**1.Develop a program to display grayscale image using read and write operation.**

**------------------------------------------------------------------------------------**

import cv2

# function cv2.imread() to read an image

img = cv2.imread("./image/t1.jpg",0)

cv2.imshow('cat',img)

#function cv2.imshow() to display an image in a window.

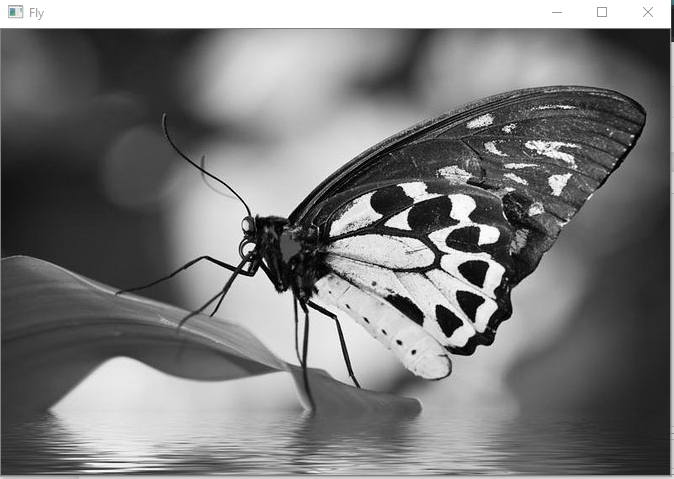
cv2.imwrite("cat\_write.jpg",img)

k = cv2.waitKey()

if k== 27:

cv2.destroyAllWindows()

**Output:**



**2. Develop a program to perform linear transformation on image.**

**------------------------------------------------------------------------------------------------**

import cv2

import numpy as np

FILE\_NAME = './image/fly2.jpg'

try:

img = cv2.imread(FILE\_NAME)

(height, width) = img.shape[:2]

#resized and interpolation

res = cv2.resize(img, (int(width / 2), int(height / 2)), interpolation = cv2.INTER\_CUBIC)

cv2.imwrite('result.jpg', res)

cv2.imshow('image',img)

cv2.imshow('result',res)

k = cv2.waitKey()

if k== 27:

cv2.destroyAllWindows()

except IOError:

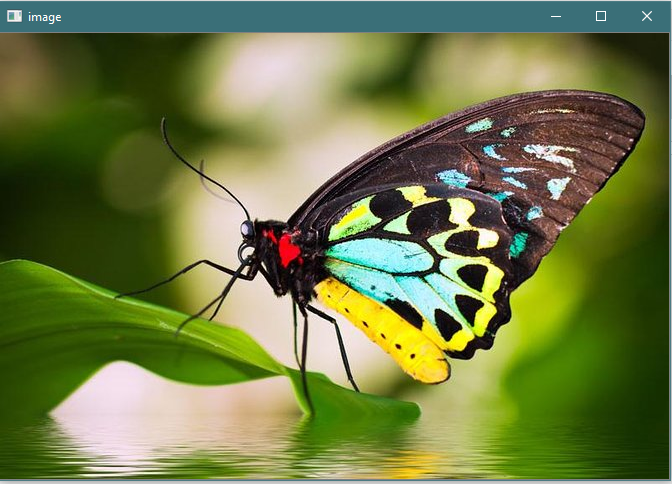
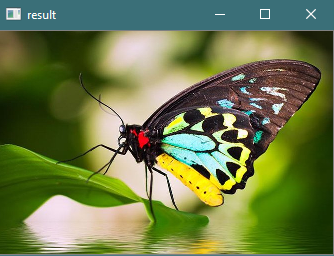
print ('Error Readinf file')

k = cv2.waitKey()

if k== 27:

cv2.destroyAllWindows()

**Output:**

import cv2

import numpy as np

FILE\_NAME = './image/fly2.jpg'

img = cv2.imread(FILE\_NAME)

(rows, cols) = img.shape[:2]

#M = cv2.getRotationMatrix2D(center, theta, 1)

M = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 1)

#rotated = cv2.warpAffine(image, M, (int(w), int(h)))

res = cv2.warpAffine(img, M, (cols, rows))

