OpenCV - Part 1

OpenCV is a cross-platform library aimed for computer vision applications. It handles image processing, video capturing and analysis features like face detection and object detection

This note book contains various commands to achieve the following using opencv

1. Various ways of reading and showing the Images (PIL,Matplotlib,Opencv)
2. Seperate the display channels(RGB) and display the images
3. Resize the image using opencv
4. Flipping the images
5. Drawing boxes, lines on the images
6. Writing text over the images

In [71]:
```
from PIL import Image
import cv2
import numpy as np
```

Display an Image

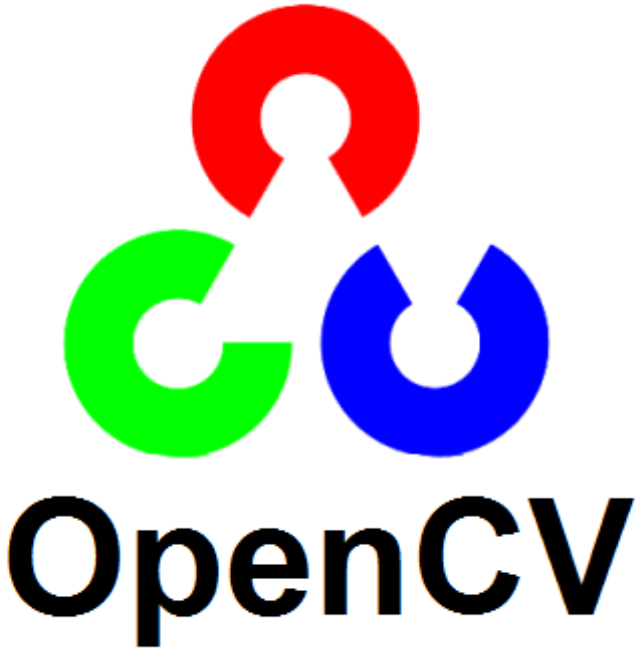
First let us see how to display image using various libraries

1. PIL
2. Matplotlib
3. OpenCv

1.Use PIL to show the image

In [72]:
```
# PIL is python imaging Library
Image.open('open_cv.png')
```

Out[72]:



2. Use numpy array & matplot lib to open an image

In [73]:
```
pup = Image.open('pup.jpg')
pup_arr = np.array(pup)
```

In [74]:
```
#using matplot lib to display an image
import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow(pup_arr);
```

