**Program 2:-**

**Write a python program using opencv to Open an image in grayscale mode along with window name**

**Display image in grey scale mode**

**import** cv2

# Using cv2.imread() method

# Using 0 to read image in grayscale mode

img **=** cv2.imread(img, 0)

# Displaying the image

cv2.imshow('image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

|  |
| --- |
| Display window name using python program |

**import** cv2

path **=** r'C:\Users\Rajnish\Desktop\geeksforgeeks.png'

image **=** cv2.imread(path)

window\_name **=** 'image'

cv2.imshow(window\_name, image)

cv2.waitKey(0)

cv2.destroyAllWindows()

program 3

write a python program to resize an image with the scaling factors height and width.

import cv2

import numpy as np

image = cv2.imread('2.jpeg')

cv2.imshow('Original Image', image)

up\_width = 800

up\_height = 600

up\_points = (up\_width, up\_height)

resized\_up = cv2.resize(image, up\_points, interpolation= cv2.INTER\_LINEAR)

cv2.imshow('Resized Up image by defining height and width', resized\_up)

cv2.waitKey()

import cv2

import numpy as np

image = cv2.imread('2.jpeg')

cv2.imshow('Original Image', image)

down\_width = 300

down\_height = 200

down\_points = (down\_width, down\_height)

resized\_down = cv2.resize(image, down\_points, interpolation= cv2.INTER\_LINEAR)

cv2.imshow('Resized Down by defining height and width', resized\_down)

cv2.waitKey()

Reducing the size of an image will require resampling of the pixels.

Increasing the size of an image requires reconstruction of the image. This means you need to interpolate new pixels.

**Program 4:-**

**Using python open cv program create a border to an image**

**OpenCV-Python** is a library of Python bindings designed to solve computer vision problems. cv2.copyMakeBorder() method is used to create a border around the image like a photo frame.

Python program to frame border in the image

**import** cv2

path **=** r'C:\Users\Rajnish\Desktop\geeksforgeeks\geeks.png'

image **=** cv2.imread(path)

  window\_name **=** 'Image'

 # Using cv2.copyMakeBorder() method

image **=** cv2.copyMakeBorder(image, 10, 10, 10, 10, cv2.BORDER\_CONSTANT, None, value **=** 0)

 # Displaying the image

cv2.imshow(window\_name, image)

# program 5:-

# Python | Image blurring using OpenCV

# Write a python program to smoothing and sharpening the image using linear filters

**import cv2**

**img = cv2.imread('2.jpeg')**

**avging = cv2.blur(img,(10,10))**

**cv2.imshow('Averaging',avging)**

**#cv2.waitKey(0)**

**#Gaussian Blurring**

**# Again, you can change the kernel size**

**gausBlur = cv2.GaussianBlur(img, (5,5),0)**

**cv2.imshow('Gaussian Blurring', gausBlur)**

**#cv2.waitKey(0)**

**# Median blurring**

**medBlur = cv2.medianBlur(img,5)**

**cv2.imshow('Media Blurring', medBlur)**

**cv2.waitKey(0)**

import numpy as np

import cv2

img = cv2.imread('2.jpeg')

#check is input image exists

#define sharpening kernel

sharpeningKernel = np.array(([0, -1, 0],[-1, 5, -1],[0, -1, 0]), dtype="int")

#filter2D is used to perform the convolution.

# The third parameter (depth) is set to -1 which means the bit-depth of the output image is the

# same as the input image. So if the input image is of type CV\_8UC3, the output image will also be of the same type

output = cv2.filter2D(img, -1, sharpeningKernel)

#create windows to display images

cv2.namedWindow("image", cv2.WINDOW\_AUTOSIZE)

cv2.namedWindow("output", cv2.WINDOW\_AUTOSIZE)

#display images

cv2.imshow("image", img)

cv2.imshow("output", output)

#press esc to exit the program

cv2.waitKey(0)

#close all the opened windows

cv2.destroyAllWindows()

# note: develop the program to smoothen the background image using opencv library

# program 6:-

# Filter Color with OpenCV

# import cv2 as cv

# import numpy as np

# #read the image

# img = cv.imread("4.jpeg")

# #convert the BGR image to HSV colour space

# hsv = cv.cvtColor(img, cv.COLOR\_BGR2HSV)

# #set the lower and upper bounds for the green hue

# lower\_green = np.array([50,100,50])

# upper\_green = np.array([70,255,255])

# #create a mask for green colour using inRange function

# mask = cv.inRange(hsv, lower\_green, upper\_green)

# #perform bitwise and on the original image arrays using the mask

# res = cv.bitwise\_and(img, img, mask=mask)

# #create resizable windows for displaying the images

# cv.namedWindow("res", cv.WINDOW\_NORMAL)

# cv.namedWindow("hsv", cv.WINDOW\_NORMAL)

# cv.namedWindow("mask", cv.WINDOW\_NORMAL)

# #display the images

# cv.imshow("mask", mask)

# cv.imshow("hsv", hsv)

# cv.imshow("res", res)

# if cv.waitKey(0):

# cv.destroyAllWindows()