In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imshow, imread
from skimage.color import rgb2hsv, hsv2rgb
import cv2
```

In [2]:

```
image = imread(r'C:\Users\hp\Downloads/tree.jpg')
plt.figure(num=None, figsize=(8, 6), dpi=80)
imshow(image);
```

In [6]:

```
from PIL import Image
im = Image.open(r'C:\Users\hp\Downloads/tree.jpg', 'r')
width, height = im.size
pixel_values = list(im.getdata())
```

In [8]:

```
import cv2
```

In [9]:

```
flags = [i for i in dir(cv2) if i.startswith('COLOR_')]
```

In [10]:

```
len(flags)
```

Out[10]:

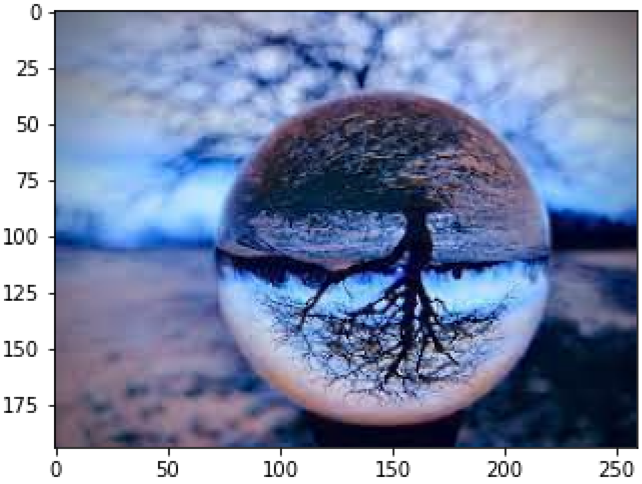
```
274
```

```
In [11]: flags[40]
```

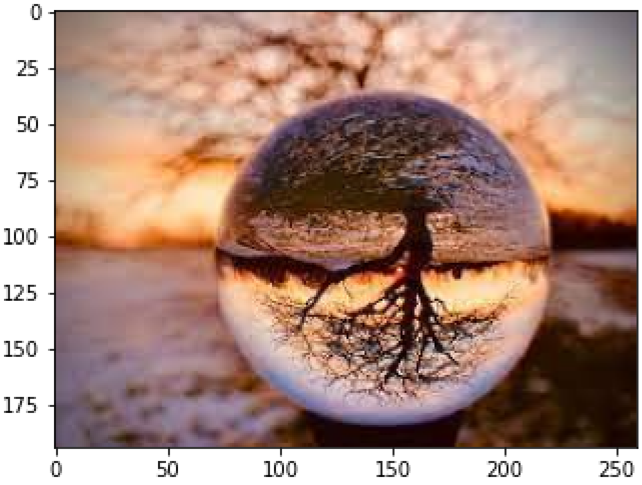
```
Out[11]: 'COLOR_BGR2HLS'
```

```
In [12]: import matplotlib.pyplot as plt
         import numpy as np
```

```
In [14]: image = cv2.imread(r'C:\Users\hp\Downloads/tree.jpg')
         plt.imshow(image)
         plt.show()
```



```
In [15]: image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
         plt.imshow(image)
         plt.show()
```



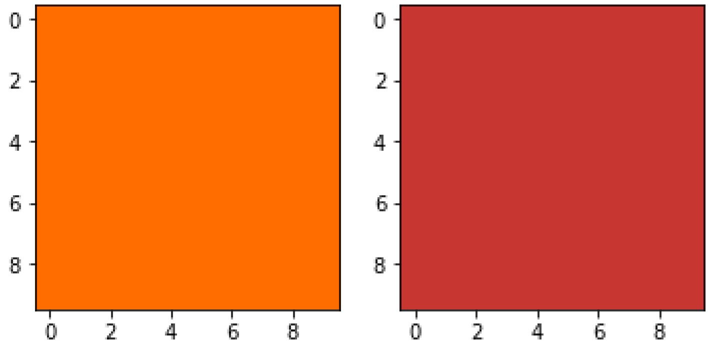
```
In [16]: hsv_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
```

