1. Introduction of Computer vision

- Computer vision is a field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos
- NOTE : for better understanding go through notes

Difference between Human vision system and Computer vision system

- Human vision uses eyes and brain to see and understand, while computer vision uses a camera and computer to detect and recognize objects

2. What is Image ?

- A real image will represent as an array & that image will convert into pixel
- Pixel range between (0 - 255), where 0 = dark value, 255 = Highest / Brightest value
- Every pixel in image is stored in value between (0-255)
- Coloured image will be represented as RGB (Red, Green, Blue)
- Channels in image : 2D channel -> black and white || 3D channel -> Red, Green, Blue (RGB)

3. Numpy & Image connection

- Image reading with Numpy & Matplotlib

In [1]:
```
# NumPy is a Python library used to work with numbers and arrays
import numpy as np
```

In [2]:
```
arr_of_ones = np.ones((5,5))
```

In [3]:
```
arr_of_ones
```

Out[3]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [4]:
```
# changing the values datatype
arr_of_ones = np.ones((5,5), dtype = int)
arr_of_ones
```

Out[4]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [5]:
```
# changing array values to 255
arr_of_ones * 255
```

Out[5]:
```
array([[255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255],
       [255, 255, 255, 255, 255]])
```

In [6]:
```
arr_of_zeros = np.zeros((3,3), dtype = int)
arr_of_zeros
```

Out[6]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [7]:
```
arr_of_ones
```

Out[7]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [8]:
```
# Matplotlib is used for visualization
import matplotlib.pyplot as plt
```

In [9]:
```
# It tells Jupyter to display plots directly inside the notebook (just below the
%matplotlib inline
```

In [10]:
```
# It lets you open, edit, and save images (like JPEG, PNG, etc.) using the Pillo
from PIL import Image
```

In [11]:
```
# load the image by using the path location
eye = Image.open(r"C:\Users\chara\OneDrive\Desktop\colorful eye.jpg") # r = raw
```

In [12]:
```
eye
```

Out[12]:

In [13]:
```
type(eye) #type of image
```

Out[13]:
```
PIL.JpegImagePlugin.JpegImageFile
```

In [14]:
```
# representing image in an array
eye_arr = np.asarray(eye)
eye_arr
```

Out[14]:
```
array([[[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [226, 141, 121],
        [226, 141, 112],
        [211, 123,  85]],

       [[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [231, 138,  81],
        [233, 139,  77],
        [225, 132,  65]],

       [[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [247, 141,  41],
        [249, 143,  43],
        [247, 144,  43]],

       ...,

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 16,  15,  21],
        ...,
        [253, 158, 100],
        [239, 147, 106],
        [223, 135, 111]],

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 17,  16,  22],
        ...,
        [255, 151,  94],
        [240, 142, 107],
        [221, 129, 114]],

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 17,  16,  22],
        ...,
        [252, 148,  95],
        [237, 142, 114],
        [219, 131, 127]]], dtype=uint8)
```

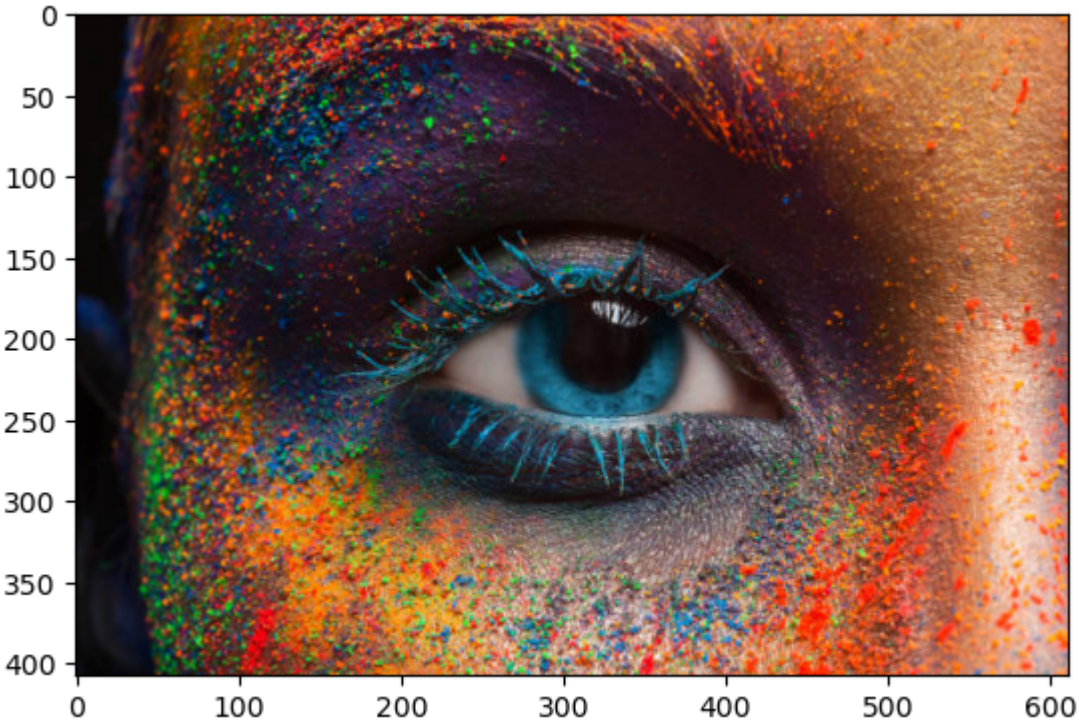
In [15]:
```
type(eye_arr) # tyepe of arr
```