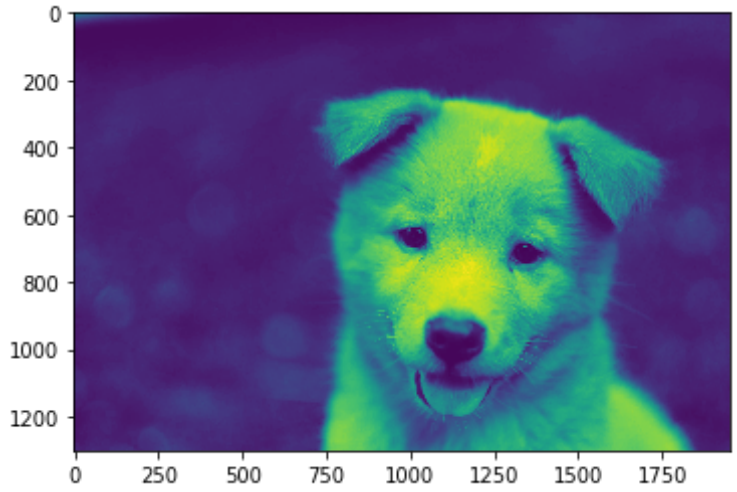
use matplot lib to show color channels

In [75]:
```
print(pup_arr.shape)
```

```
(1300, 1950, 3)
```

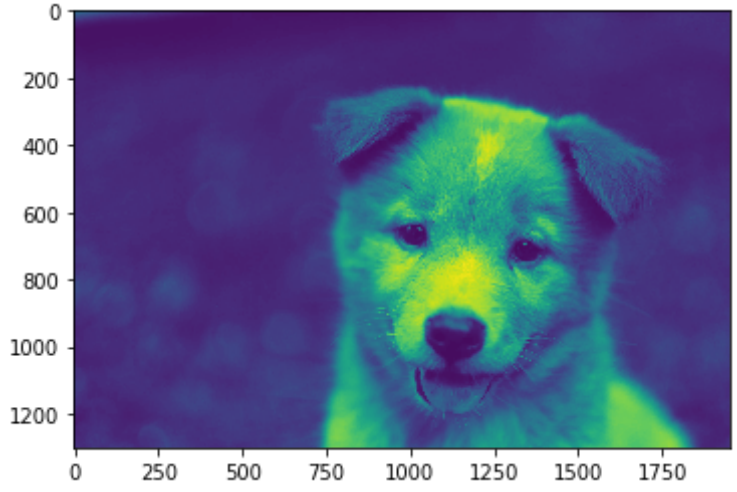
In [76]:
```
r_channel = pup_arr[:,:,0]
plt.imshow(r_channel)
```

Out[76]:
```
<matplotlib.image.AxesImage at 0x1c600f3f0b8>
```



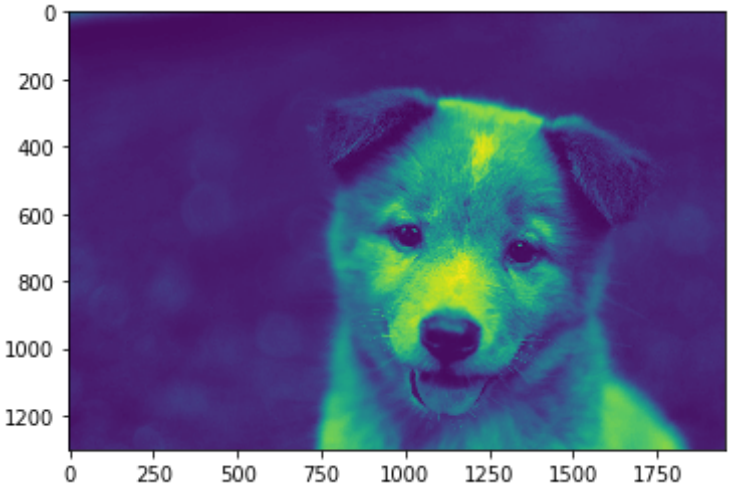
In [77]:
```
g_channel = pup_arr[:,:,1]
plt.imshow(g_channel)
```

Out[77]: <matplotlib.image.AxesImage at 0x1c602383da0>



In [78]:
```
b_channel = pup_arr[:,:,2]
plt.imshow(b_channel)
```

Out[78]: <matplotlib.image.AxesImage at 0x1c60091a908>



3. Use OpenCV to show Image

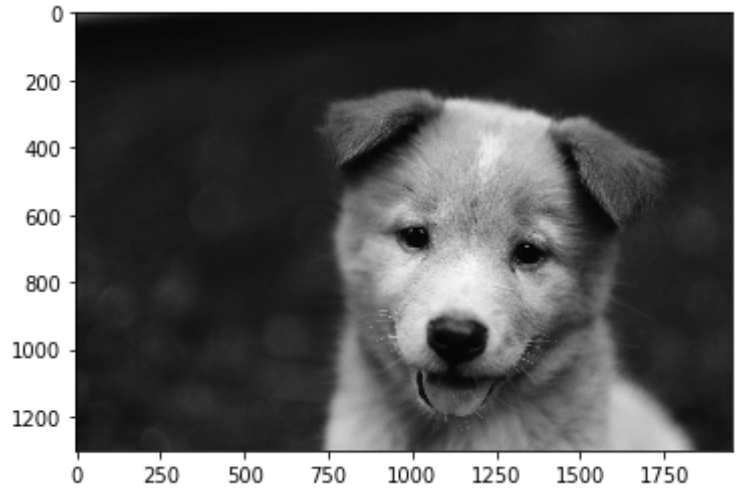
In [79]:
```
img = cv2.imread('pup.jpg')
plt.imshow(img);
```

Matplot lib reads the color channels in the order RGB where as the cv2 considers in the order of BGR. That's the reason the above image is looks decolored So when using opencv to read image and matplotlib to display the image should be coverted to RGB order as shown below and it displays the image with original color

In [80]:
```
img_convtd = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img_convtd);
```