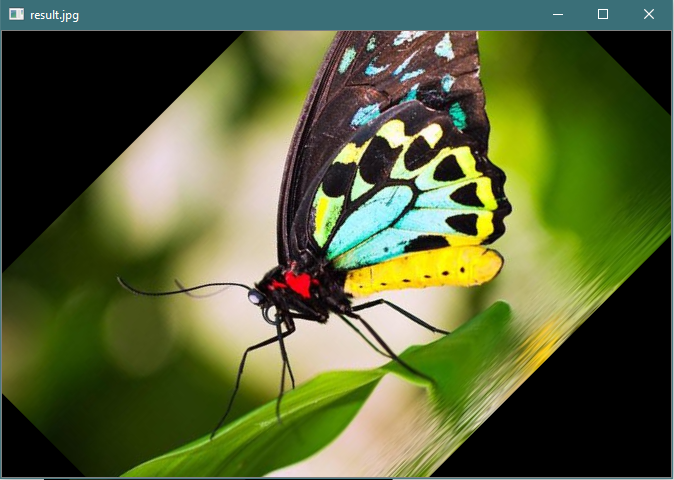
cv2.imshow('result.jpg', res)

k = cv2.waitKey()

if k== 27:

cv2.destroyAllWindows()

**Output:**



**3. Create n number of images and read the directory and perform operation. Sum and Mean.**

**------------------------------------------------------------------------------------------------**

import cv2

import os

path = "./image/"

img\_lst = []

files = os.listdir(path)

#print(files)

for file in files:

filepath=path+"\\"+file

img\_lst.append(cv2.imread(filepath))

i=0

im = []

for im in img\_lst:

#cv2.imshow(files[i],imgs[i])

im= im +img\_lst[i]

i=i+1

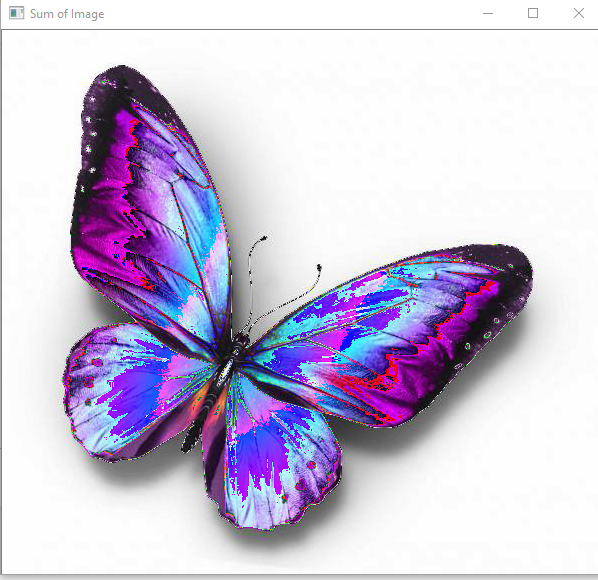
cv2.imshow("Sum of Image",im)

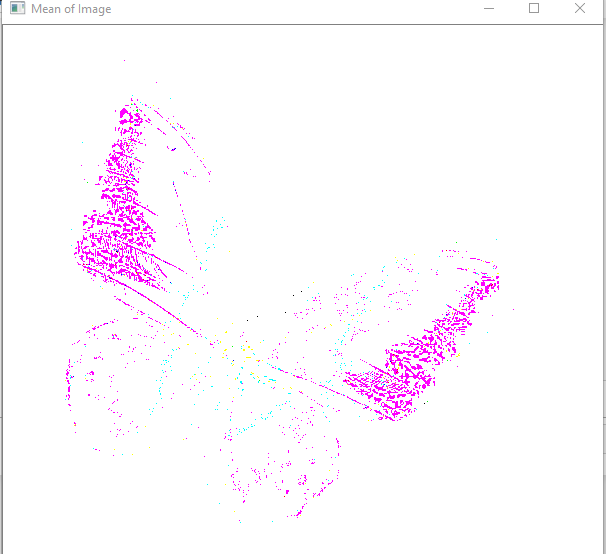
meanImg = im/len(files)

cv2.imshow("Mean of Image",meanImg)

cv2.waitKey(0)

**Output:**

****

****

**4.Write a program to convert color image into gray scale and binary image.**

**------------------------------------------------------------------------------------------------**

import cv2

#reading image

img = cv2.imread("./image/fly2.jpg")

cv2.imshow("Org",img)

#Converting to Gray

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

cv2.imshow("Gray",gray)

cv2.waitKey(0)

cv2.destroyAllWindows()

#Converting to binary

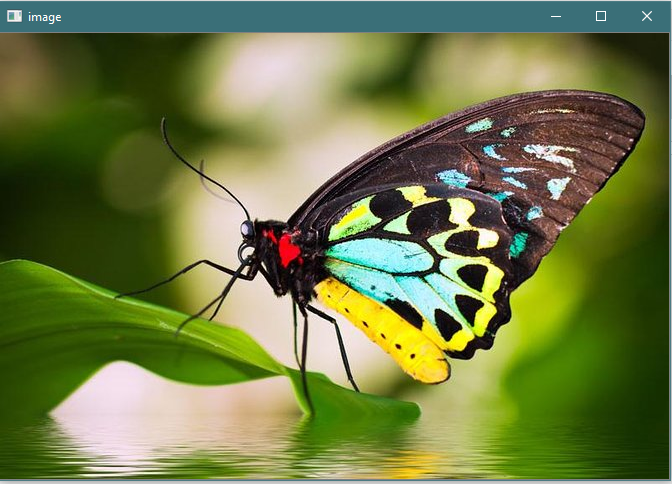
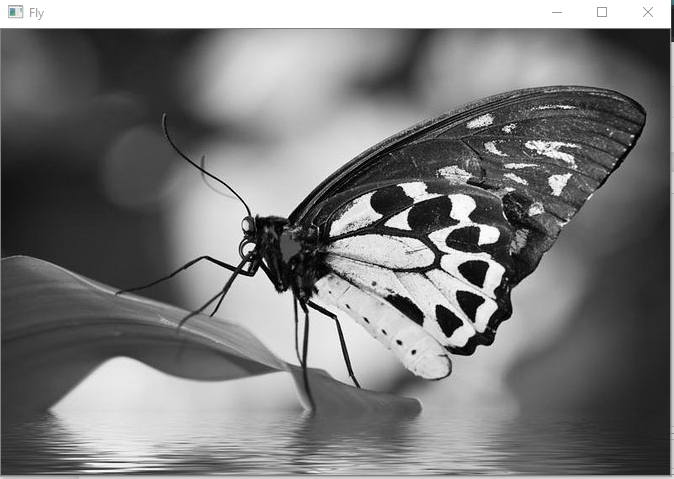
ret, bw\_img = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

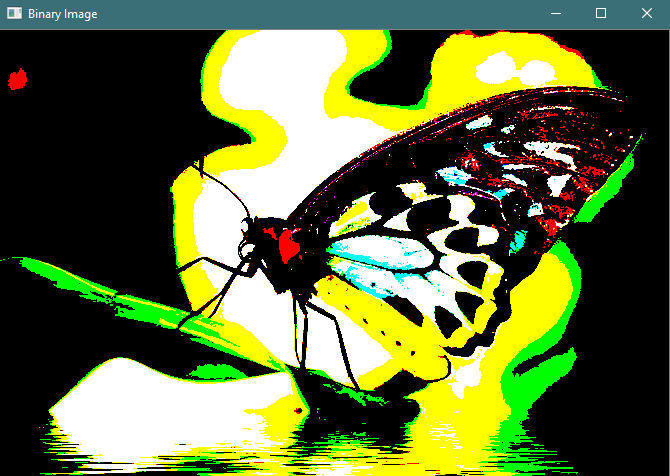
cv2.imshow("Binary Image",bw\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output:**





**5.Write a program to convert color image into different color space.**

**------------------------------------------------------------------------------------------------**

import cv2

img = cv2.imread("./image/fly1.jpg")

#Gray

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

cv2.imshow("GRAY image",gray)

#HSV

hsv=cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

cv2.imshow("HSV image",hsv)

#HLS

hls=cv2.cvtColor(img,cv2.COLOR\_BGR2HLS)

cv2.imshow("HLS image",hls)

#YUV

yuv=cv2.cvtColor(img,cv2.COLOR\_BGR2YUV)

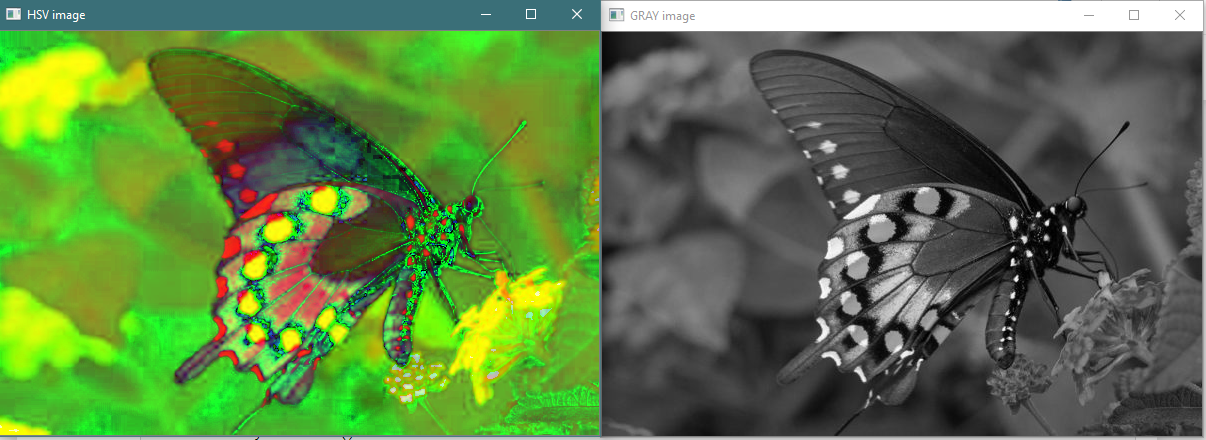
cv2.imshow("YUV image",yuv)

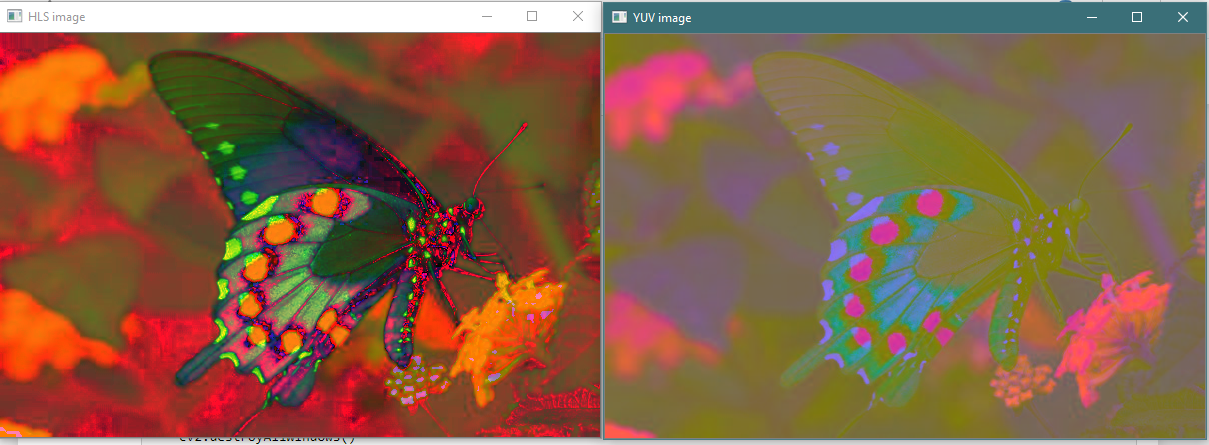
k=cv2.waitKey(0)

if k == 27:

cv2.destroyAllWindows()

**Output:**





**6.Develop a program to create an image from 2D array.**

**------------------------------------------------------------------------------------------------**

import numpy as np

from PIL import Image

import cv2

array = np.zeros([512, 256, 3], dtype=np.uint8)

#print((array))

array[:,:200] = [ 0,175, 120]

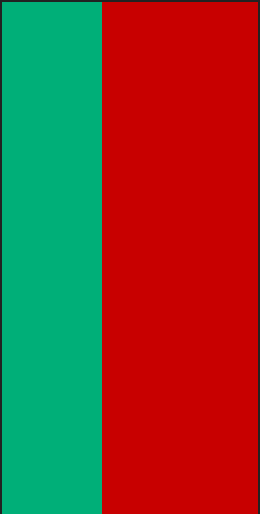
array[:,100:] = [200, 0,0 ]

img = Image.fromarray(array)

img.save('DIFFCOLOR.jpg')

img.show()

**Output:**



#7.Find the sum of all neighborhood values of the matrix.

import numpy as np

M = [[1, 2, 3],

[4, 5, 6],

[7, 8, 9]]

M = np.asarray(M)

N = np.zeros(M.shape)

def sumNeighbors(M,x,y):

l = []

for i in range(max(0,x-1),x+2): # max(0,x-1), such that no negative values in range()

for j in range(max(0,y-1),y+2):

try:

t = M[i][j]

l.append(t)

except IndexError: # if entry doesn&#39;t exist

pass

return sum(l)-M[x][y] # exclude the entry itself

for i in range(M.shape[0]):

for j in range(M.shape[1]):

N[i][j] = sumNeighbors(M, i, j)

print ("Original matrix:\n", M)

print ("Summed neighbors matrix:\n", N)

Output:

Original matrix:

[[1 2 3]

[4 5 6]

[7 8 9]]

Summed neighbors matrix:

[[11. 19. 13.]

[23. 40. 27.]

[17. 31. 19.]]