Out[15]:
```
numpy.ndarray
```

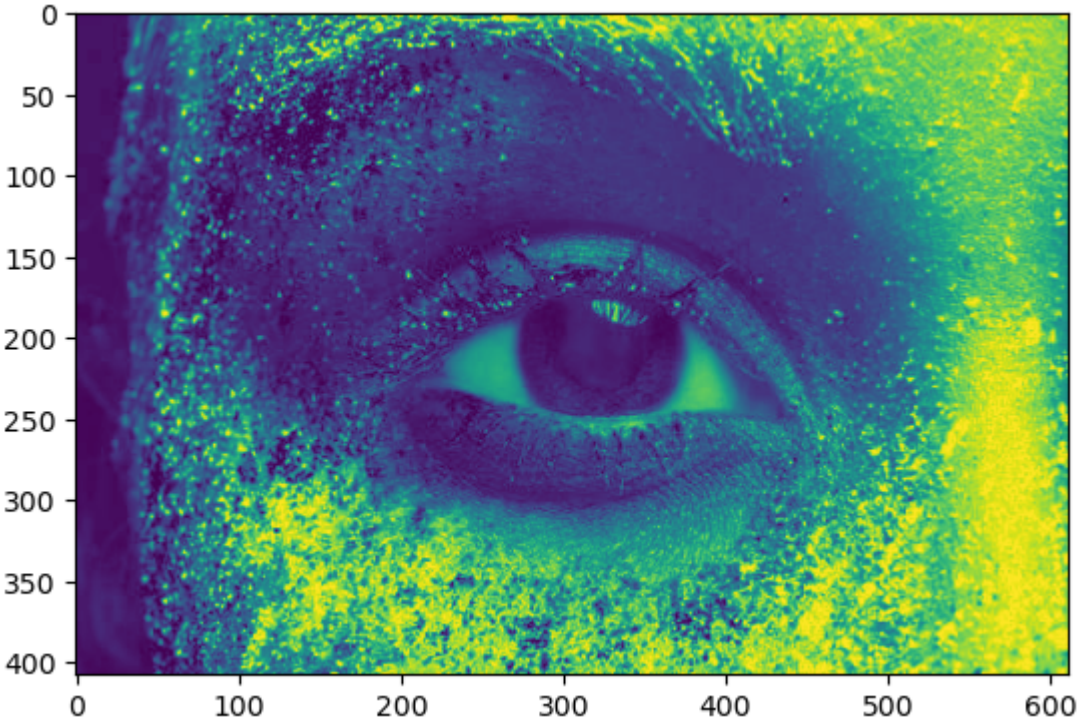
In [16]:
```
# finding the dimensions of an image / shape
eye_arr.shape # (height, width, 3D channel)
```

Out[16]:
```
(408, 612, 3)
```

In [17]:
```
# showing image with dimensions
plt.imshow(eye_arr)
plt.show()
```



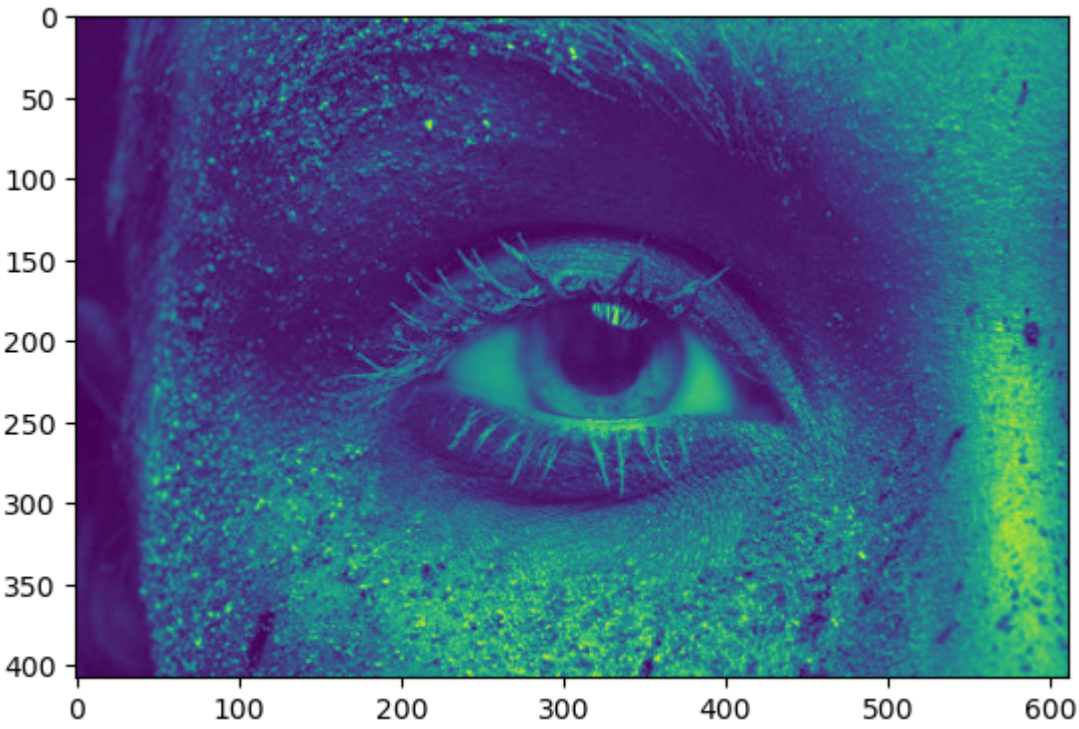
In [18]:
```
# eye_arr[:, :, 0] means: Take all rows (:) , Take all columns (:) , Take only t
plt.imshow(eye_arr[:,:,0]) # 0 = red channel
plt.show()
```