Display gray scale image using opencv

In [81]:
```
img_gray = cv2.imread('pup.jpg',cv2.IMREAD_GRAYSCALE)
plt.imshow(img_gray,cmap='gray');
```



Gray scale will not have the channel dimension

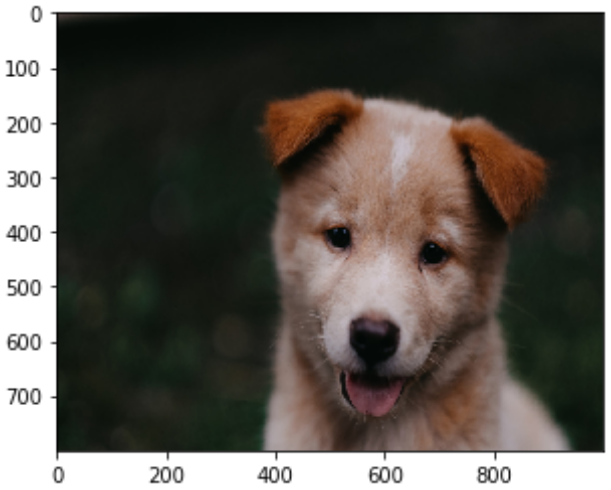
In [82]:
```
img_gray.shape
```

Out[82]:
```
(1300, 1950)
```

Image Resize

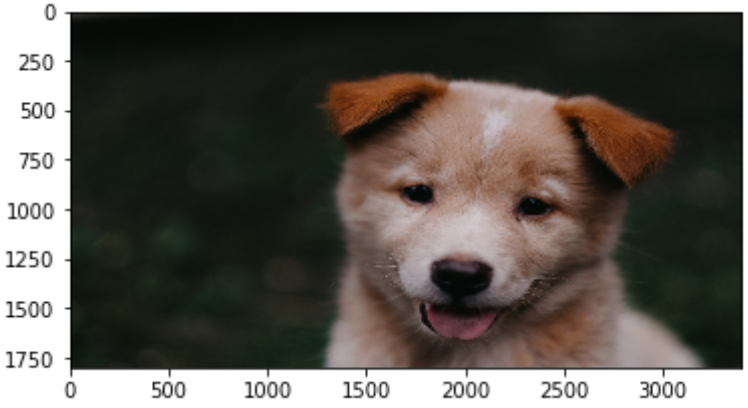
In [83]:
```
pup1 = cv2.imread('pup.jpg')
pup1 = cv2.cvtColor(pup1,cv2.COLOR_BGR2RGB)
print("Shape before resize -- > ",pup1.shape)
pup1_resized = cv2.resize(pup1,(1000,800))
print("Shape after resize -- >",pup1_resized.shape)
plt.imshow      (pup1_resized);
```

```
Shape before resize -- >  (1300, 1950, 3)
Shape after resize -- > (800, 1000, 3)
```



In [84]:
```
pup1 = cv2.imread('pup.jpg')
pup1 = cv2.cvtColor(pup1,cv2.COLOR_BGR2RGB)
print("Shape before resize -- > ",pup1.shape)
pup1_resized = cv2.resize(pup1,(3400,1800))
print("Shape after resize -- >",pup1_resized.shape)
plt.imshow      (pup1_resized);
```

```
Shape before resize --  >  (1300, 1950, 3)
Shape after resize -- > (1800, 3400, 3)
```



Flip Image

In [85]:
```
flip_img1 =  cv2.flip(pup1,0)
plt.imshow(flip_img1);
```

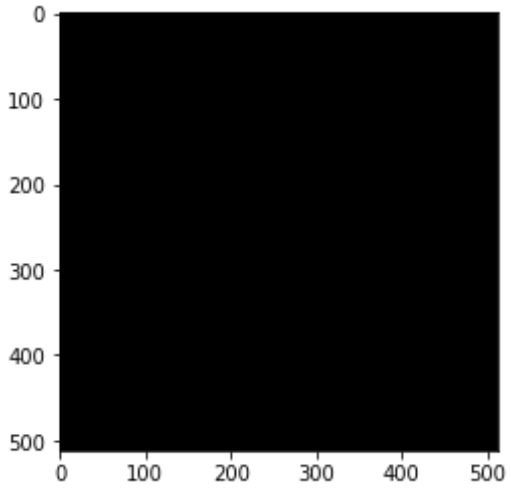
In [86]:
```
flip_img2 =  cv2.flip(pup1,-1)
plt.imshow(flip_img2);
```

