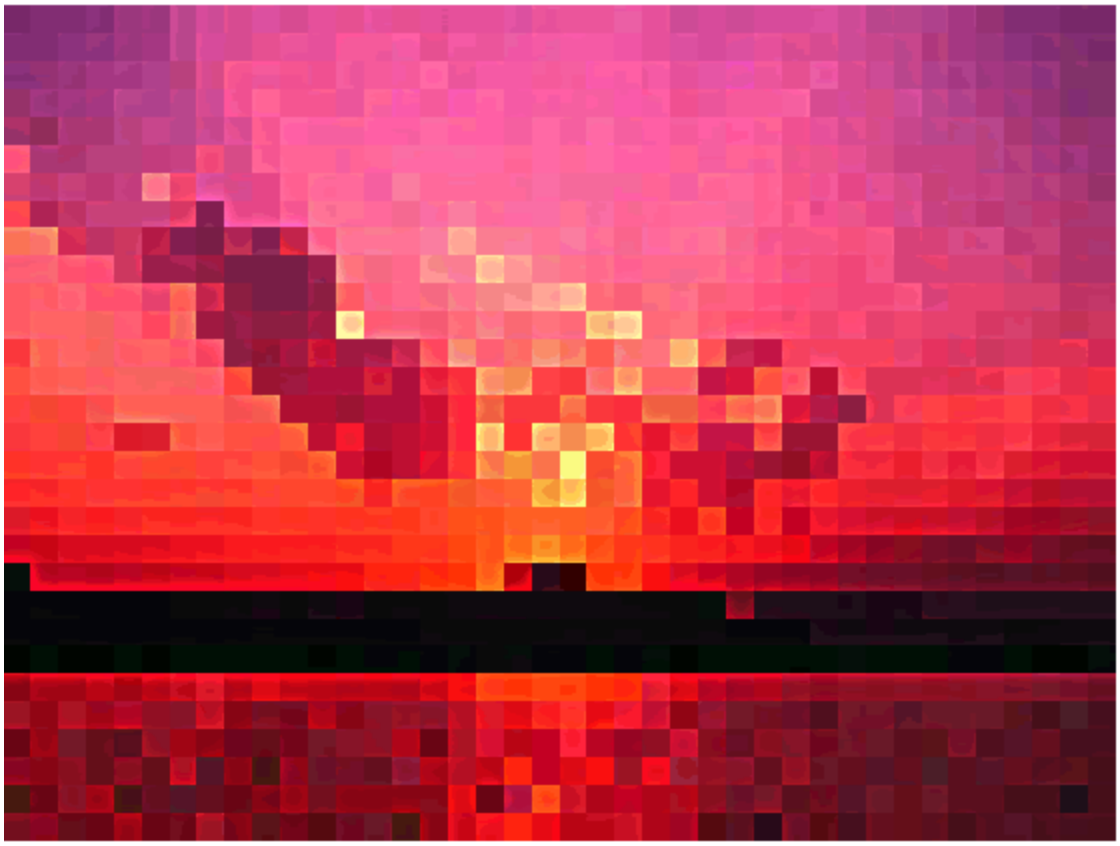
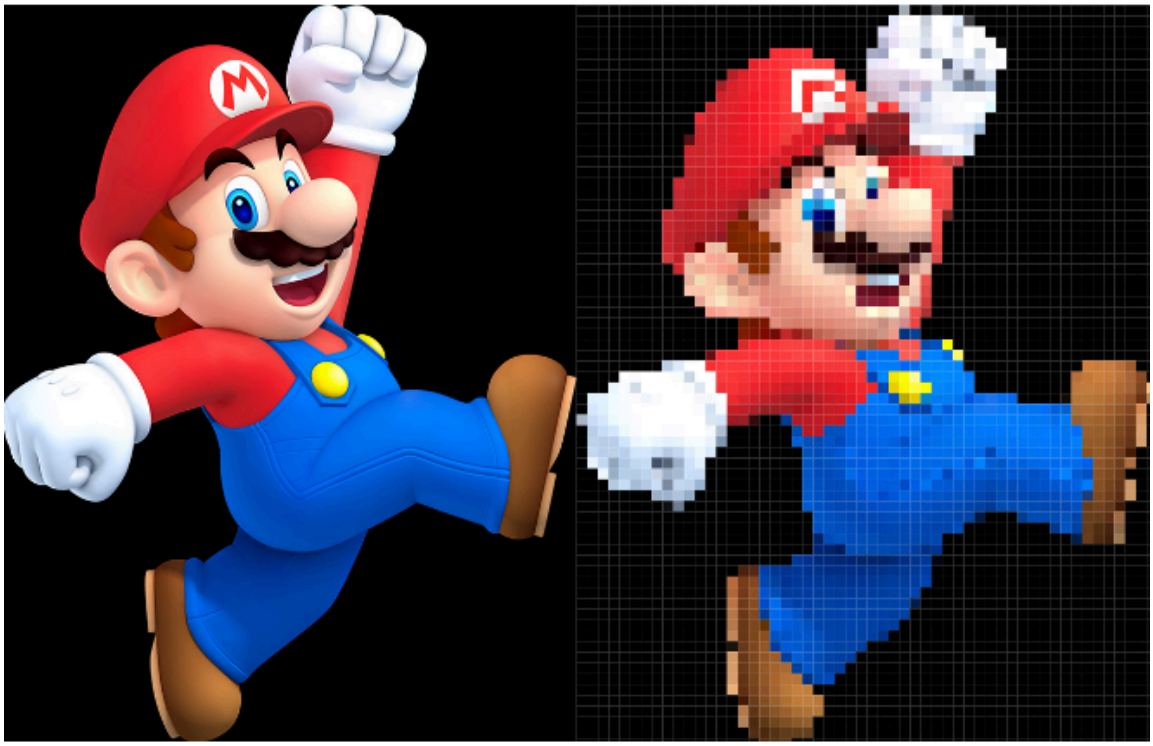
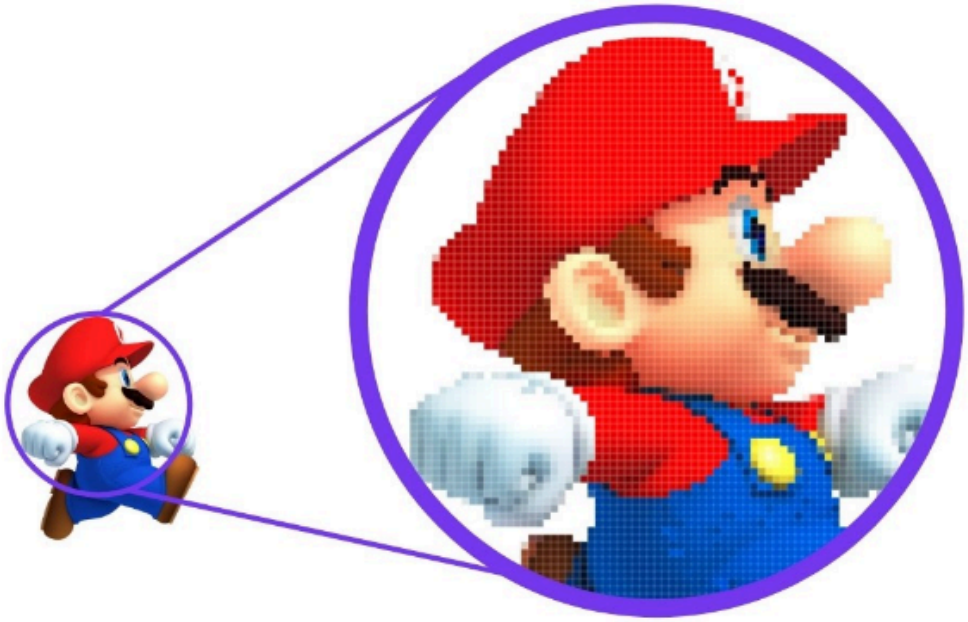
COMPUTER VISION:-

- It is a field where we provide machines the ability to visualize, remember and recognize.
- Computer Vision helps the machine to extract meaningful information from any digital input i.e: images,videos etc.







In [1]:
```
# to work with computer vision we need to
# install the library:-
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\lab25\anaconda3\lib\site-pa
ckages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\lab25\anaconda3\lib\site-pa
ckages (from opencv-python) (1.26.3)
```

In [2]:
```
import cv2
import matplotlib.pyplot as plt
```

In [3]:
```
# Reading an Image:-
img = cv2.imread(r"C:\Users\lab25\Desktop\AIML_2nd_June\shin-chan-india-movie_sgq4.
img
```

Out[3]:
```
array([[[ 84, 100, 137],
        [ 93, 108, 147],
        [ 96, 105, 148],
        ...,
        [ 42,  48,  67],
        [ 39,  45,  64],
        [ 36,  42,  61]],

       [[ 86,  99, 137],
        [ 88, 100, 140],
        [ 98, 107, 150],
        ...,
        [ 42,  48,  67],
        [ 40,  46,  65],
        [ 38,  44,  63]],

       [[ 93, 104, 142],
        [ 96, 107, 145],
        [ 99, 107, 147],
        ...,
        [ 40,  46,  65],
        [ 42,  46,  65],
        [ 42,  46,  65]],

       ...,

       [[155, 155, 173],
        [159, 159, 177],
        [157, 157, 175],
        ...,
        [110, 116, 129],
        [111, 117, 130],
        [113, 119, 132]],

       [[160, 160, 176],
        [166, 166, 182],
        [162, 162, 178],
        ...,
        [108, 114, 127],
        [109, 115, 128],
        [111, 117, 130]],

       [[128, 128, 144],
        [151, 151, 167],
        [166, 166, 182],
        ...,
        [106, 112, 125],
        [106, 112, 125],
        [108, 114, 127]]], dtype=uint8)
```

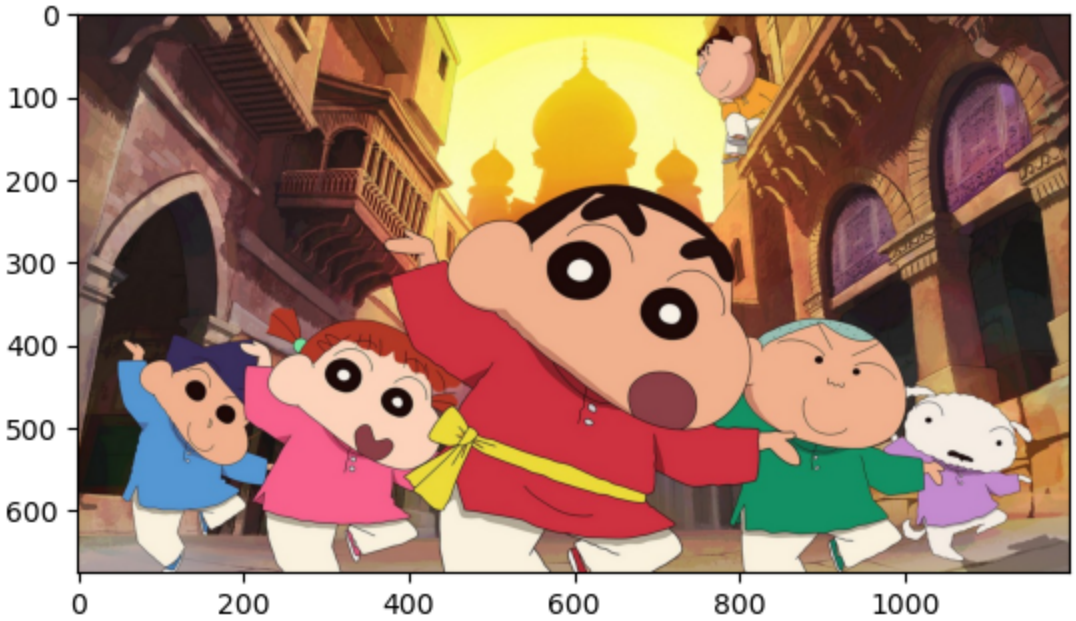
In [5]:
```
img.shape
```

Out[5]:
```
(675, 1200, 3)
```

In [6]:
```
# to show the image using cv2:-
cv2.imshow('Shinchan',img)
cv2.waitKey(0) # to open the image window for infinite time
cv2.destroyAllWindows() # to end the execution by destroying all image windows.
```

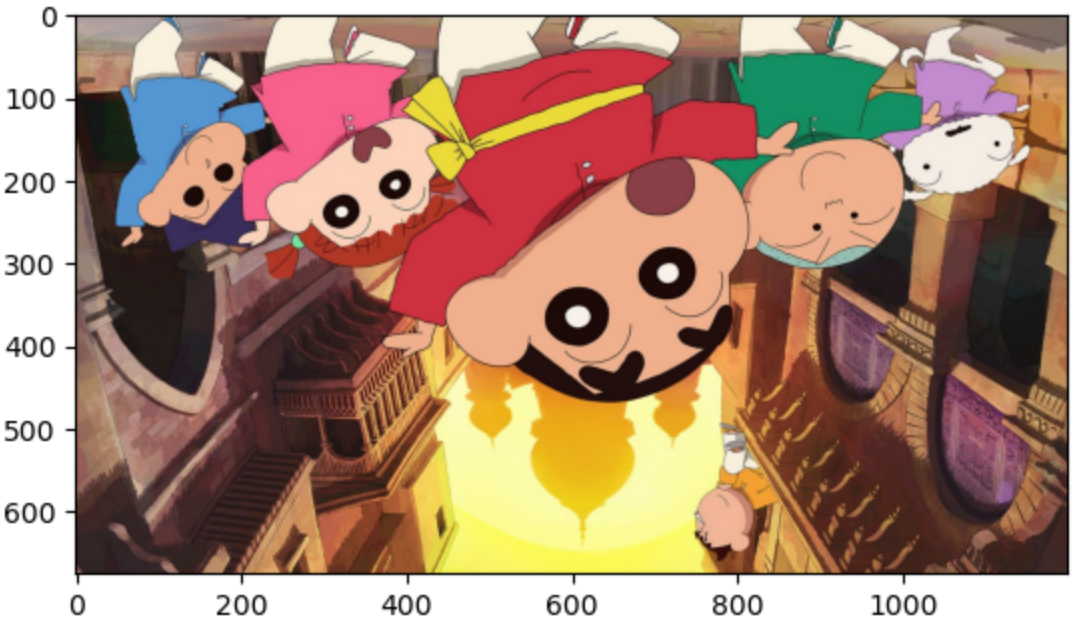
In [7]:
```
# to plot the image:
plt.imshow(img)
plt.show()
```

In [9]:
```
# to plot the image in RGB.
# 3D array slicing:-
# [rowslice, columnslice, channelslice]
plt.imshow(img[::,::,::-1])
plt.show()
```

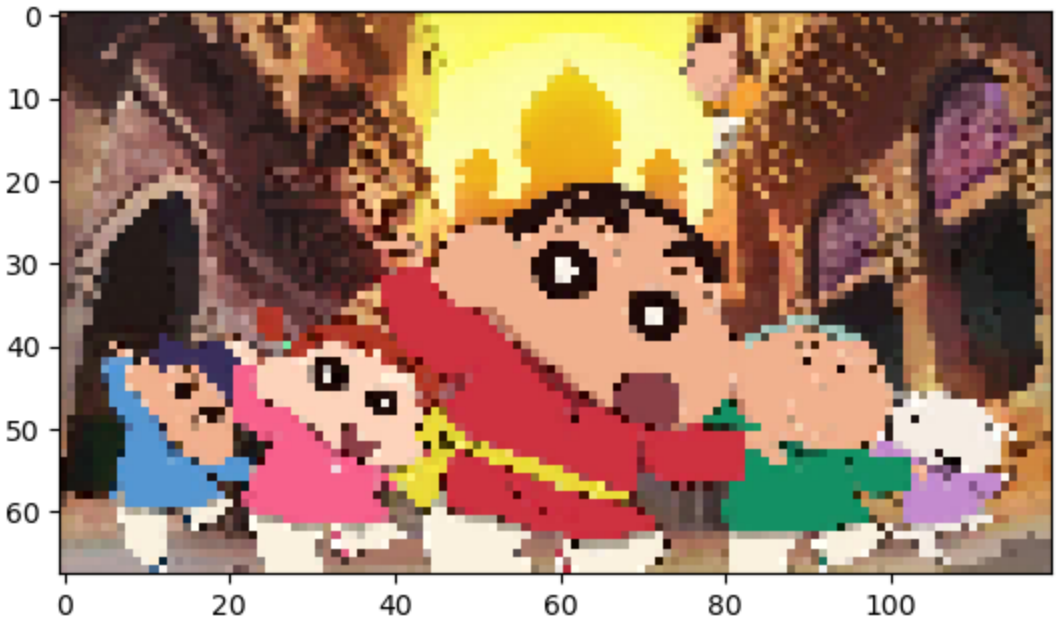


In [11]:
```
# reversing the column returns horizontally flipped image.
plt.imshow(img[::,::-1,::-1])
plt.show()
```

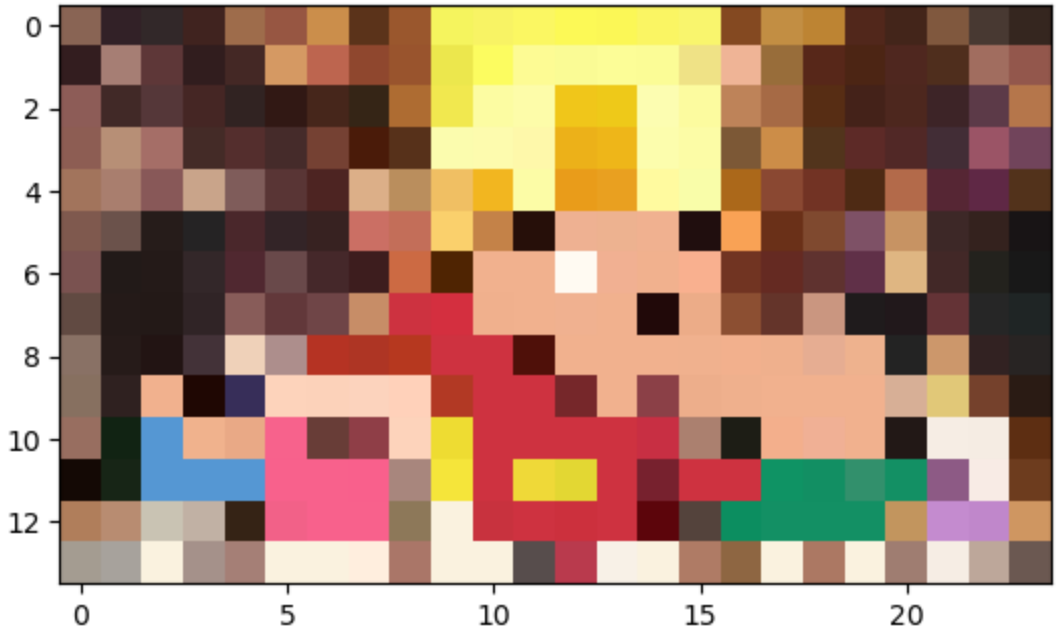
In [13]:
```
# reversing the column returns vertically flipped image.
plt.imshow(img[::-1,::,::-1])
plt.show()
```



In [14]:
```
plt.imshow(img[::10,::10,::-1])
plt.show()
```



In [15]:
```
plt.imshow(img[::50,::50,::-1])
plt.show()
```



In [16]:
```
img
```

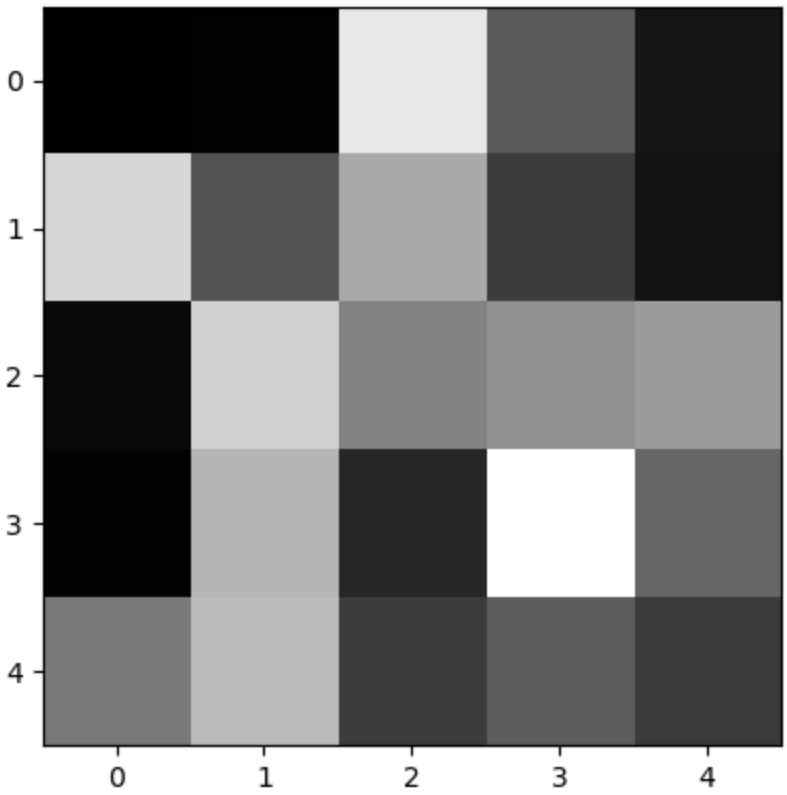
Out[16]:
```
array([[[ 84, 100, 137],
        [ 93, 108, 147],
        [ 96, 105, 148],
        ...,
        [ 42,  48,  67],
        [ 39,  45,  64],
        [ 36,  42,  61]],

       [[ 86,  99, 137],
        [ 88, 100, 140],
        [ 98, 107, 150],
        ...,
        [ 42,  48,  67],
        [ 40,  46,  65],
        [ 38,  44,  63]],

       [[ 93, 104, 142],
        [ 96, 107, 145],
        [ 99, 107, 147],
        ...,
        [ 40,  46,  65],
        [ 42,  46,  65],
        [ 42,  46,  65]],

       ...,

       [[155, 155, 173],
        [159, 159, 177],
        [157, 157, 175],
        ...,
        [110, 116, 129],
        [111, 117, 130],
        [113, 119, 132]],

       [[160, 160, 176],
        [166, 166, 182],
        [162, 162, 178],
        ...,
        [108, 114, 127],
        [109, 115, 128],
        [111, 117, 130]],

       [[128, 128, 144],
        [151, 151, 167],
        [166, 166, 182],
        ...,
        [106, 112, 125],
        [106, 112, 125],
        [108, 114, 127]]], dtype=uint8)
```