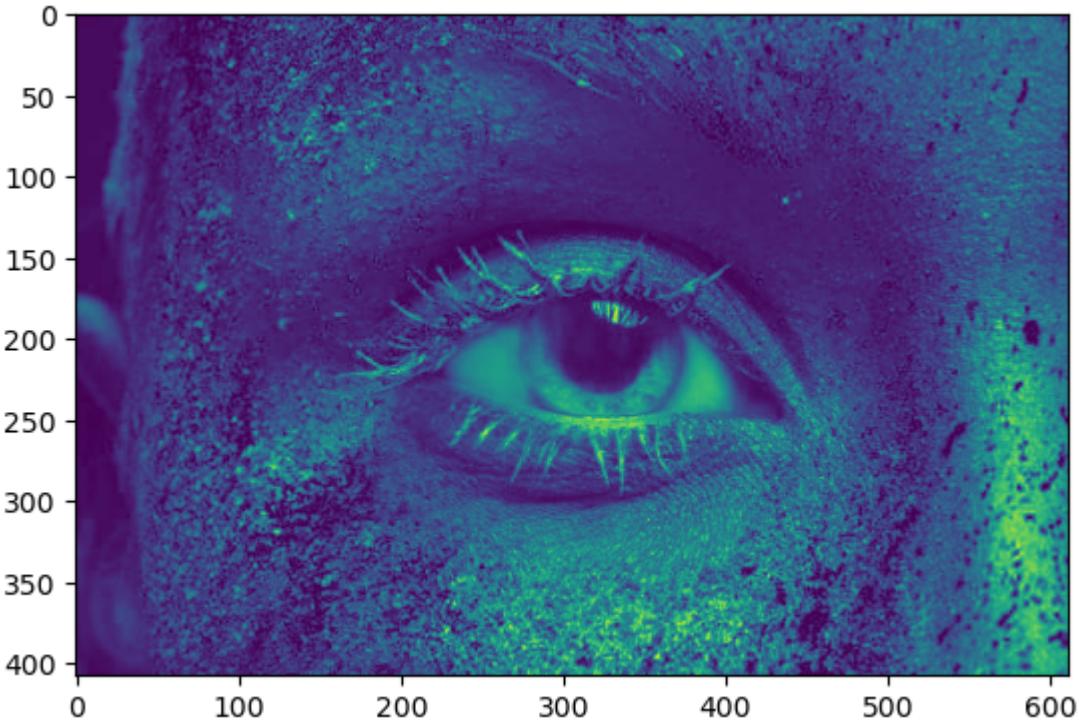
In [19]:
```
eye_arr[:,:,0]
```

Out[19]:
```
array([[ 12,  12,  12, ..., 226, 226, 211],
       [ 12,  12,  12, ..., 231, 233, 225],
       [ 12,  12,  12, ..., 247, 249, 247],
       ...,
       [ 19,  18,  16, ..., 253, 239, 223],
       [ 19,  18,  17, ..., 255, 240, 221],
       [ 19,  18,  17, ..., 252, 237, 219]], dtype=uint8)
```

In [20]:
```
plt.imshow(eye_arr[:,:,1]) # 1 = green channel
plt.show()
```



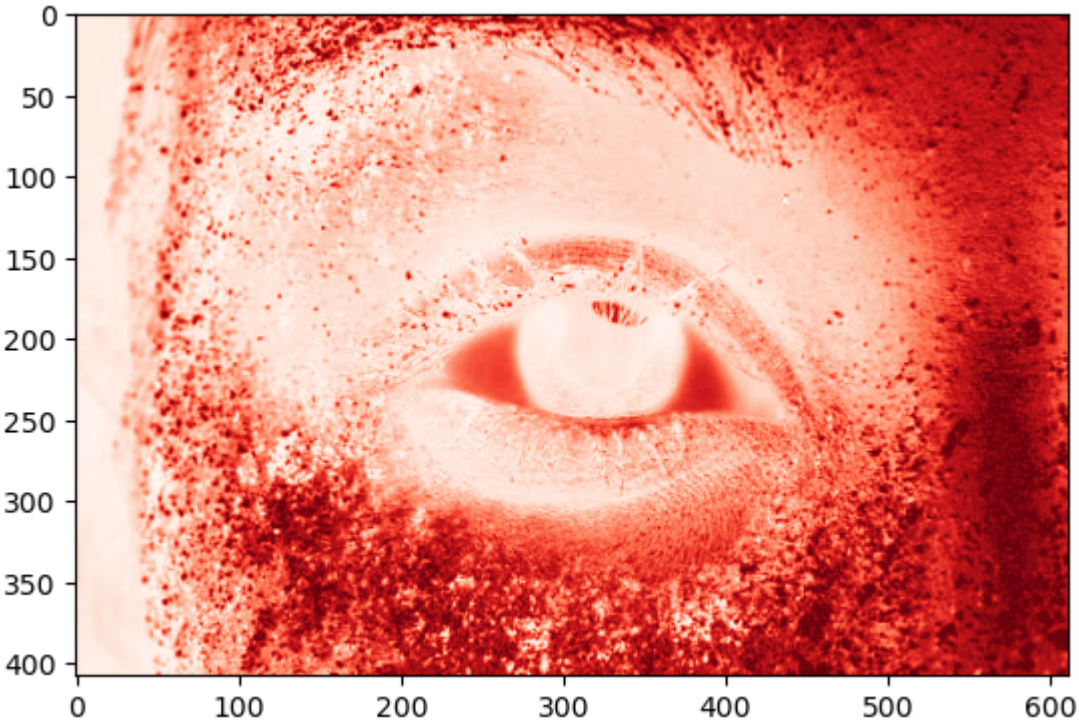
In [21]:
```
plt.imshow(eye_arr[:,:,2]) # 2 = blue channel
plt.show()
```