In [87]:
```
pup1 = cv2.imread('pup.jpg')
pup1 = cv2.cvtColor(pup1,cv2.COLOR_BGR2RGB)
print("Shape before resize -- > ",pup1.shape)
#resize to ratios of original size,80% width and 50% height of original
pup1_resized = cv2.resize(pup1,(0,0),pup1,0.8,0.5)
print("Shape after resize -- >",pup1_resized.shape)
plt.imshow      (pup1_resized);
```

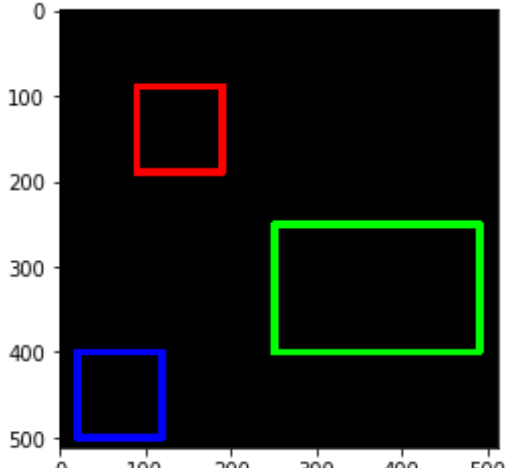
```
Shape before resize --> (1300, 1950, 3)
Shape after resize --> (650, 1560, 3)
```

Draw Rectangle on a image

In [88]:
```
#create a empty image
empty_img = np.zeros(shape=(512,512,3),dtype=np.int16)
plt.imshow(empty_img);
```



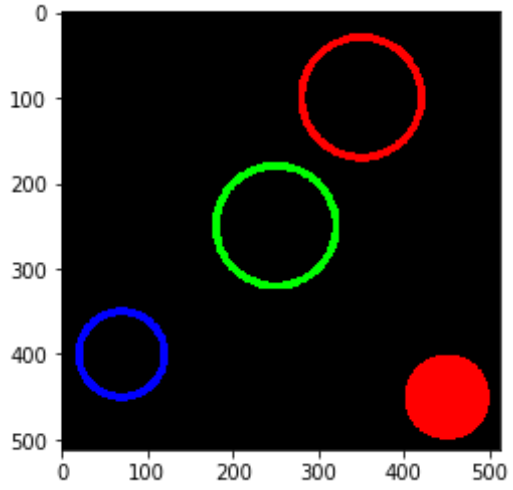
In [89]:
```
#draw green rectangle
cv2.rectangle(empty_img,(250,250),(490,400), color=(0,255,0),thickness=
8)
#draw green rectangle
cv2.rectangle(empty_img,(90,90),(190,190), color=(255,0,0),thickness=8)
#draw blue rectangle
cv2.rectangle(empty_img,(20,400),(120,500), color=(0,0,255),thickness=8
)
plt.imshow(empty_img);
```



0 100 200 300 400 500

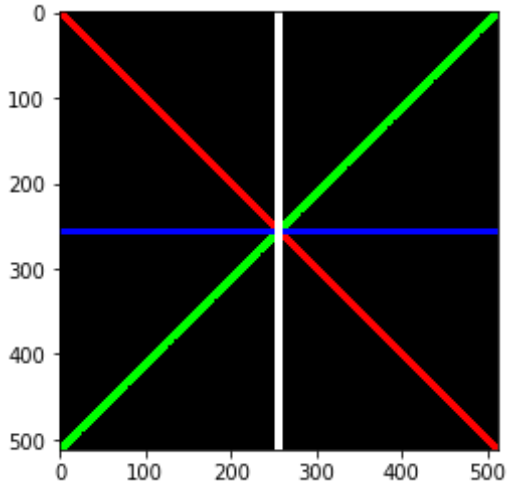
Draw Circles on a image

In [90]:
```
#draw green circle
empty_img = np.zeros(shape=(512,512,3),dtype=np.int16)
cv2.circle(empty_img,center= (250,250),radius=70, color=(0,255,0),thick
ness=8)
plt.imshow(empty_img);
#draw blue circle
cv2.circle(empty_img,center= (70,400),radius=50, color=(0,0,255),thickn
ess=8)
plt.imshow(empty_img);
#draw red circle
cv2.circle(empty_img,center= (350,100),radius=70, color=(255,0,0),thick
ness=8)
plt.imshow(empty_img);
#draw red filled circle
cv2.circle(empty_img,center= (450,450),radius=50, color=(255,0,0),thick
ness=-1)
plt.imshow(empty_img);
```



