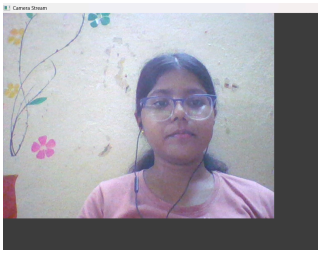
CODE 1-
INPUT- The input to this program is the **live video feed** captured from a **USB webcam** (Camera Index `0`).

Each frame captured from the webcam acts as input in the loop.

OUTPUT- 

LOGIC-The program continuously captures frames from the webcam and displays them in a window, stopping the video stream only when the user presses the **'q'** key.

CODE2-

INPUT- Live video feed captured from the **webcam** (camera index `0`).

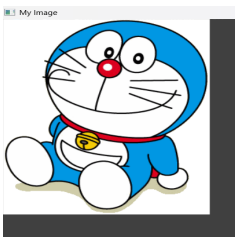
User interaction: pressing **‘q’** key to stop the program.

OUTPUT- Real-time display of webcam video stream.

LOGIC- The program continuously captures frames from the webcam, displays them, and saves each frame as an image inside the **frames** folder until the user presses **‘q’**.

CODE3-

INPUT- 

OUTPUT- 

LOGIC- The program reads an image from the computer and displays it in a window; it waits for a key press and then closes the window.

CODE 4-

INPUT- 

OUTPUT- 

LOGIC-The program loads an image and creates three flipped versions — vertical, horizontal, and both — then displays the original and flipped images until a key is pressed.

CODE 5-

INPUT- 

OUTPUT- 

LOGIC-The program reads an image, resizes it to a fixed dimension (300×300), displays both the original and the resized image, and then saves the resized image.

CODE6- INPUT-

OUTPUT-

LOGIC-The program reads an image, converts it to grayscale, displays both the original and grayscale images, and optionally saves the grayscale image.

CODE 7-

INPUT-

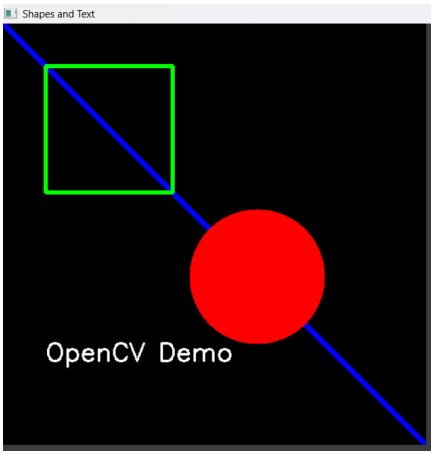
OUTPUT-

LOGIC-The program reads an image, applies a Gaussian blur using a 15×15 kernel, displays both the original and blurred images, and optionally saves the blurred image.

CODE 8-

INPUT- No external image is required; the program creates a **blank 500×500 black image**.

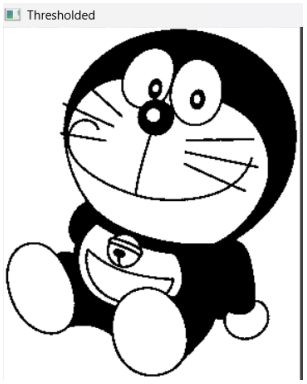
The user presses any key to close the displayed window.

OUTPUT- 

LOGIC- The program creates a blank image, draws geometric shapes (line, rectangle, circle), adds text on it, and displays the result until a key is pressed.

CODE 9-

INPUT - 

OUTPUT- 

LOGIC-The program reads an image in grayscale, applies binary thresholding at a value of 127, and displays both the original and thresholded images.

CODE 10 -

INPUT-

OUTPUT- 

LOGIC- The program reads an image in grayscale, applies the Canny edge detection algorithm with thresholds 100 and 200, and displays the detected edges.

A display window titled **"Edges"** showing the edges of the input image as white lines on a black background

CODE 11-

INPUT-

OUTPUT- 

LOGIC- The program loads a pre-trained Haar cascade classifier, detects faces in the input image, draws rectangles around detected faces, and displays the result.

CODE 12-

OUTPUT- 

LOGIC- A display window titled **contours** showing the original image with **green outlines** highlighting all detected contours.

CODE 13-

INPUT-

OUTPUT- Three display windows:

- **Original** → Original image
- **Mask** → Binary mask showing detected blue regions in white
- **Filtered** → Original image with only the blue regions visible

LOGIC- The program converts the image to HSV color space, creates a mask for the blue color range, and extracts only the blue regions from the image.

CODE 14-

INPUT -

OUTPUT- 

LOGIC- Image showing only the segmented foreground with the background removed

CODE 15 -

INPUT- Live video feed from the **webcam** (camera index 0).

OUTPUT- Three real-time display windows:

- **Frame** → Original webcam feed
- **Mask** → Binary mask highlighting blue regions
- **Tracked** → Webcam feed showing only the blue regions

LOGIC- The program captures frames from the webcam, converts them to HSV color space, creates a mask for a specified blue color range, and displays the original, mask, and color-tracked output in real time.

CODE 16-

OUTPUT-

LOGIC- The program converts the image to binary using thresholding, then applies **erosion** to shrink and **dilation** to expand the foreground regions, displaying all results.

Three display windows:

- **Original** → Binary inverted image
- **Erosion** → Foreground objects shrunk
- **Dilation** → Foreground objects expanded