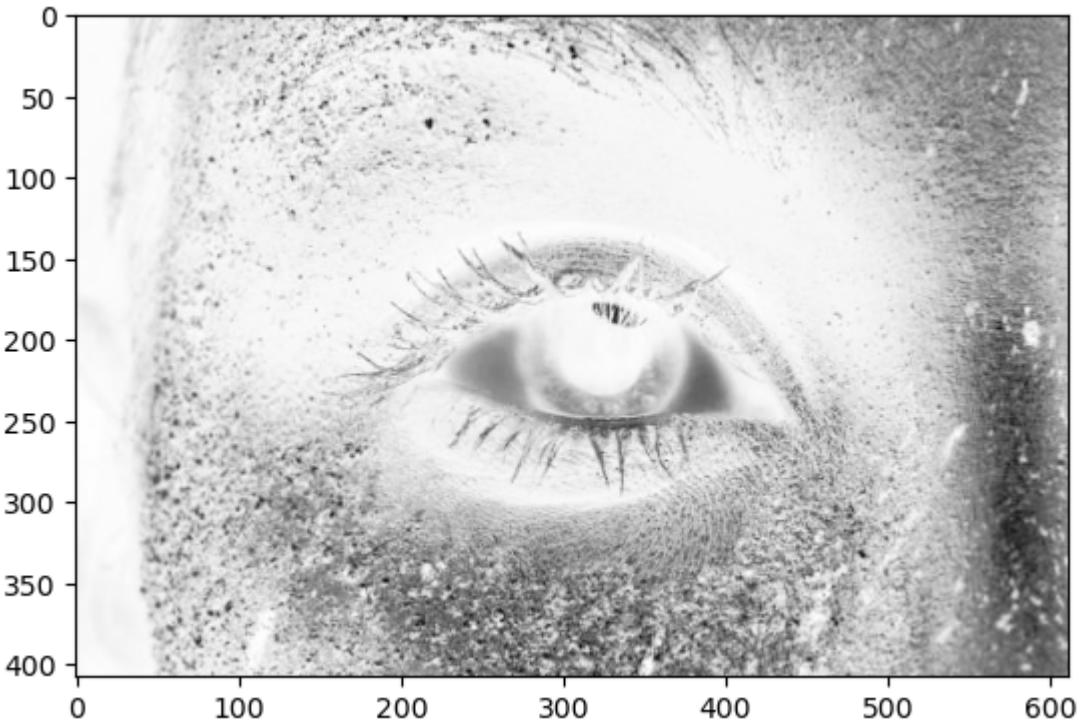
Colormaps in Matplotlib Documentation

https://matplotlib.org/stable/users/explain/colors/colormaps.html

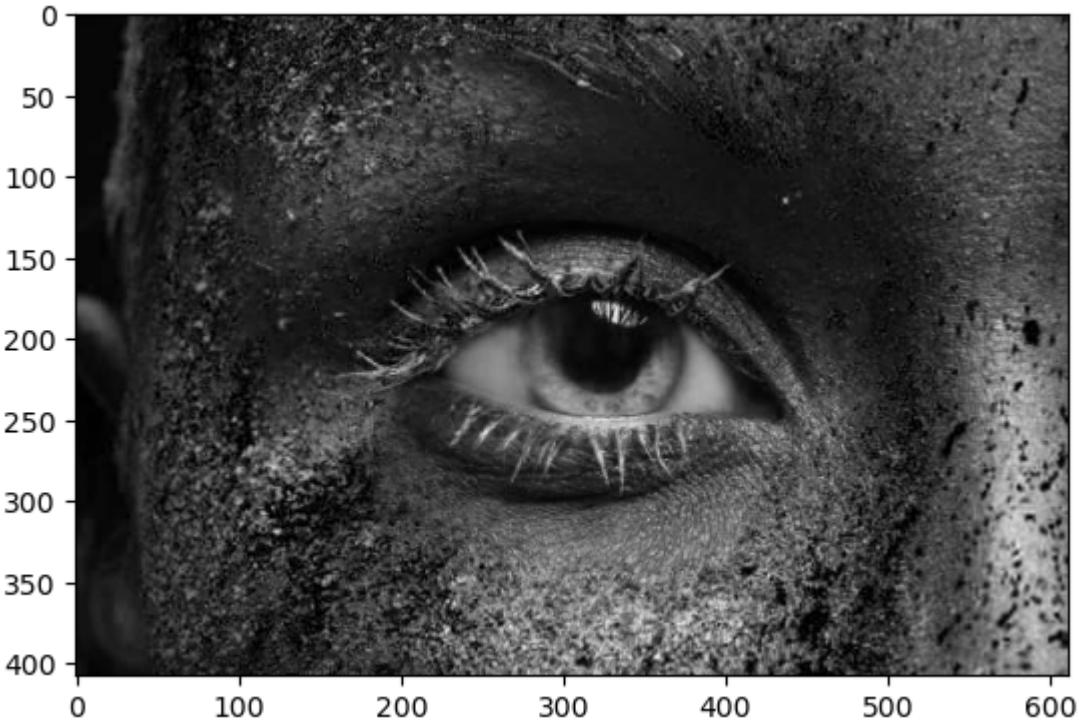
In [22]:
```
plt.imshow(eye_arr[:,:,0], cmap = 'Reds')
plt.show()
```



In [23]:
```
plt.imshow(eye_arr[:,:,1], cmap = 'Greys')
plt.show()
```



In [24]:
```
plt.imshow(eye_arr[:,:,2], cmap = 'gray')
plt.show()
```



In [25]:
```
eye_arr[:,:,0] # array of image with red channel
```

Out[25]:
```
array([[ 12,  12,  12, ..., 226, 226, 211],
       [ 12,  12,  12, ..., 231, 233, 225],
       [ 12,  12,  12, ..., 247, 249, 247],
       ...,
       [ 19,  18,  16, ..., 253, 239, 223],
       [ 19,  18,  17, ..., 255, 240, 221],
       [ 19,  18,  17, ..., 252, 237, 219]], dtype=uint8)
```

In [26]:
```
eye_arr[:,:,1] # array of image with Green channel
```

Out[26]:
```
array([[  6,   6,   6, ..., 141, 141, 123],
       [  6,   6,   6, ..., 138, 139, 132],
       [  6,   6,   6, ..., 141, 143, 144],
       ...,
       [ 18,  17,  15, ..., 158, 147, 135],
       [ 18,  17,  16, ..., 151, 142, 129],
       [ 18,  17,  16, ..., 148, 142, 131]], dtype=uint8)
```

In [27]:
```
eye_arr[:,:,2] # array of image with Blue channel
```

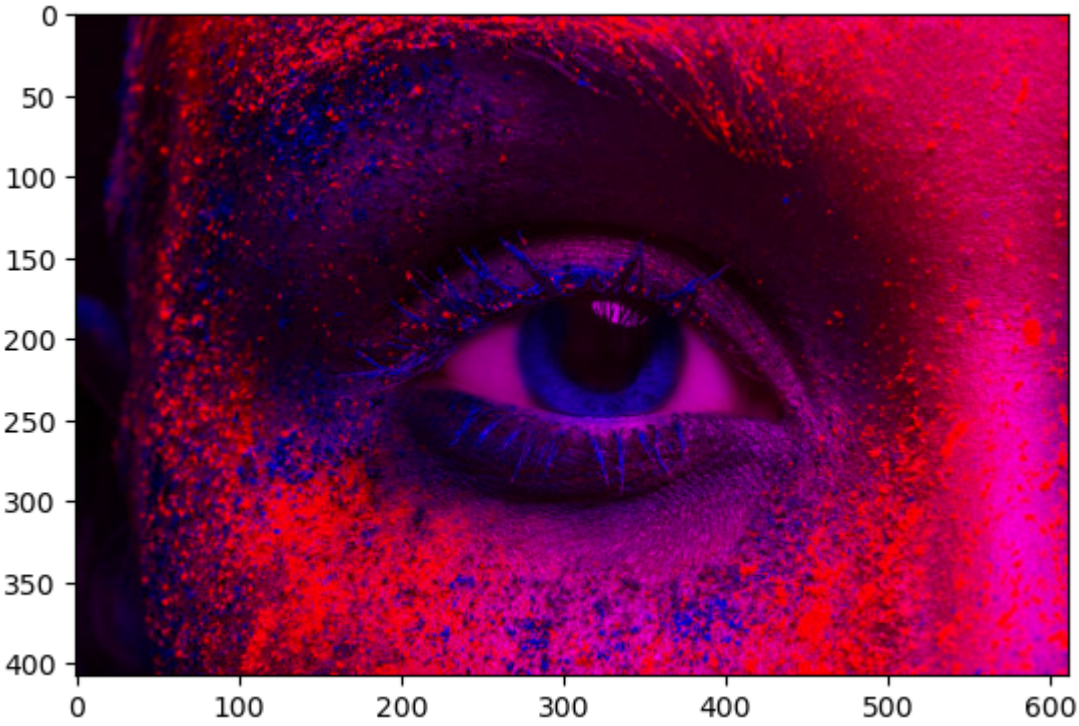
Out[27]:
```
array([[  6,   6,   6, ..., 121, 112,  85],
       [  6,   6,   6, ...,  81,  77,  65],
       [  6,   6,   6, ...,  41,  43,  43],
       ...,
       [ 24,  23,  21, ..., 100, 106, 111],
       [ 24,  23,  22, ...,  94, 107, 114],
       [ 24,  23,  22, ...,  95, 114, 127]], dtype=uint8)
```

In [28]:
```
eye_img1 = eye_arr.copy()
eye_img1
```

Out[28]:
```
array([[[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [226, 141, 121],
        [226, 141, 112],
        [211, 123,  85]],

       [[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [231, 138,  81],
        [233, 139,  77],
        [225, 132,  65]],

       [[ 12,   6,   6],
        [ 12,   6,   6],
        [ 12,   6,   6],
        ...,
        [247, 141,  41],
        [249, 143,  43],
        [247, 144,  43]],

       ...,

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 16,  15,  21],
        ...,
        [253, 158, 100],
        [239, 147, 106],
        [223, 135, 111]],

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 17,  16,  22],
        ...,
        [255, 151,  94],
        [240, 142, 107],
        [221, 129, 114]],

       [[ 19,  18,  24],
        [ 18,  17,  23],
        [ 17,  16,  22],
        ...,
        [252, 148,  95],
        [237, 142, 114],
        [219, 131, 127]]], dtype=uint8)
```

In [29]:
```
eye_img1[:,:,1] = 0
```

In [30]:
```
plt.imshow(eye_img1)
plt.show()
```



4. Open CV & Image processing

- OpenCV (Open Source Computer Vision Library) is a Python library used for image and video processing, computer vision, and machine learning.
- Open CV libraries are written in C++ programming language
- It's fast and widely used in real-world applications like face detection, object tracking, and image filtering.

In [31]:
```
# insatlling package
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\chara\anaconda3\lib\site
-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\chara\anaconda3\lib\site
-packages (from opencv-python) (1.26.4)
```

In [32]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [33]:
```
# cv2 is the Python module for OpenCV, a powerful library used for image and vid
import cv2
```

In [34]:
```
# reading the image in the form of array
img = cv2.imread(r"C:\Users\chara\OneDrive\Desktop\images.jpg")
```

In [35]:
```
img
```

Out[35]:
```
array([[[ 96,  72,  36],
        [102,  76,  40],
        [109,  83,  47],
        ...,
        [ 23,  12,  14],
        [ 23,  12,  14],
        [ 23,  12,  14]],

       [[ 98,  72,  36],
        [102,  76,  40],
        [110,  82,  47],
        ...,
        [ 24,  13,  15],
        [ 23,  12,  14],
        [ 23,  12,  14]],

       [[ 99,  73,  37],
        [104,  76,  41],
        [110,  82,  47],
        ...,
        [ 25,  14,  16],
        [ 24,  13,  15],
        [ 24,  13,  15]],

       ...,

       [[ 52, 153,  75],
        [ 42, 144,  67],
        [ 24, 128,  57],
        ...,
        [ 83, 160, 122],
        [ 83, 152, 119],
        [118, 176, 152]],

       [[ 46, 148,  71],
        [ 33, 135,  63],
        [ 14, 119,  52],
        ...,
        [ 56, 152, 105],
        [ 51, 137,  97],
        [102, 173, 146]],

       [[ 37, 139,  68],
        [ 23, 126,  59],
        [  7, 110,  49],
        ...,
        [ 71, 169, 115],
        [ 47, 134,  90],
        [ 97, 165, 134]]], dtype=uint8)
```

In [36]:
```
# type of image
type(img)
```

Out[36]:
```
numpy.ndarray
```

In [37]:
```
# dimensions of a image
img.shape
```

Out[37]:
```
(183, 275, 3)
```

In [38]:
```
plt.imshow(img)
plt.show()
```

- Here we can see the picture generates with some diffrent colour compares to original which i have downloaded
- Bcoz, Matplotlib -> RGB | Open CV -> BGR

OpenCV Color Space Conversions Documentation

https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html

cvtColor()

- cvtColor() - is a function in OpenCV used to convert an image from one color space to another.

In [39]:
```
# Now, here we're fixing the colour of the image to original
fix_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [40]:
```
fix_img
```

Out[40]:
```
array([[[ 36,  72,  96],
        [ 40,  76, 102],
        [ 47,  83, 109],
        ...,
        [ 14,  12,  23],
        [ 14,  12,  23],
        [ 14,  12,  23]],

       [[ 36,  72,  98],
        [ 40,  76, 102],
        [ 47,  82, 110],
        ...,
        [ 15,  13,  24],
        [ 14,  12,  23],
        [ 14,  12,  23]],

       [[ 37,  73,  99],
        [ 41,  76, 104],
        [ 47,  82, 110],
        ...,
        [ 16,  14,  25],
        [ 15,  13,  24],
        [ 15,  13,  24]],

       ...,

       [[ 75, 153,  52],
        [ 67, 144,  42],
        [ 57, 128,  24],
        ...,
        [122, 160,  83],
        [119, 152,  83],
        [152, 176, 118]],

       [[ 71, 148,  46],
        [ 63, 135,  33],
        [ 52, 119,  14],
        ...,
        [105, 152,  56],
        [ 97, 137,  51],
        [146, 173, 102]],

       [[ 68, 139,  37],
        [ 59, 126,  23],
        [ 49, 110,   7],
        ...,
        [115, 169,  71],
        [ 90, 134,  47],
        [134, 165,  97]]], dtype=uint8)
```

In [41]:
```
# here we converted BGR to RGB
plt.imshow(fix_img)
plt.show()
```

cv2.IMREAD_GRAYSCALE

- cv2.IMREAD_GRAYSCALE is a flag used with the cv2.imread() function to load an image in grayscale.
- Grayscale refers to an image that contains only shades of gray, ranging from black to white. It doesn't have color (i.e., no red, green, or blue), just varying intensities of light.

In [42]:
```
img_gray = cv2.imread(r"C:\Users\chara\OneDrive\Desktop\images.jpg", cv2.IMREAD_
```

In [43]:
```
img_gray
```

Out[43]:
```
array([[ 64,  68,  75, ...,  14,  14,  14],
       [ 64,  68,  75, ...,  15,  14,  14],
       [ 65,  69,  75, ...,  16,  15,  15],
       ...,
       [118, 109,  95, ..., 140, 134, 162],
       [113, 102,  87, ..., 127, 115, 157],
       [106,  94,  80, ..., 142, 111, 148]], dtype=uint8)
```

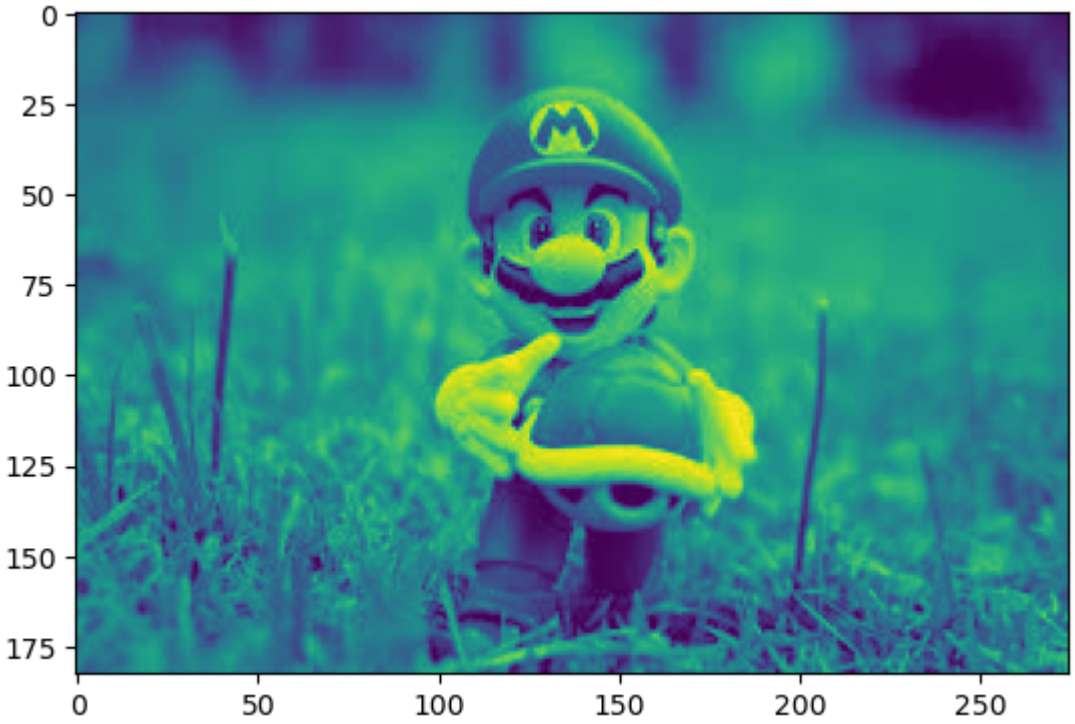
In [44]:
```
img_gray.min() # min value
```

Out[44]:
```
0
```

In [45]:
```
img_gray.max() # max value
```

Out[45]:
```
255
```

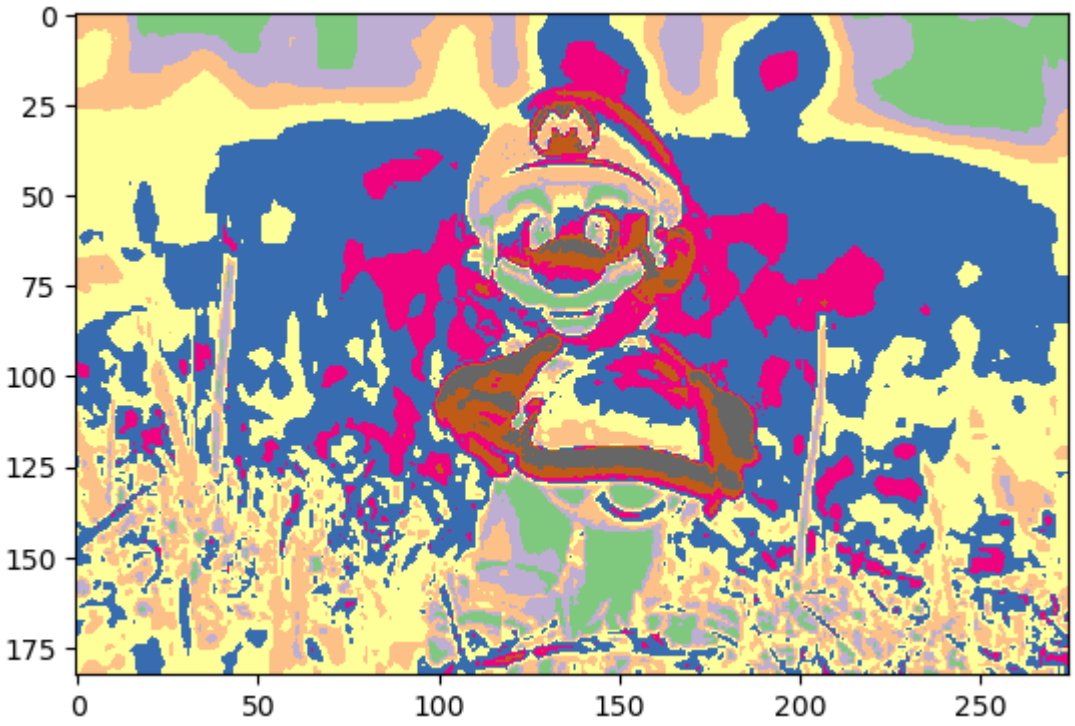
In [46]:
```
plt.imshow(img_gray) # used GRAYSCALE Flag for this output
plt.show()
```



In [47]:
```
plt.imshow(img_gray, cmap = 'gray') # gray is a color
plt.show()
```

In [48]:
```
plt.imshow(img_gray, cmap = 'magma') # magma is a color
plt.show()
```

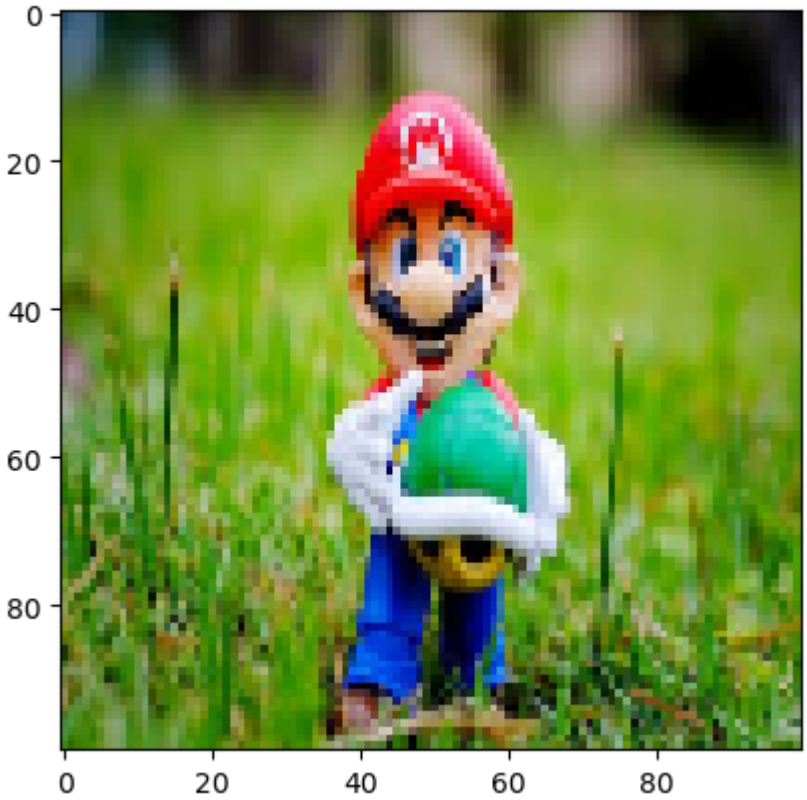
In [49]:
```
plt.imshow(img_gray, cmap = 'Accent') # Accent is color
plt.show()
```



In [50]:
```
fix_img2 = cv2.resize(fix_img,(100,100))
fix_img2
```

Out[50]:
```
array([[[ 39,  75, 101],
        [ 57,  92, 124],
        [ 65,  99, 136],
        ...,
        [ 24,  23,  31],
        [ 16,  15,  23],
        [ 14,  12,  23]],

       [[ 41,  76, 105],
        [ 57,  92, 123],
        [ 63,  97, 131],
        ...,
        [ 24,  23,  31],
        [ 18,  17,  25],
        [ 15,  13,  24]],

       [[ 49,  84, 116],
        [ 62,  97, 127],
        [ 65, 100, 130],
        ...,
        [ 24,  23,  31],
        [ 21,  20,  28],
        [ 18,  16,  27]],

       ...,

       [[ 75, 144,  62],
        [ 65, 136,  33],
        [ 75, 133,  17],
        ...,
        [137, 167, 135],
        [150, 183, 131],
        [160, 185, 153]],

       [[ 69, 145,  45],
        [ 54, 117,   7],
        [ 71, 126,   5],
        ...,
        [127, 167,  99],
        [111, 159,  69],
        [131, 162,  99]],

       [[ 62, 131,  29],
        [ 49, 107,   0],
        [ 61, 124,   1],
        ...,
        [ 61, 118,  16],
        [ 50, 103,  12],
        [ 99, 139,  55]]], dtype=uint8)
```

In [51]:
```
plt.imshow(fix_img2)
plt.show()
```



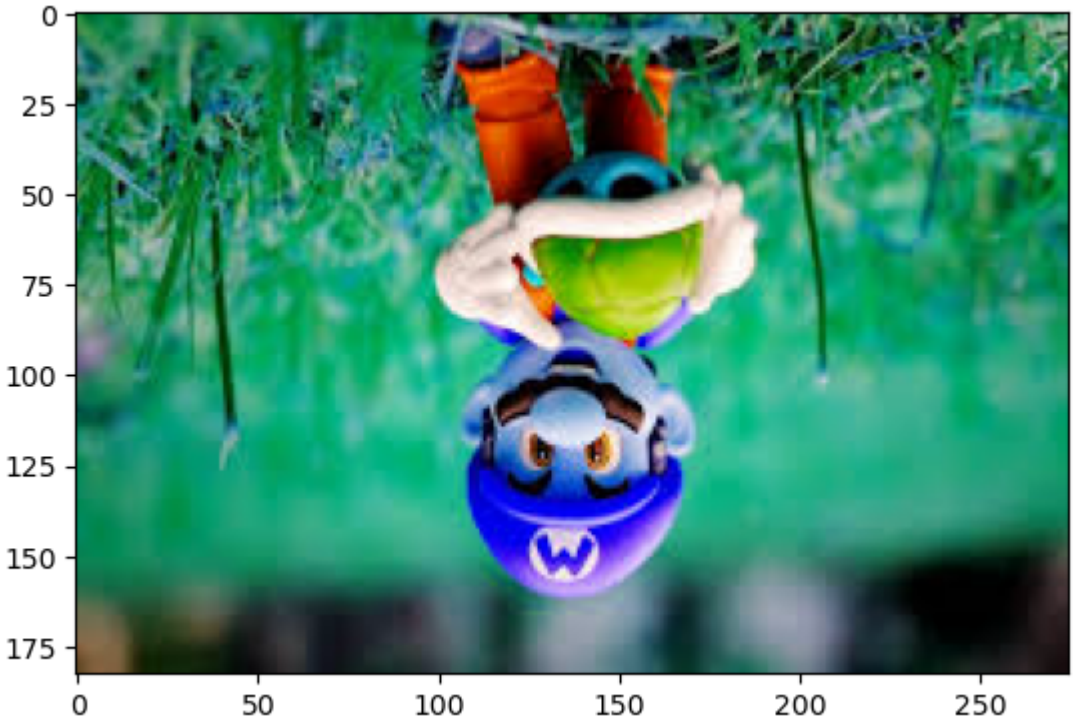
In [52]:
```
fix_img2.shape
```

Out[52]:
```
(100, 100, 3)
```

cv2.flip

- cv2.flip() is a function in OpenCV used to flip an image either vertically, horizontally, or both.

In [53]:
```
img2 = cv2.flip(img, 0) # 0 = flips the image vertically (upside down)
plt.imshow(img2)
plt.show()
```



In [54]:
```
img2 = cv2.flip(img, 1) # 1 =  flips the image horizontally, meaning it mirrors
plt.imshow(img2)
plt.show()
```

In [55]:
```
# saving this image
cv2.imwrite("Super mario.jpg",img2)
```

Out[55]: True

Image processing using OpenCV with out Matplotlib

In [56]:
```
import cv2
```

In [57]:
```
btfy = cv2.imread(r"C:\Users\chara\OneDrive\Desktop\butterfly.jpg")
```

In [58]:
```
btfy
```

Out[58]:
```
array([[[255, 255, 254],
        [253, 251, 250],
        [255, 255, 254],
        ...,
        [255, 254, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[252, 250, 249],
        [255, 255, 254],
        [255, 255, 254],
        ...,
        [255, 254, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[247, 245, 244],
        [254, 252, 251],
        [246, 247, 245],
        ...,
        [238, 240, 241],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

- NOTE : even though the folder is empty cv2.imread will execute and returns nothing (doesn't throw an error)

cv2.waitKey()

-cv2.waitKey() is a function that waits for a key press from the user while displaying an image or video frame.

In [ ]:
```
# returns the image in another window, with the "Butterfly" Frame name
cv2.imshow("Butterfly1", btfy)
cv2.waitKey()
```

In [ ]:
```
'''btfy = cv2.imread(r"C:\Users\chara\OneDrive\Desktop\butterfly.jpg")

while True:
    cv2.imshow("Butterfly2", btfy)
    if cv2.waitKey(1) & 0xFF == 27 :
        break
cv2.destroyAllWindows()'''
```

5. Video Detection Using OpenCV

In [ ]:
```
import cv2
cap = cv2.VideoCapture(0) # 0 → Default webcam (built-in camera)
width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)

while True:
    ret, frame = cap.read()
    cam = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    frame = cv2.flip(frame, 2)
    cv2.imshow('frame', cam)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

VideoCapture()

- cv2.VideoCapture() is an OpenCV function used to capture video from a camera, a video file, or even a stream.

get()

- get() is a method used with cv2.VideoCapture to retrieve properties of a video or webcam, such as frame width, height, FPS, position, etc.

cap.release()

- cap.release() is a method used in OpenCV to safely release the video capture object (usually created with cv2.VideoCapture()).

cv2.destroyAllWindows()

- cv2.destroyAllWindows() is an OpenCV function used to close all image or video windows that were opened using cv2.imshow().

In [ ]: