

INDIVIDUAL REPORT 1

AI & VISION SYSTEM LABORATORY



January 20, 2024

JAISURYA S

21R218

**Question No. 7**

1. Write a function in Python that accepts a decimal number and returns the equivalent binary number.To make this simple, the decimal number will always be less than 1,024, so the binary number returned will always be less than ten digits long.

**Solution**

**Using built-in function :**

**def decimal\_to\_binary**(decimal\_number):

binary\_representation = bin(decimal\_number)[2:]

return binary\_representation

decimal\_number = int(input("Enter a decimal number: "))

binary\_result = decimal\_to\_binary(decimal\_number)

print(f"The binary representation of {decimal\_number} is: {binary\_result}")

**Output :**

Enter a decimal number: 226

The binary representation of 226 is: 11100010

**Using without built-in function :**

**def decimal\_to\_binary**(decimal\_number):

if decimal\_number == 0:

return '0'

binary\_representation = ''

while decimal\_number > 0:

remainder = decimal\_number % 2

binary\_representation = str(remainder) + binary\_representation

decimal\_number //= 2

return binary\_representation

decimal\_number = int(input("Enter a decimal number: "))

if decimal\_number <= 1024:

binary\_result = decimal\_to\_binary(decimal\_number)

print(f"The binary representation of {decimal\_number} is: {binary\_result}")

else:

print("Binary value exceeds ten digits.")

**Output :**

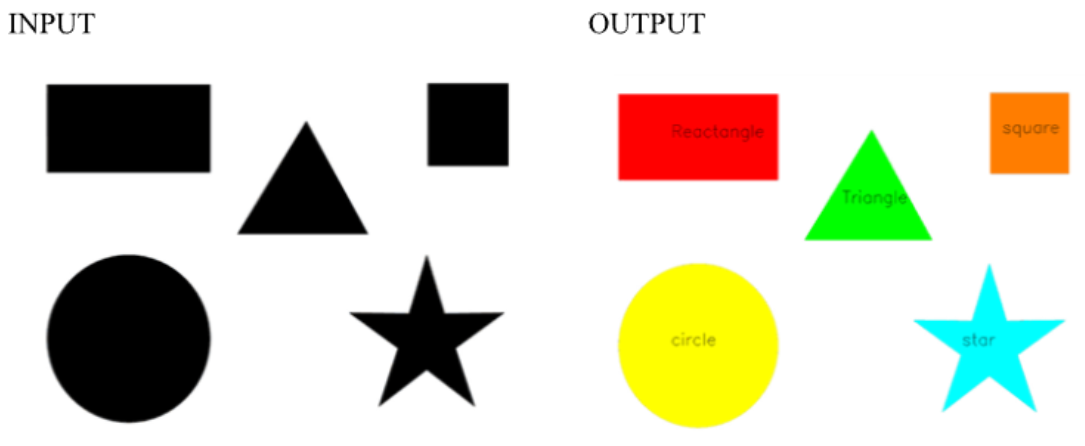
Enter a decimal number: 226

The binary representation of 226 is: 11100010

Enter a decimal number: 2024

Binary value exceeds ten digits.

1. Find the shapes in the given input image and show the output given.



**Software Packages Used**

1. Pycharm IDE

2. Libraries used:

* opencv-python
* numpy

**Shape detection :**

Shape recognition in vision is crucial for automated visual analysis in various applications, including image processing, robotics, and artificial intelligence. It enables machines to identify and understand shapes, facilitating object recognition, quality control in manufacturing, and safer navigation in autonomous vehicles. In medical imaging, shape recognition supports diagnostics and treatment planning. Overall, it enhances the efficiency and accuracy of automated visual processing across diverse domains.

**Program**

import cv2

import numpy as np

**def text**():

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 0.6

font\_thickness = 2

text\_size = cv2.getTextSize(shape\_name, font, font\_scale, font\_thickness)[0]

text\_x = x - text\_size[0] // 2

text\_y = y + text\_size[1] // 2

cv2.putText(img, shape\_name, (text\_x, text\_y),

font, font\_scale, (0, 0, 0), font\_thickness)

**def put\_text\_in\_middle\_rectangle**(img, shape\_name, x, y, w, h):

font = cv2.FONT\_HERSHEY\_SIMPLEX

font\_scale = 0.6

font\_thickness = 2

text\_size = cv2.getTextSize(shape\_name, font, font\_scale, font\_thickness)[0]

text\_x = x + w // 2 - text\_size[0] // 2

text\_y = y + h // 2 + text\_size[1] // 2

cv2.putText(img, shape\_name, (text\_x, text\_y),

font, font\_scale, (0, 0, 0), font\_thickness)

img = cv2.imread('img.png')

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

\_, threshold = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY)

contours, \_ = cv2.findContours(

threshold, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

i = 0

for contour in contours:

# here we are ignoring the first counter because

# findcontour function detects the whole image as a shape

if i == 0:

i = 1

continue

# cv2.approxPloyDP() function to approximate the shape

approx = cv2.approxPolyDP(

contour, 0.01 \* cv2.arcLength(contour, True), True)

color = np.random.randint(0, 255, size=3).tolist()

cv2.drawContours(img, [contour], 0, color, -1) # Fill the shape with color

M = cv2.moments(contour)

if M['m00'] != 0.0:

x = int(M['m10'] / M['m00'])

y = int(M['m01'] / M['m00'])

shape\_name = ""

if len(approx) == 3:

shape\_name = 'Triangle'

text()

elif len(approx) == 4:

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

aspect\_ratio = float(w) / h

if 0.95 <= aspect\_ratio <= 1.05:

shape\_name = 'Square'

put\_text\_in\_middle\_rectangle(img, shape\_name, x, y,w,w)

else:

shape\_name = 'Rectangle'

put\_text\_in\_middle\_rectangle(img, shape\_name, x, y, w, h)

elif len(approx) == 5:

shape\_name = 'Pentagon'

text()

elif len(approx) == 6:

shape\_name = 'Hexagon'

text()

elif len(approx) > 10:

shape\_name = 'Circle'

text()

else:

shape\_name = 'Star'

text()

cv2.imshow('shapes', img)

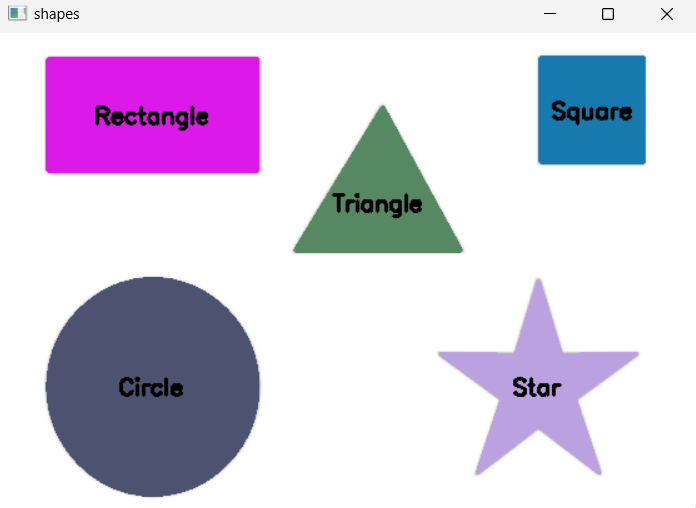
cv2.waitKey(0)

cv2.destroyAllWindows()

**Sample input Image 1**

****

**Output :**

****

**Explanation**

Certainly! Here's an explanation of shape detection in points without code:

**1. Contour Detection:**

- Utilize the `findContours` function to identify contours in a binary image.

**2. Shape Approximation:**

- Use the `approxPolyDP` function to approximate each contour, reducing the number of vertices.

**3. Vertices Count for Shape Identification:**

- Determine the shape based on the number of vertices in the approximated contour.

- Commonly, triangles have 3 vertices, rectangles and squares have 4, pentagons have 5, hexagons have 6, circles have many vertices, and stars often have more than 10.

**4. Drawing Contours:**

- Draw contours with random colors to visualize the identified shapes.

**5. Center Calculation using Moments**

- Calculate the center of each shape using the moments of the contour.

- Moments provide information about the spatial distribution of pixel intensities.

**6. Text Placement:**

- Position text at the center of each shape to label them.

- Utilize the `cv2.putText` function to add text to the image.

**7. Shape Recognition**

- Classify shapes based on their geometrical properties, such as the number of sides and aspect ratio.

**8. Visual Output:**

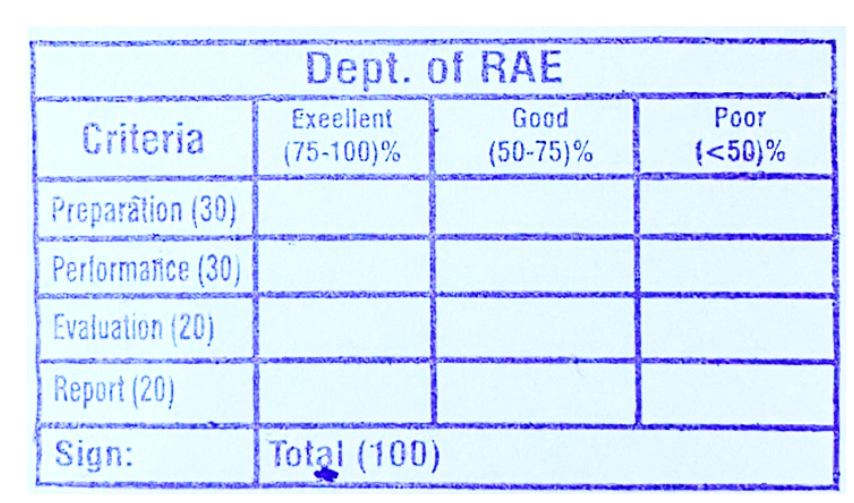
- Display the image with drawn shapes and labeled text for user interpretation.

**9.Random Colors:**

- Apply random colors when drawing contours to distinguish different shapes visually.

**10. Algorithm Flow:**

- The algorithm processes the image by detecting contours, approximating shapes, calculating centers, and labeling shapes with corresponding text.



**RESULT**

Thus the implementation of Multiple shapes recognition was done using OpenCV in python.