In [17]:
```
import numpy as np
```

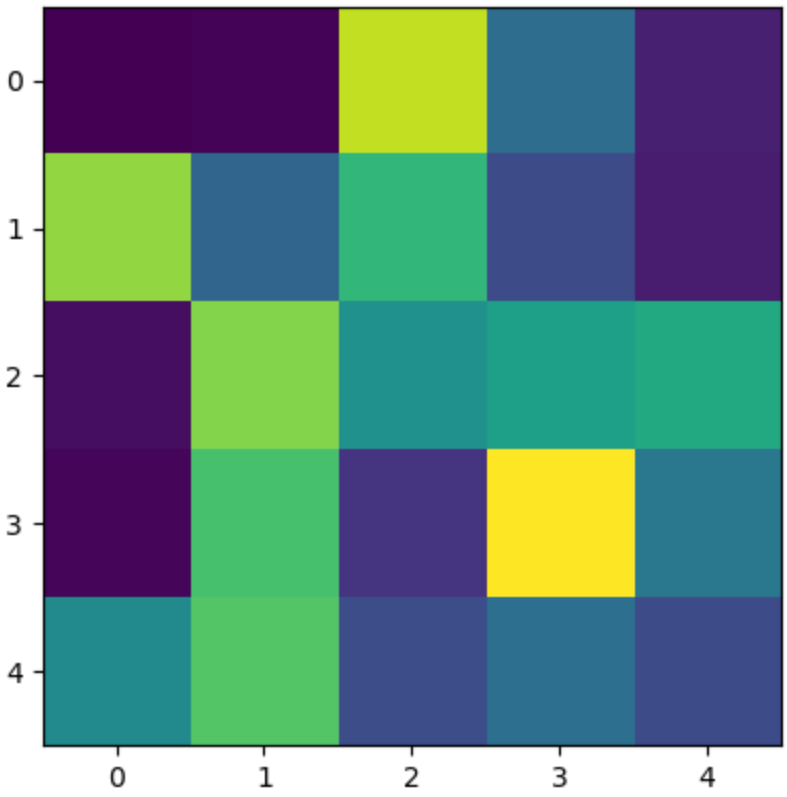
In [18]:
```
a = np.random.randint(0,255,(5,5))
a
```

Out[18]:
```
array([[  4,   6, 222,  90,  25],
       [205,  82, 163,  60,  22],
       [ 13, 199, 125, 139, 150],
       [  7, 174,  41, 244, 100],
       [119, 180,  61,  92,  60]])
```

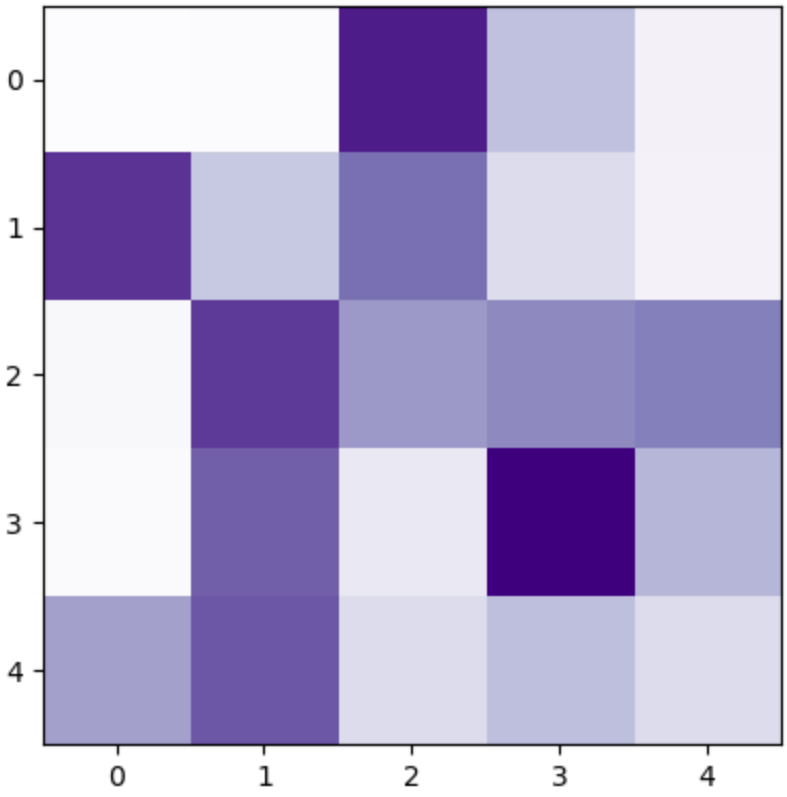
In [19]:
```
plt.imshow(a,cmap='gray')
plt.show()
```