```
In [22]: from matplotlib.colors import hsv_to_rgb
```

```
In [23]: lo_square = np.full((10, 10, 3), light_orange, dtype=np.uint8) / 255.0
         do_square = np.full((10, 10, 3), dark_orange, dtype=np.uint8) / 255.0
```

In [24]:

```
plt.subplot(1, 2, 1)
plt.imshow(hsv_to_rgb(do_square))
plt.subplot(1, 2, 2)
plt.imshow(hsv_to_rgb(lo_square))
plt.show()
```



In [25]:

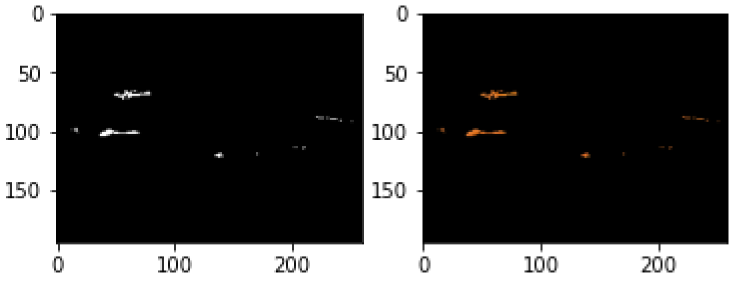
```
mask = cv2.inRange(hsv_image, light_orange, dark_orange)
```

In [26]:

```
result = cv2.bitwise_and(image, image, mask=mask)
```

In [27]:

```
plt.subplot(1, 2, 1)
plt.imshow(mask, cmap="gray")
plt.subplot(1, 2, 2)
plt.imshow(result)
plt.show()
```

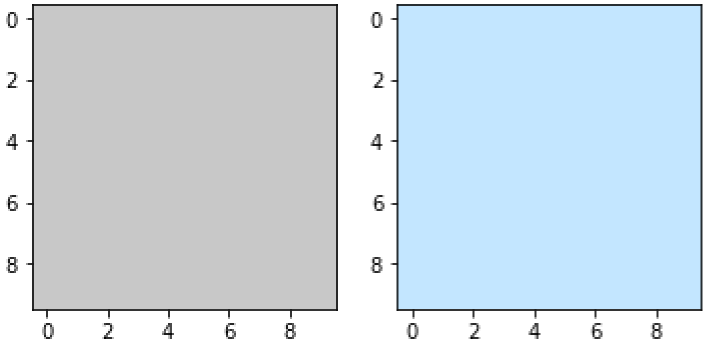


In [28]:

```
light_white = (0, 0, 200)
dark_white = (145, 60, 255)
```

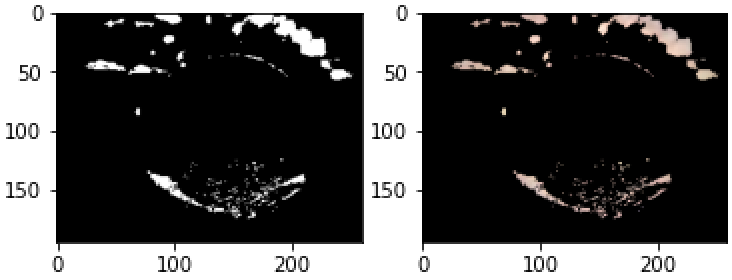
In [29]:

```
lw_square = np.full((10, 10, 3), light_white, dtype=np.uint8) / 255.0
dw_square = np.full((10, 10, 3), dark_white, dtype=np.uint8) / 255.0

