

## LAB-02: Introduction to Python, OpenCV & Numpy

### Objective:

The objective of this lab is to introduce the student with OpenCV especially with respect to image processing.

### Some Useful Commands:

1. To create a 2D array of zeros using NumPy: **`my_array = numpy.zeros (( row, columns ))`**
2. To create a 2D array of ones using NumPy: **`my_array = numpy.ones (( row, columns ))`**
3. To check the size of a 2D array: **`size = numpy.shape(my_array)`**
4. Reading an image using OpenCV: **`my_image = cv2.imread("test_image.jpg",0)`** The second argument determines whether the image is read as a grayscale image or a colored image. 0 is used for reading an image as grayscale and while 1 is used for reading in color. If no argument is passed then the image is read as is.
5. Displaying an image using OpenCV: **`cv2.imshow ("Title of the window", my_image)`**. Two more commands that need to be used while displaying an image are: `cv2.waitKey(x)` `cv2.destroyAllWindows()`. The `waitKey()` function waits for a key being pressed for x number of milliseconds. If 0 is passed to `waitKey()` as an argument, it will wait indefinitely for a key press. `cv2.destroyAllWindows()` closes all the open image windows.
6. Writing an image to disk: **`cv2.imwrite("image_name.jpg", my_image)`**
7. Resizing an image: **`cv2.resize(my_image, (new_height,new_width))`**

### Lab Tasks:

#### Lab Task 1:

Read any image that you want using the right command and display it.

#### Lab Task 2:

a) Now mirror the image that you have read at center i.e. the lower half of the image should be the copy of the upper half. (HINT: You can use nested loops). Write the image to the disk using the right command.

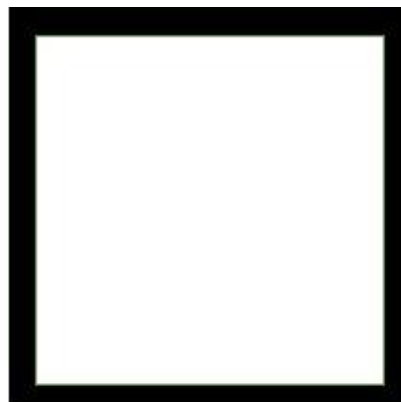


b) Now completely flip the image so that the image is upside down as shown below:



### Lab Task 3:

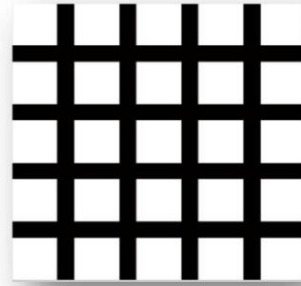
Create an  $r \times c$  matrix of ones and pad 10 pixels wide border of zeros across each side of it, such that its order will become  $(4+500+4) \times (4+500+4) = 508 \times 508$  as shown below:



Create a generic function so that the values (500 and 4) can be passed by the user.

### Lab Task 4:

Write 3 different Python functions that can create the images given below. Code them in such so that the size of the image itself and the boxes and lines can be changed.



### Lab Task 5:

Read an image and resize it to 512x512 using the appropriate function. Then down sample the image by 4 so that the final size of the image is 128x128. Display and save the image to the disk.

### Lab Task 6:

Write a function to create a white image of 500x500 (or any other size entered by the user) and then create 4 boxes of Red, Green, Blue and Black respectively on each corner of the image as shown below. The size of the colored boxes should be  $1/8^{\text{th}}$  the size of the image. (HINT: the arrays of ones and zeros can be in more than 2 dimensions)