In [22]:
```
plt.imshow(a)
plt.show()
```

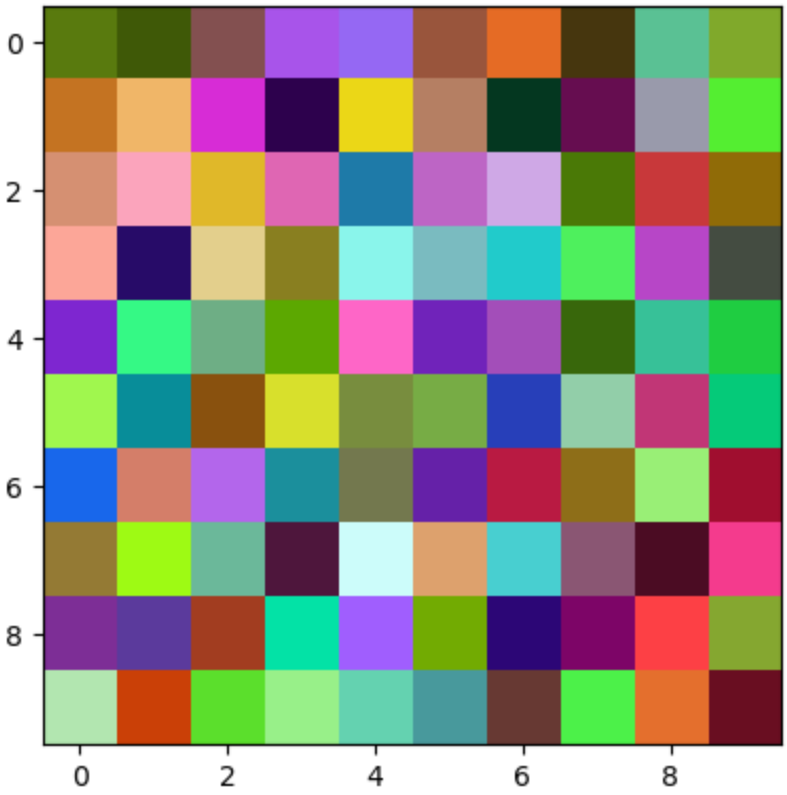


In [56]:
```
plt.imshow(a,cmap='Purples')
plt.show()
```

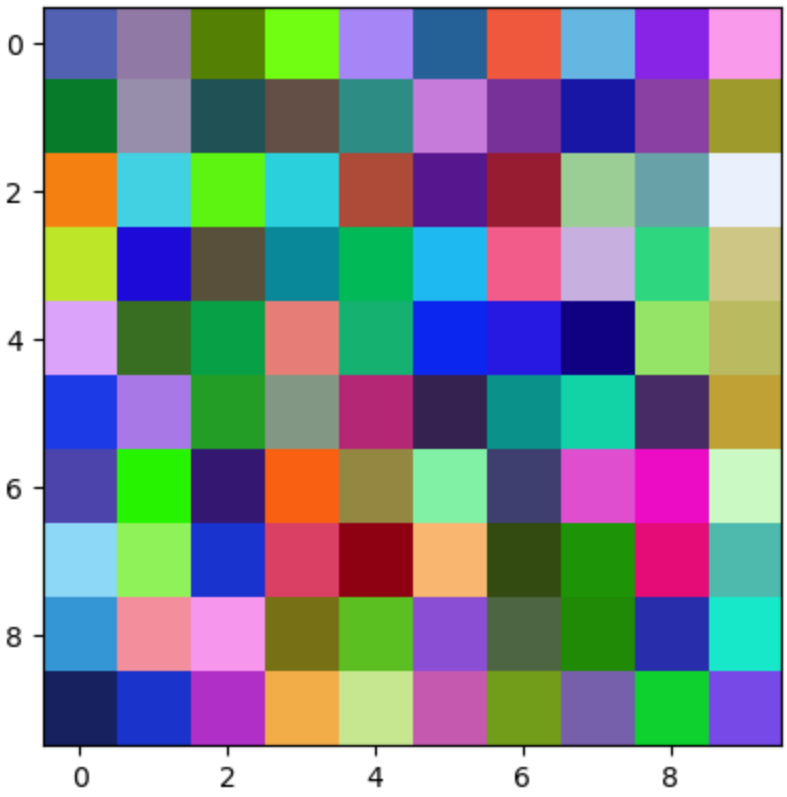


In [24]:
```
# colorful palette:
col = np.random.randint(0,255,(10,10,3))
```

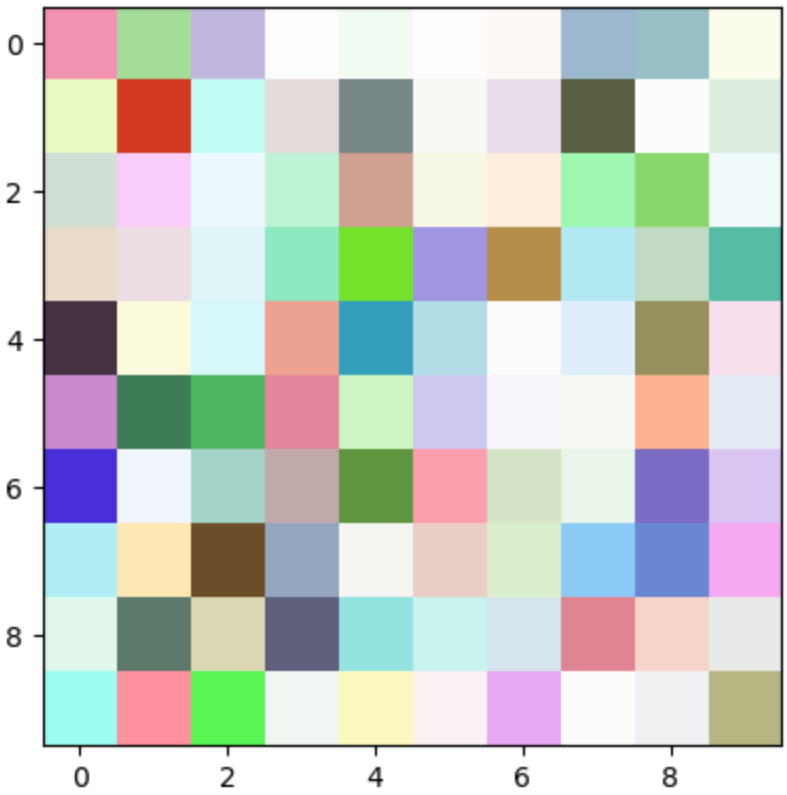
```
plt.imshow(col)
plt.show()
```



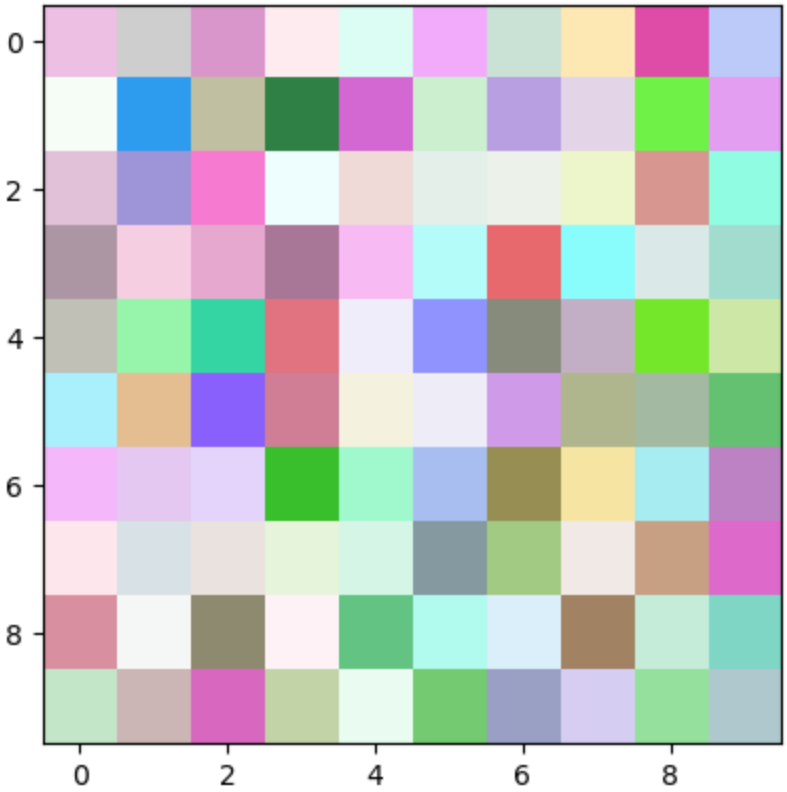
In [26]:
```
# Plotting the color palette in RGB.
col = np.random.randint(0,255,(10,10,3))
plt.imshow(col[::,::,::-1])
plt.show()
```



In [27]:
```
# Plotting a 4 channel (RGBA) color palette:-
# Alpha value -> varies from 0 ~ 1.
b = np.random.randint(0,255,(10,10,4))
plt.imshow(b)
plt.show()
```

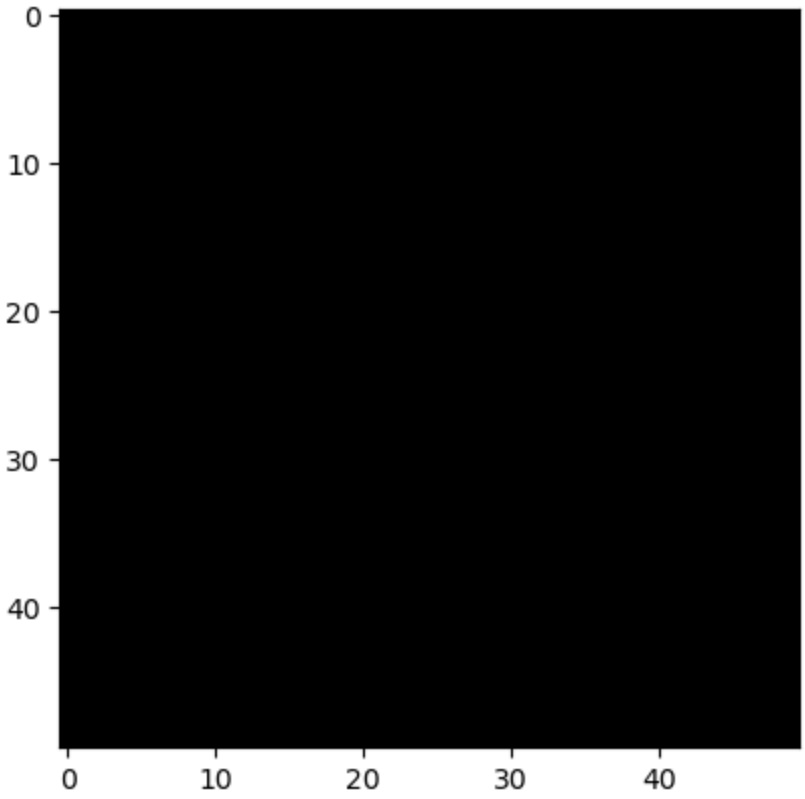


In [28]:
```
# Plotting a 4 channel (RGBA) color palette:-
# Alpha value -> varies from 0 ~ 1.
b = np.random.randint(0,255,(10,10,4))
plt.imshow(b[::,::,::-1])
plt.show()
```

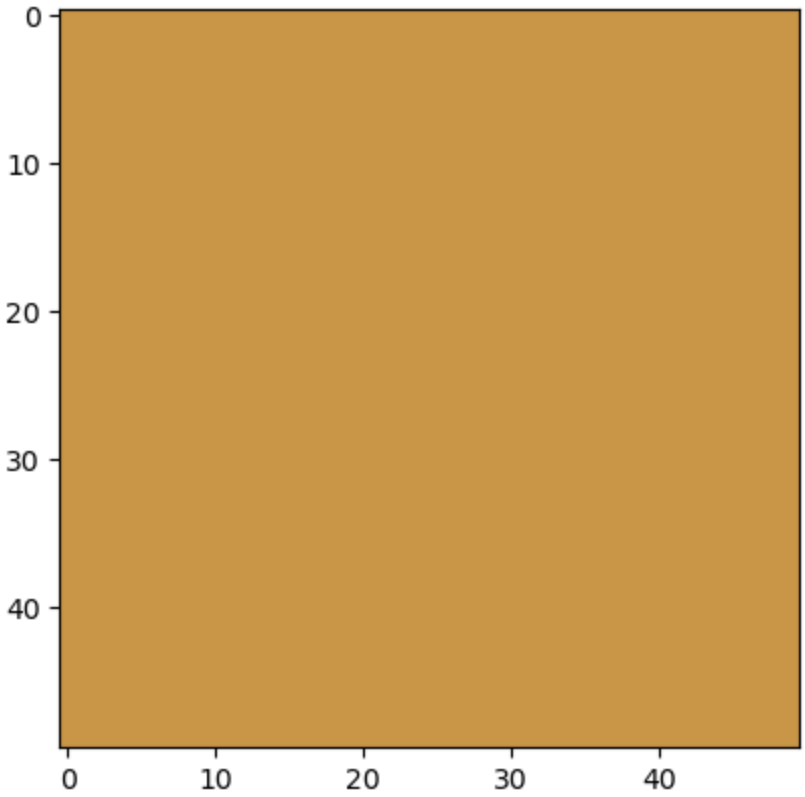


Color Combinations:-

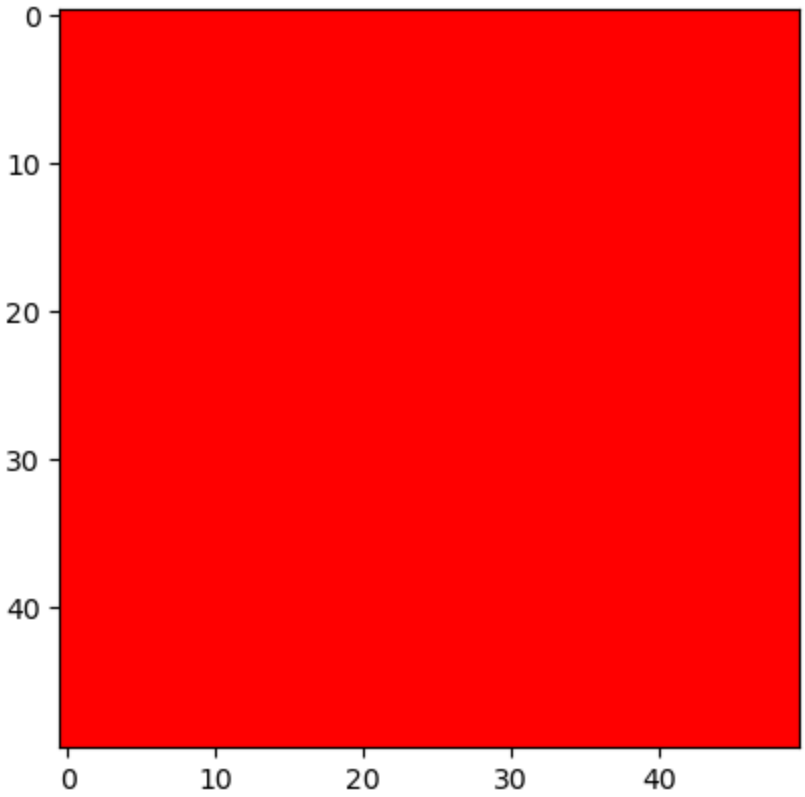
In [38]:
```
r = np.full((50,50),0)
g = np.full((50,50),0)
b = np.full((50,50),0)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.show()
```



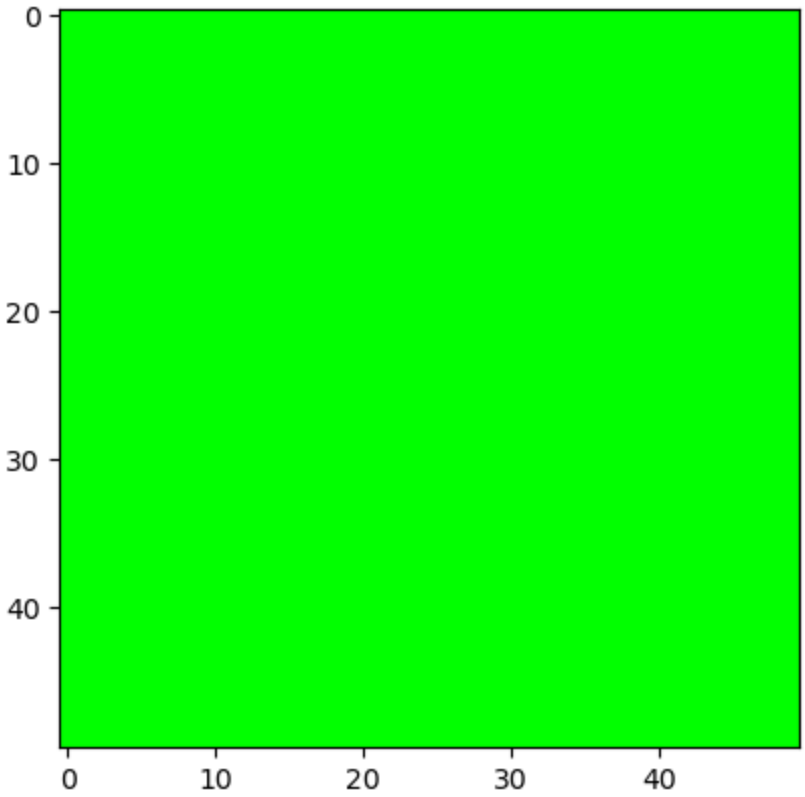
In [39]:
```
r = np.full((50,50),200)
g = np.full((50,50),150)
b = np.full((50,50),70)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.show()
```



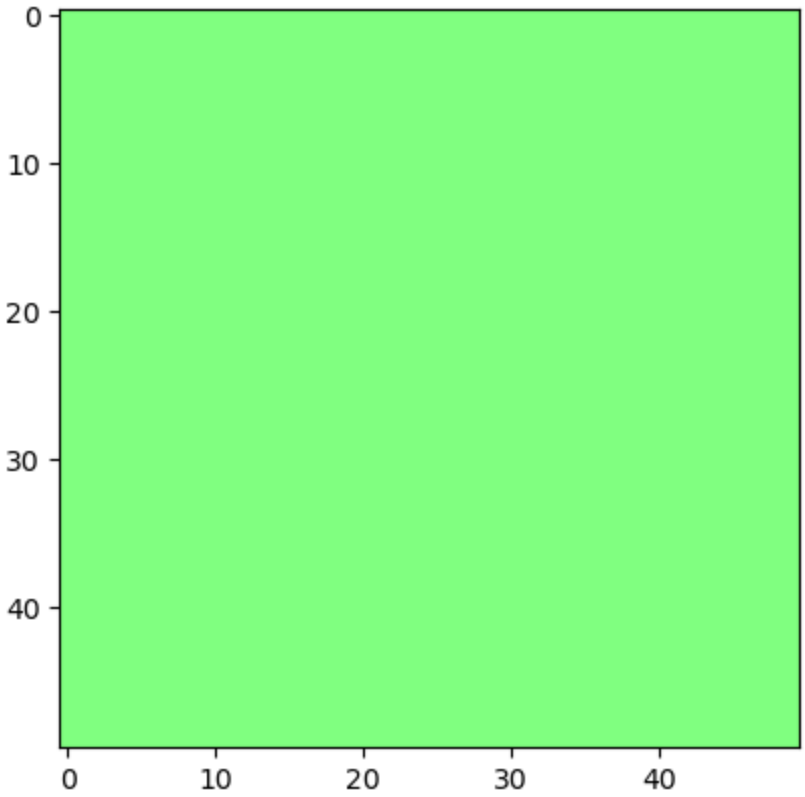
In [40]:
```
r = np.full((50,50),255)
g = np.full((50,50),0)
b = np.full((50,50),0)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.show()
```



In [43]:
```
r = np.full((50,50),0)
g = np.full((50,50),255)
b = np.full((50,50),0)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.show()
```

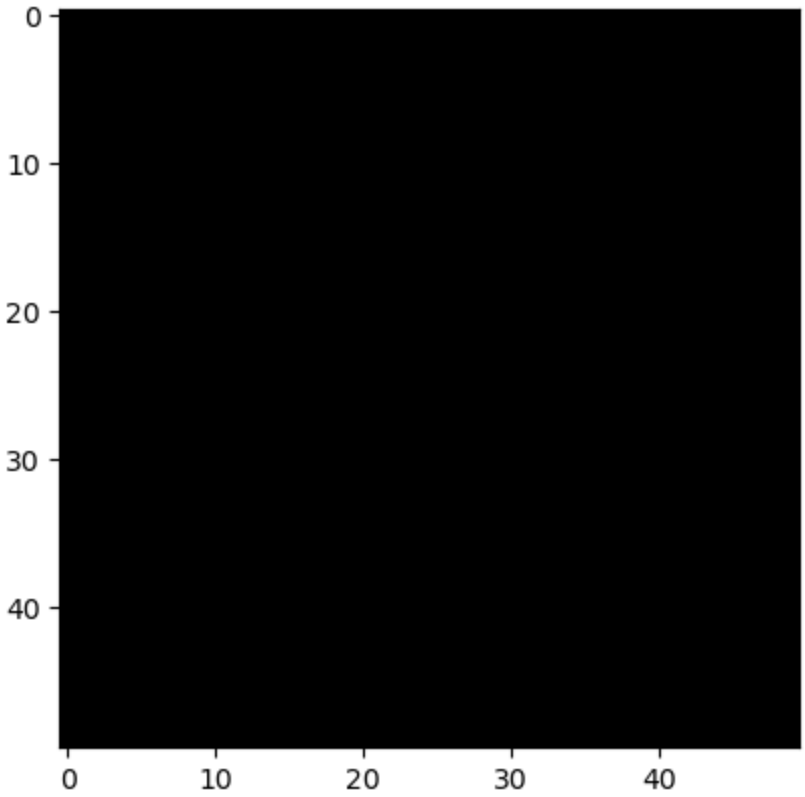


In [42]:
```
r = np.full((50,50),0)
g = np.full((50,50),255)
b = np.full((50,50),0)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1,alpha=0.5)
plt.show()
```

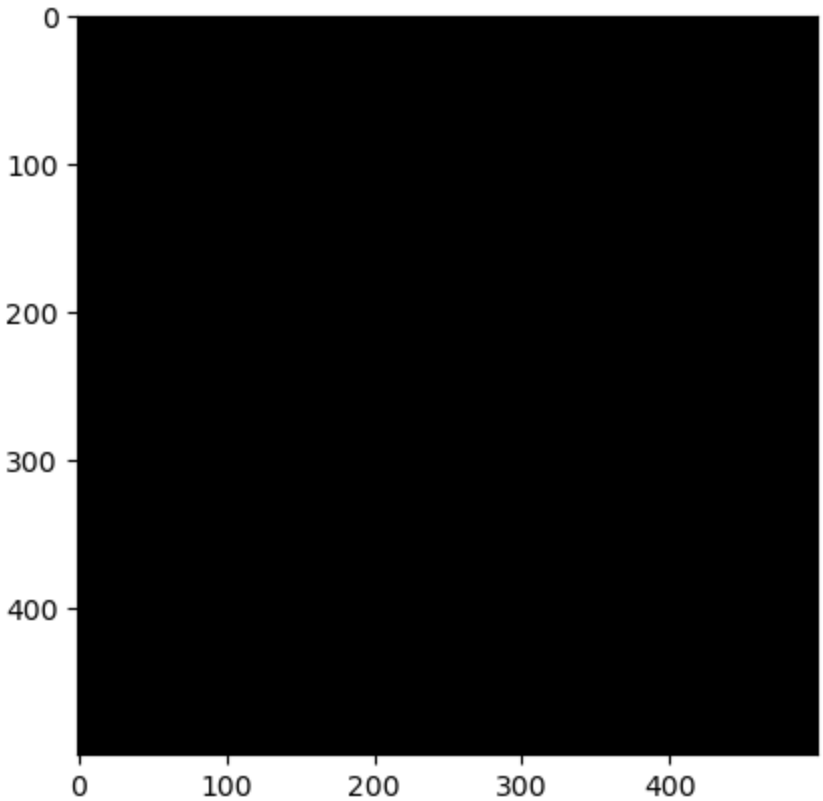


Creating Shapes :-

In [44]:
```
# Drawing a line:-
# Syntax:- cv2.line(startcoordinate(x,y),
# endcoordinate(x1,y1),color(rgb),thickness)
# creating an array of black color:-
r = np.full((50,50),0)
g = np.full((50,50),0)
b = np.full((50,50),0)
rgb1 = np.dstack((r,g,b))
plt.imshow(rgb1)
plt.show()
```

