus, the output is verified sucessfully.

10 B(i). Create a turtle programs for the following graphic designs (national flag)

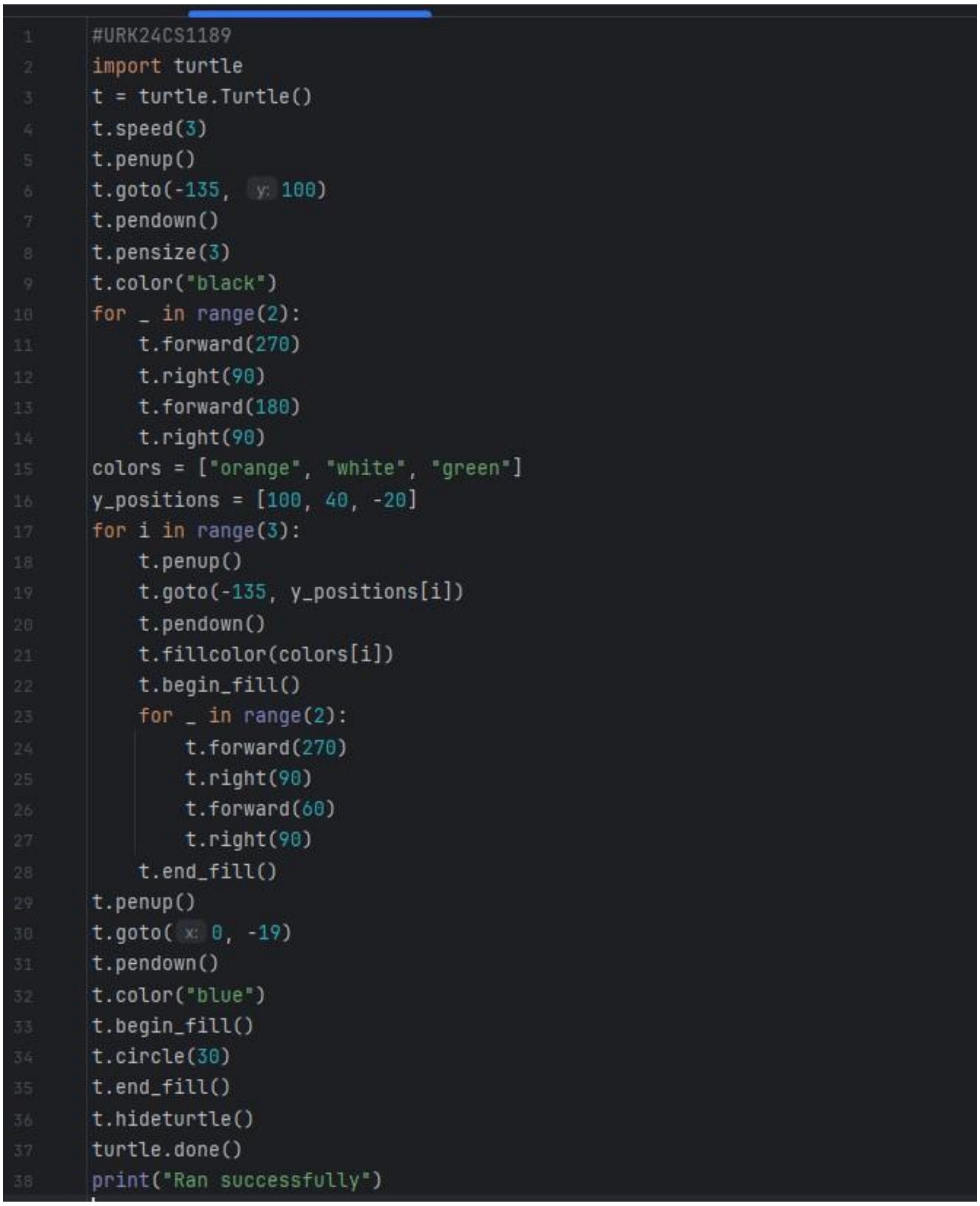
Aim: -

To create a simple traffic light graphic using the Turtle graphics library in Python.

Algorithm: -

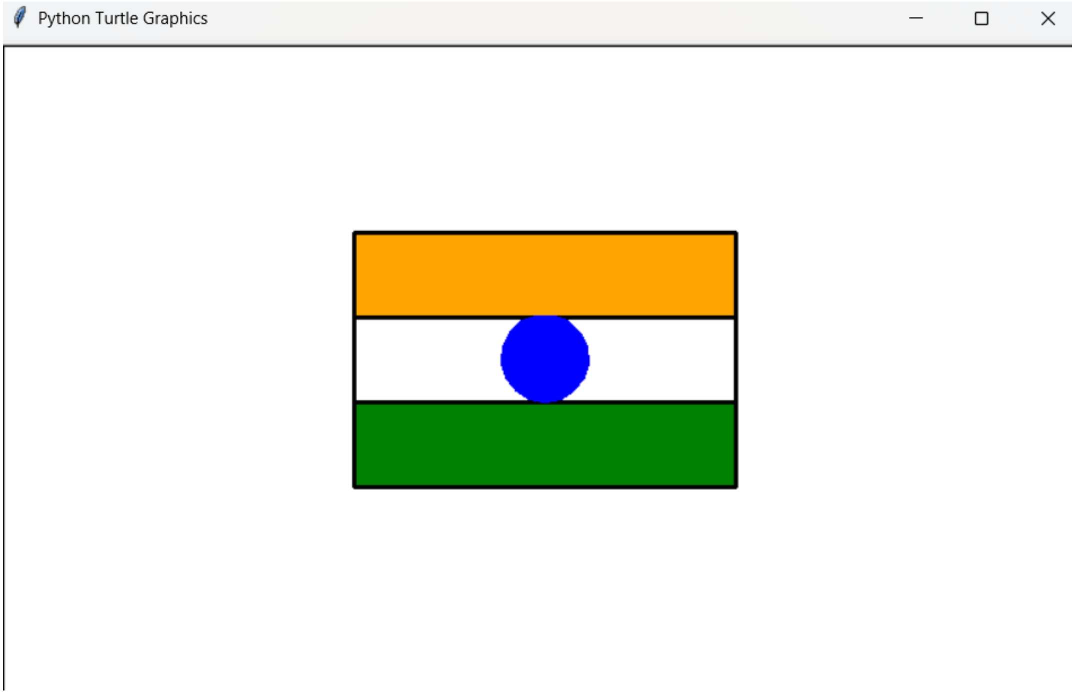
Step 1: Start.
Step 2: Initialize Turtle: Set up the turtle.
Step 3: Draw Traffic Light Body: Draw a rectangle.
Step 4: Draw Lights: Draw three colored rectangles (orange, white, green) for the lights.
Step 5: Draw Bottom Light: Draw a blue circle at the bottom.
Step 6: Finalize: Hide the turtle and complete the drawing.
Step 7: End

Program: -



```python
#URK24CS1189
import turtle
t = turtle.Turtle()
t.speed(3)
t.penup()
t.goto(-135, 100)
t.pendown()
t.pensize(3)
t.color("black")
for _ in range(2):
    t.forward(270)
    t.right(90)
    t.forward(180)
    t.right(90)
colors = ["orange", "white", "green"]
y_positions = [100, 40, -20]
for i in range(3):
    t.penup()
    t.goto(-135, y_positions[i])
    t.pendown()
    t.fillcolor(colors[i])
    t.begin_fill()
    for _ in range(2):
        t.forward(270)
        t.right(90)
        t.forward(60)
        t.right(90)
    t.end_fill()
t.penup()
t.goto(0, -19)
t.pendown()
t.color("blue")
t.begin_fill()
t.circle(30)
t.end_fill()
t.hideturtle()
turtle.done()
print("Ran successfully")
```

Output: -



Result: -

Thus, the programs had been done successfully.

10.B(ii) – A house with light

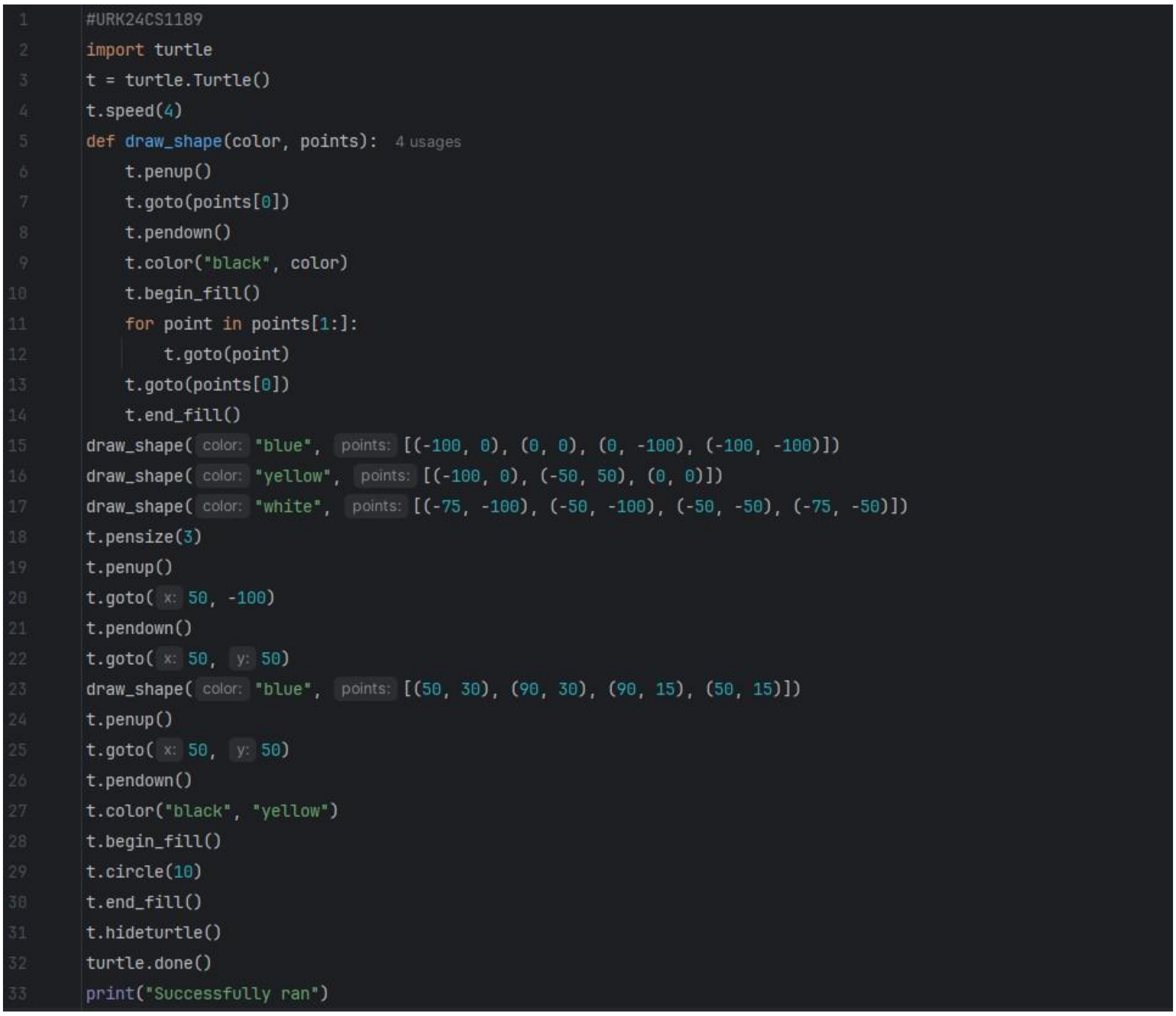
Aim: -

To create a colorful geometric design using the Turtle graphics library in Python.

Algorithm: -

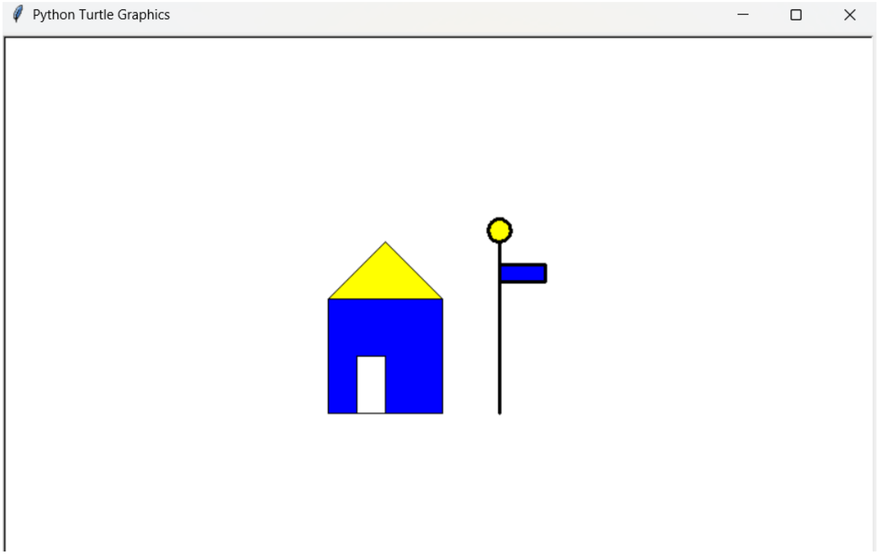
Step 1: Start
Step 2: Initialize Turtle: Set up the turtle.
Step 3: Define draw_shape Function: Create a function to draw filled shapes.
Step 4: Draw Shapes: Use the function to draw a blue rectangle, a yellow triangle, a white rectangle, and a blue rectangle.
Step 5: Draw Vertical Line: Draw a vertical line.
Step 6: Draw Circle: Draw a filled yellow circle.
Step 7: Finalize: Hide the turtle and complete the drawing.
Step 8: End

Program: -



```
#URK24CS1189
import turtle
t = turtle.Turtle()
t.speed(4)
def draw_shape(color, points):  4 usages
    t.penup()
    t.goto(points[0])
    t.pendown()
    t.color("black", color)
    t.begin_fill()
    for point in points[1:]:
        t.goto(point)
    t.goto(points[0])
    t.end_fill()
draw_shape( color: "blue",  points: [(-100, 0), (0, 0), (0, -100), (-100, -100)])
draw_shape( color: "yellow",  points: [(-100, 0), (-50, 50), (0, 0)])
draw_shape( color: "white",  points: [(-75, -100), (-50, -100), (-50, -50), (-75, -50)])
t.pensize(3)
t.penup()
t.goto( x: 50, -100)
t.pendown()
t.goto( x: 50,  y: 50)
draw_shape( color: "blue",  points: [(50, 30), (90, 30), (90, 15), (50, 15)])
t.penup()
t.goto( x: 50,  y: 50)
t.pendown()
t.color("black", "yellow")
t.begin_fill()
t.circle(10)
t.end_fill()
t.hideturtle()
turtle.done()
print("Successfully ran")
```

Output: -



Result: -

Thus, the programs had been done successfully.