Draw Lines on a Image

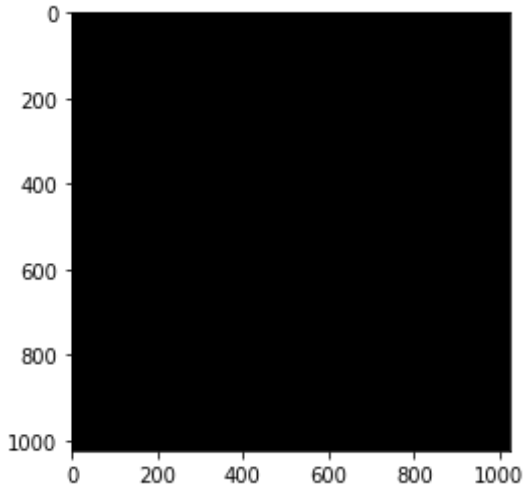
In [91]:
```
empty_img = np.zeros(shape=(512,512,3),dtype=np.int16)
#Draw red line
cv2.line(empty_img, pt1=(0,0),pt2=(512,512),color=(255,0,0),thickness=8
)
plt.imshow(empty_img);
#Draw Green Line
cv2.line(empty_img, pt1=(512,0),pt2=(0,512),color=(0,255,0),thickness=8
)
plt.imshow(empty_img);

#Draw Blue Line
cv2.line(empty_img, pt1=(0,256),pt2=(512,256),color=(0,0,255),thickness
=5)
plt.imshow(empty_img);

#Draw white Line
cv2.line(empty_img, pt1=(256,0),pt2=(256,512),color=(255,255,255),thick
ness=8)
plt.imshow(empty_img);
```



Writing Text on the Image

In [100]:
```
# create one empty image of size 1024, 1024,3
blank_img = np.zeros(shape=(1024,1024,3))
print("blank image size -- >",blank_img.shape)
plt.imshow(blank_img);
```

```
blank image size -- > (1024, 1024, 3)
```

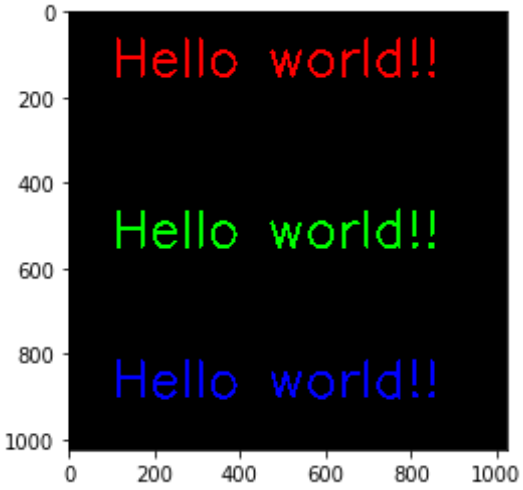


In [101]:
```
# blue color font
cv2.putText(blank_img,"Hello world!!",org=(100,900), fontFace=cv2.FONT_
HERSHEY_SIMPLEX,color=(0,0,255), fontScale=4, lineType=cv2.LINE_AA, thi
ckness=8)

#red color font
cv2.putText(blank_img,"Hello world!!",org=(100,150), fontFace=cv2.FONT_
HERSHEY_SIMPLEX,color=(255,0,0), fontScale=4, lineType=cv2.LINE_AA, thi
ckness=8)

#green color font
cv2.putText(blank_img,"Hello world!!",org=(100,550), fontFace=cv2.FONT_
HERSHEY_SIMPLEX,color=(0,255,0), fontScale=4, lineType=cv2.LINE_AA, thi
```

```
ckness=8)
plt.imshow(blank_img)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1]
for floats or [0..255] for integers).
```

Out[101]: <matplotlib.image.AxesImage at 0x1c6008b89b0>



In [ ]:

In [ ]: