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

import cv2

image=cv2.imread("dog.jpg")

cv2.imshow('Original Image',image)

cv2.waitKey(0)

gray\_img = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imwrite('gray scale',gray\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()



**2.Develop a program to perform Linear Transformation on image(Scaling and Rotation).**

**Scaling:**

import cv2

import numpy as np

img=cv2.imread('dog.jpg')

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

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

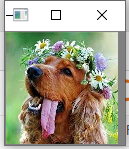
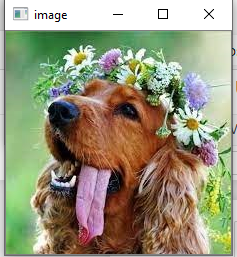
cv2.imwrite('result.jpg',res)

cv2.imshow('Result',res)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()



**Rotation:**

import cv2

import numpy as np

img=cv2.imread('dog.jpg')

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

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

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

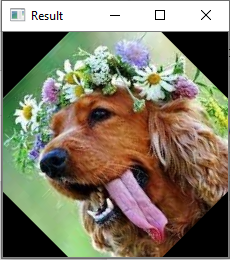
cv2.imwrite('result.jpg',res)

cv2.imshow('Result',res)

cv2.imshow('image',img)

cv2.waitKey(0)

cv2.destroyAllWindows()



**3. Develop a program to find sum and mean of a set of images.**

**(i)Create ‘n’ number of images and read the directory and perform**

**Operation.**

import cv2

import os

path ="D:\images"

imgs=[]

dirs=os.listdir(path)

for file in dirs:

fpath=path+"\\"+file

imgs.append(cv2.imread(fpath))

i=0

for im in imgs:

cv2.imshow(dirs[i],imgs[i])

i=i+1

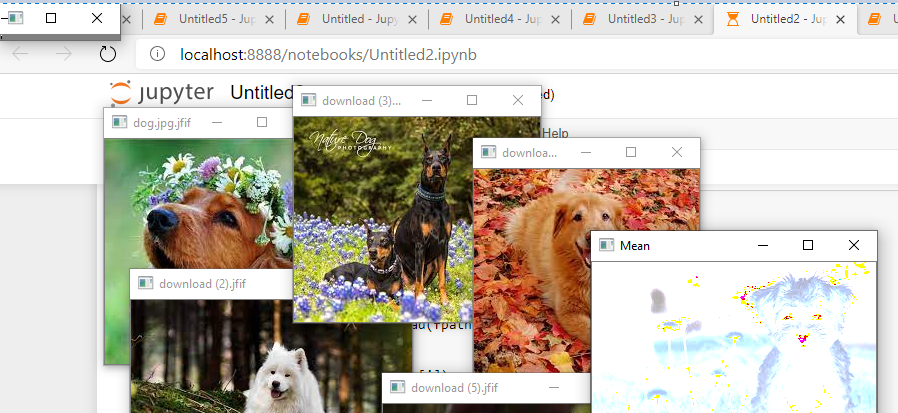
print(i)

cv2.imshow('Sum',im)

mean=im/len(dirs.)

cv2.imshow('Mean',mean)

cv2.waitKey()

****

**4.Convert color image to Gray scale and binary image**

import cv2

image=cv2.imread("dog.jpg")

gray\_img = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cv2.imshow('gray scale',gray\_img)

cv2.waitKey(0)

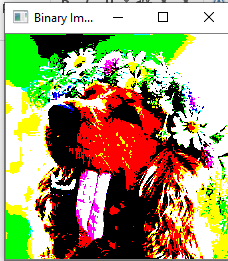
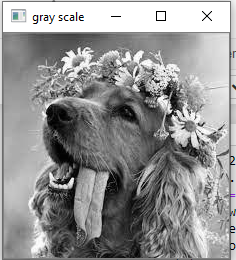
cv2.destroyAllWindows()

ret,bw\_image=cv2.threshold(image,127,255,cv2.THRESH\_BINARY)

cv2.imshow('Binary Image',bw\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()



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

import cv2

image=cv2.imread("dog.jpg")

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

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

lab=cv2.cvtColor(image,cv2.COLOR\_BGR2LAB)

cv2.imshow('GRAY IMAGE',gray)

cv2.waitKey(0)

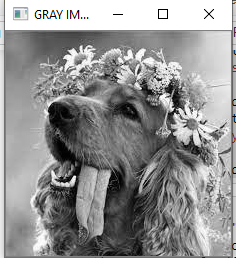
cv2.imshow('HSV iamge',hsv)

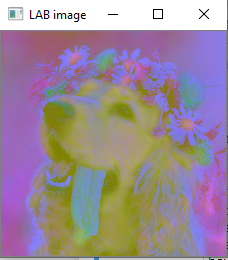
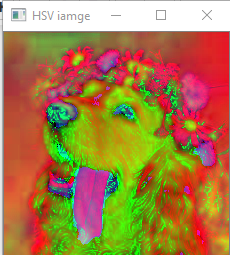
cv2.waitKey(0)

cv2.imshow('LAB image',lab)

cv2.waitKey(0)

cv2.destroyAllWindows()

****

****

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

**random size.**

import numpy as np

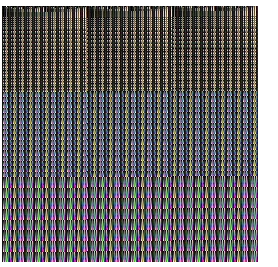
from PIL import Image

array=np.linspace(0,1,256\*256)

mat=np.reshape(array,(256,256))

img=Image.fromarray(mat,'HSV')

img.show()

****

**7.program to find the neighbor of matrix.**

X = [[1,2,3], [4 ,5,6], [7 ,8,9]]

Y = [[9,8,7], [6,5,4], [3,2,1]]

result = [[0,0,0], [0,0,0], [0,0,0]]

for i in range(len(X)):

for j in range(len(Y)):

result[i][j] = X[i][j] + Y[i][j]

print("Resultant array:")

for r in result:

print(r)

def neighbors(radius, rowNumber, columnNumber):

return [[result[i][j]

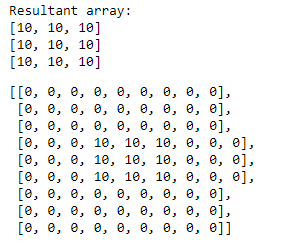
if i >= 0 and i < len(result) and j >= 0 and j < len(result[0]) else 0

for j in range(columnNumber-1-radius, columnNumber+radius)]

for i in range(rowNumber-1-radius, rowNumber+radius)]

neighbors(4,2,2)

**OUTPUT:**

****

**8. Program to find the Sum of neighbour value of 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):

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

try:

t = M[i][j]

l.append(t)

except IndexError:

pass

return sum(l)-M[x][y]

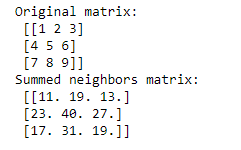
for i in range(M.shape[0]):

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)

****

**9. Operator Overloading in C++:Assignment operator of 2 Matrix.**

#include <iostream>

int findSum(int n)

{

// Generate matrix

int a[100][100],b[100][100];

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

std::cin>>a[i][j] ;

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

b[i][j]=a[i][j];

// Compute sum

int sum = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

sum += b[i][j];

std::cout <<"sum of elements: ";

return sum;

}

int main() {

int n = 3;

std::cout << findSum(n) ;

return 0;

}

**Ouput:**

1 2 3

4 5 6

7 8 9

sum of elements: 45

**Describe:**

**(i) Anaconda :**

Anaconda is a distribution of the Python and R programming languages for scientific computing , that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

Anaconda is popular because it brings many of the tools used in data science and machine learning with just one install, so it's great for having short and simple setup. Like Virtualenv, Anaconda also uses the concept of creating environments so as to isolate different libraries and versions.

**(ii)Spyder :**

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software.

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. ... Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

**(iii)Jupiter :**

The Jupyter Notebook application allows you to create and edit documents that display the input and output of a Python or R language script. Once saved, you can share these files with others. NOTE: Python and R language are included by default, but with customization, Notebook can run several other kernel environments.