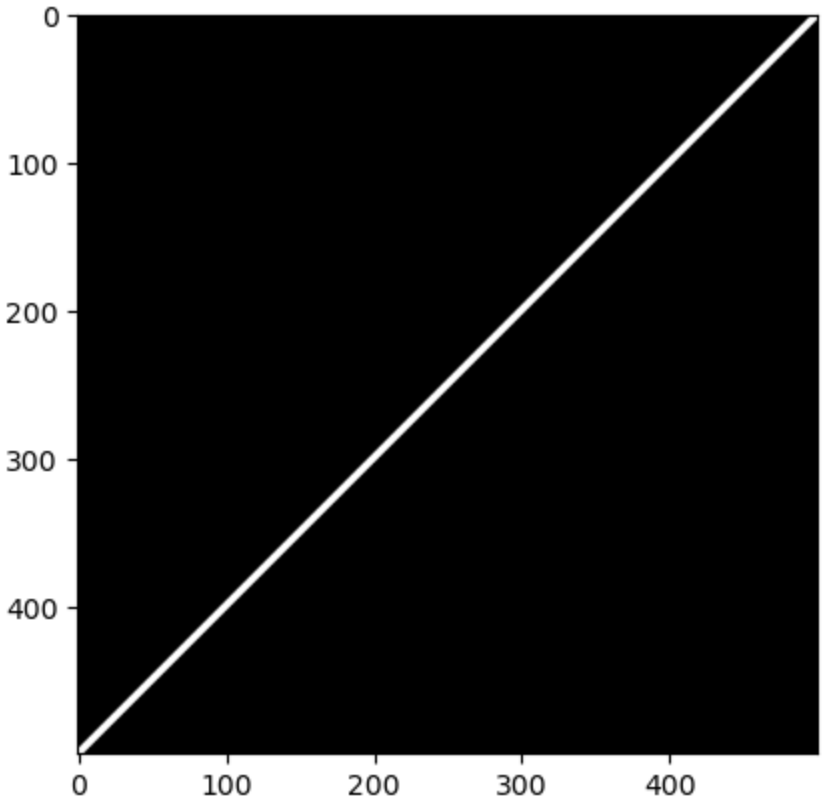


In [45]:
```
p = np.zeros((500,500,3))
plt.imshow(p)
plt.show()
```



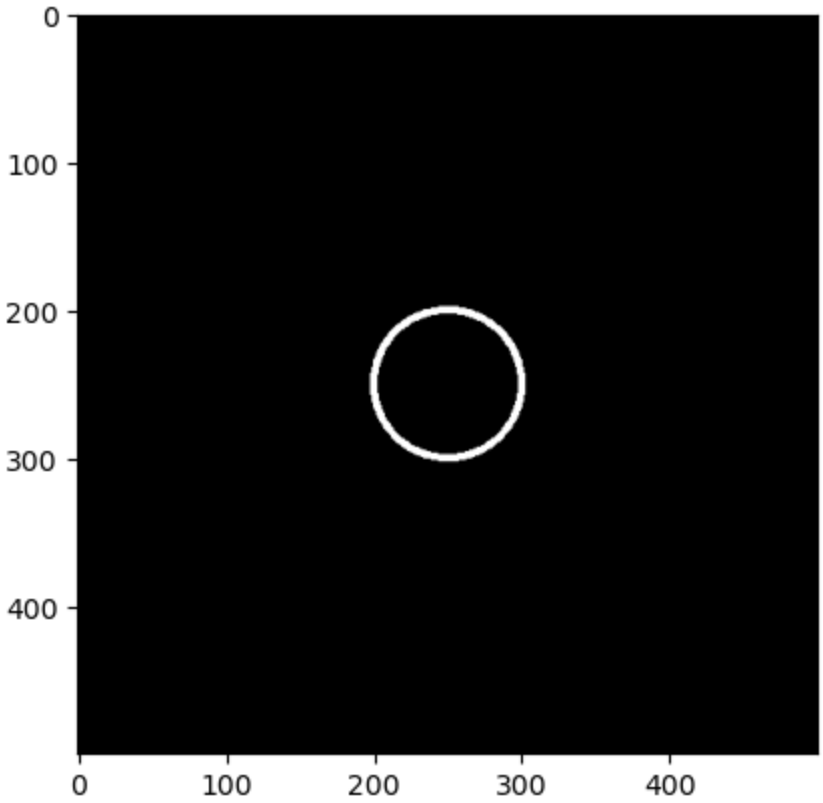
In [46]:
```
cv2.line(p,(0,500),(500,0),(255,255,255),4)
plt.imshow(p)
plt.show()
```

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



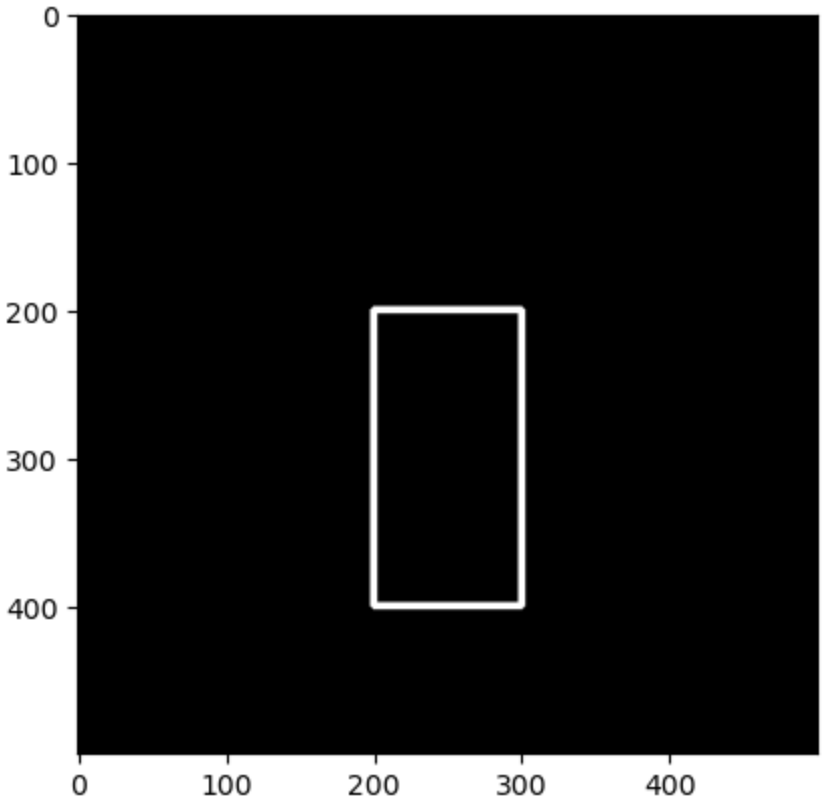
In [48]:
```
# cv2.circle(center(x,y),radius,color,thickness)
p1 = np.zeros((500,500,3))
cv2.circle(p1,(250,250),50,(255,255,255),4)
plt.imshow(p1)
plt.show()
```

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



In [53]:
```
# cv2.rectangle(1stdiagonalcord(x,y),2nddiagonalcord(x1,y1),
# color, thickness)
p2 = np.zeros((500,500,3))
cv2.rectangle(p2,(200,400),(300,200),
              (255,255,255),4)
plt.imshow(p2)
plt.show()
```

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



In [ ]: