EXPERIMENT-

**AIM:** Finding the area of given shape.

**SOFTWARE:** Python(spyder)

**THEORY**:

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection. Contour is a boundary around something that has well defined edges, which means that the machine is able to calculate difference in gradient (significant difference in magnitude of pixel value), try to see if the same difference continues and forms a recognisable shape and draw a boundary around it.

**PROGRAM:**

import cv2

import numpy as np

# Read the image

img = cv2.imread("C:/Users/HP/Desktop/cv img.jpg")

# Convert the image to grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Apply Gaussian blur to reduce noise and improve contour detection

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Use Canny edge detection to find edges in the image

edges = cv2.Canny(blurred, 50, 150)

# Find contours in the edged image

contours, \_ = cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Loop over the contours to find the rectangle

for contour in contours:

perimeter = cv2.arcLength(contour, True)

approx = cv2.approxPolyDP(contour, 0.02 \* perimeter, True)

# If the contour has four vertices, it is likely a rectangle

if len(approx) == 4:

# Draw the contour on the original image

cv2.drawContours(img, [approx], 0, (0, 255, 0), 2)

# Calculate the area of the rectangle using the bounding rectangle

x, y, w, h = cv2.boundingRect(approx)

area = w \* h

print(f"The area of the rectangle is: {area} pixels")

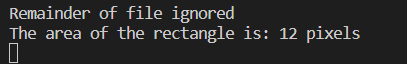
# Display the image with the detected rectangle

cv2.imshow("Detected Rectangle", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

break

**OUTPUT:**

****

**CONCLUSION:** We find the area of the given images by using contours.