plt.subplot(1, 2, 1)
plt.imshow(hsv_to_rgb(lw_square))
plt.subplot(1, 2, 2)
plt.imshow(hsv_to_rgb(dw_square))
plt.show()
```

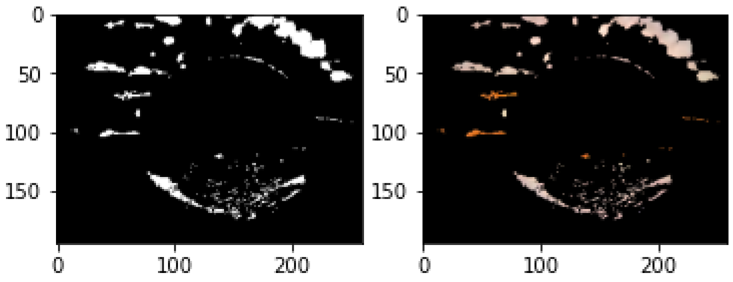


In [31]:

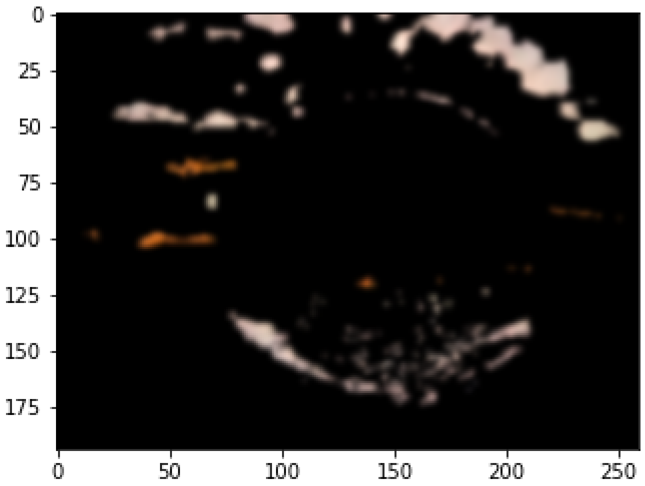
```
mask_white = cv2.inRange(hsv_image, light_white, dark_white)
result_white = cv2.bitwise_and(image, image, mask=mask_white)

plt.subplot(1, 2, 1)
plt.imshow(mask_white, cmap="gray")
plt.subplot(1, 2, 2)
plt.imshow(result_white)
plt.show()
```



In [32]:
```python
final_mask = mask + mask_white

final_result = cv2.bitwise_and(image, image, mask=final_mask)
plt.subplot(1, 2, 1)
plt.imshow(final_mask, cmap="gray")
plt.subplot(1, 2, 2)
plt.imshow(final_result)
plt.show()
```



In [33]:
```python
blur = cv2.GaussianBlur(final_result, (7, 7), 0)
plt.imshow(blur)
plt.show()
```



In [13]:
```python
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import imageio
```

GAUSSIAN NOISE

In [15]:

```
import cv2
import numpy as np

img = cv2.imread(r'C:/Users/hp/Downloads\a.jpg')
# Generate Gaussian noise
gauss = np.random.normal(0,1,img.size)
gauss = gauss.reshape(img.shape[0],img.shape[1],img.shape[2]).astype('uint8')
# Add the Gaussian noise to the image
img_gauss = cv2.add(img,gauss)
# Display the image
cv2.imshow('a',img_gauss)
cv2.waitKey(0)
```

Out[15]:

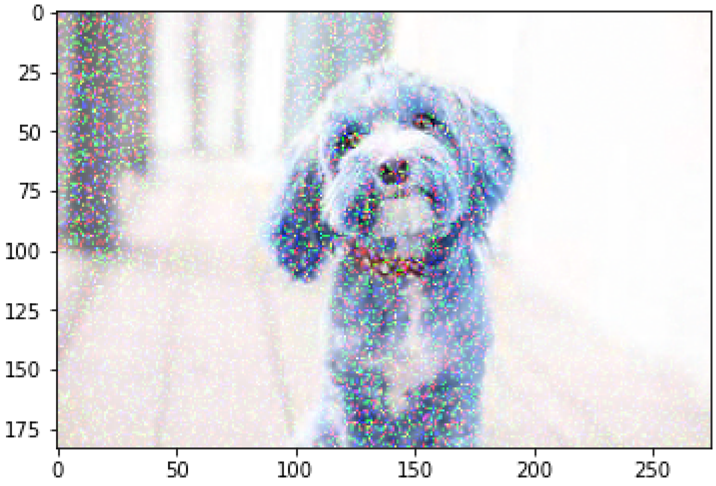
```
-1
```

In [16]:

```
plt.imshow(img_gauss)
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x27f9a1117c0>
```



SPECKLE NOISE

In [17]:

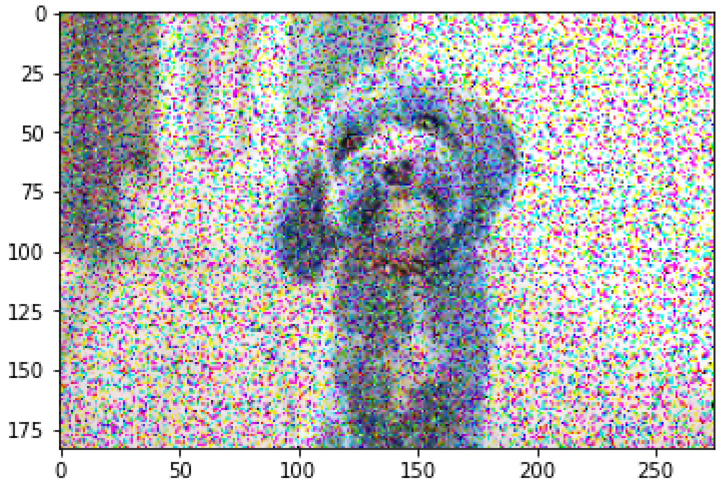
```
import cv2
import numpy as np

img = cv2.imread(r'C:/Users/hp/Downloads\a.jpg')

gauss = np.random.normal(0,1,img.size)
gauss = gauss.reshape(img.shape[0],img.shape[1],img.shape[2]).astype('uint8')
noise = img + img * gauss

cv2.imshow('a',noise)
cv2.waitKey(0)
```

Out[17]:

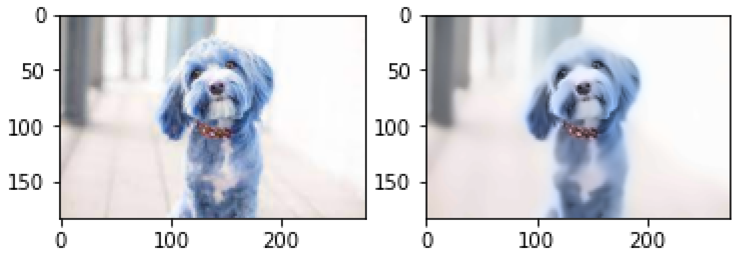
```
-1
```

```
In [19]: plt.imshow(noise)
```

```
Out[19]: <matplotlib.image.AxesImage at 0x27f9a1d6370>
```



```
In [20]: import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread(r'C:/Users/hp/Downloads\a.jpg')

dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)

plt.subplot(121),plt.imshow(img)
plt.subplot(122),plt.imshow(dst)
plt.show()
```



POISSON NOISE

In [27]:

```
import cv2
import numpy as np

img = cv2.imread(r'C:/Users/hp/Downloads\a.jpg')

noise_mask = np.random.poisson(img)

noisy_img = img + noise_mask

cv2.imshow('a',noise)
cv2.waitKey(0)
```

Out[27]:

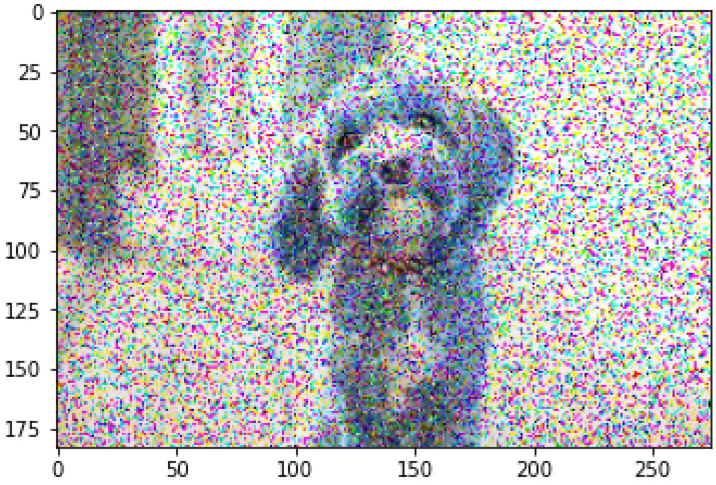
```
-1
```

In [30]:

```
plt.imshow(noise)
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x27f9a6b5280>
```



In [35]:

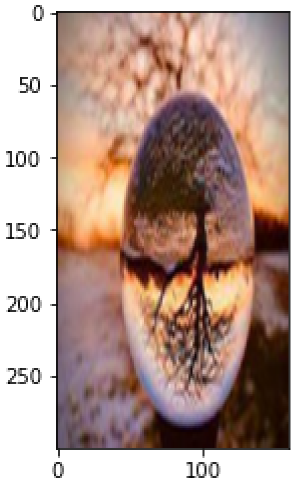
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.misc
from scipy import ndimage
import imageio
```

In [45]:

```
from PIL import Image
 # My image is a 200x374 jpeg that is 102kb large
foo = Image.open(r'C:/Users/hp/Downloads\tree.jpg')
foo.size
(200,374)
 # I downsize the image with an ANTIALIAS filter (gives the highest quality)
foo = foo.resize((160,300),Image.ANTIALIAS)
foo.save(r'C:/Users/hp/Downloads\image_scaled.jpg',quality=95)
 # The saved downsized image size is 24.8kb
foo.save(r'C:/Users/hp/Downloads\image_scaled_opt.jpg',optimize=True,quality=95)
 # The saved downsized image size is 22.9kb
```

In [46]:
```
plt.imshow(foo)
```

Out[46]:
```
<matplotlib.image.AxesImage at 0x27f9aaf56d0>
```

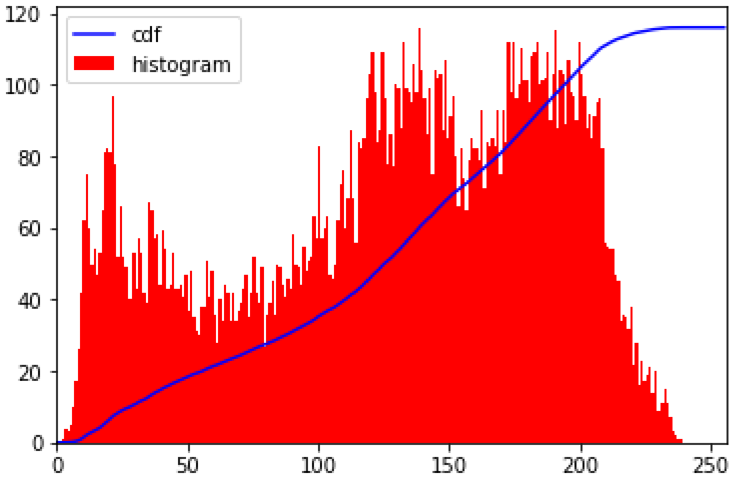


In [40]:
```
file_name = "tree.jpg"
picture = Image.open(r'C:/Users/hp/Downloads\tree.jpg')
dim = picture.size
print(f"This is the current width and height of the image: {dim}")
```

```
This is the current width and height of the image: (150, 100)
```

In [52]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread(r'C:/Users/hp/Downloads\tree.jpg',0)

hist,bins = np.histogram(img.flatten(),256,[0,256])

cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()

plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [53]:

```
img = cv2.imread(r'C:/Users/hp/Downloads\tree.jpg',0)
equ = cv2.equalizeHist(img)
res = np.hstack((img,equ)) #stacking images side-by-side
cv2.imwrite('res.png',res)
```

Out[53]:

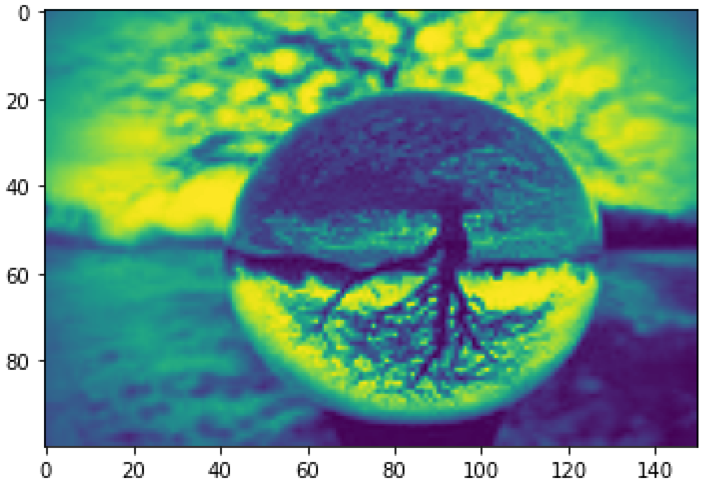
```
True
```

In [54]:

```
plt.imshow(equ)
```

Out[54]:

```
<matplotlib.image.AxesImage at 0x27f9bdd4730>
```

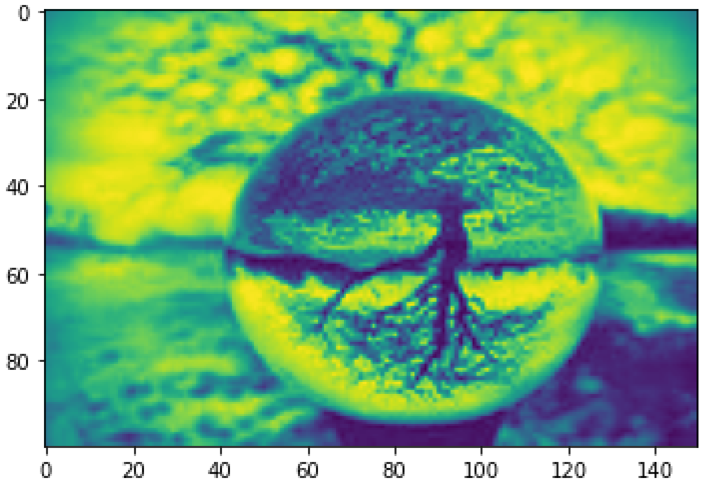


In [56]:

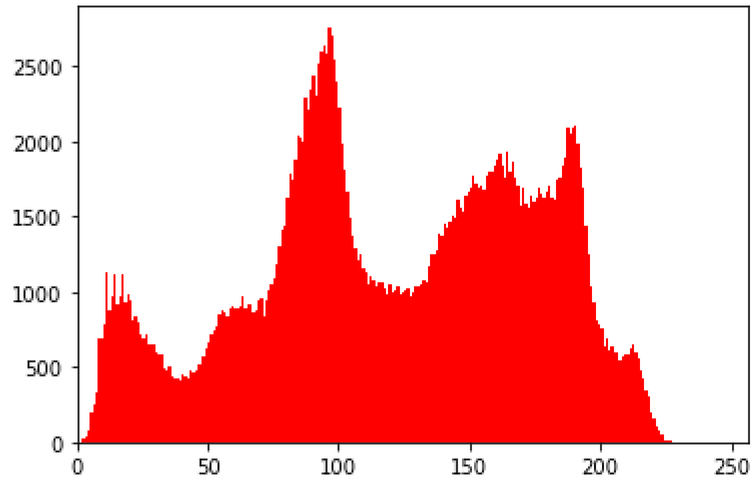
```
import numpy as np
import cv2

img = cv2.imread(r'C:/Users/hp/Downloads\tree.jpg',0)

# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)

plt.imshow(cl1)
cv2.imwrite('tree_2.jpg',cl1)
```

Out[56]:

```
True
```



In [68]:
```
import numpy as np
import cv2
from matplotlib import pyplot as plt
img = cv2.imread(r'C:/Users/hp/Downloads\peppers.bmp',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.show()
```

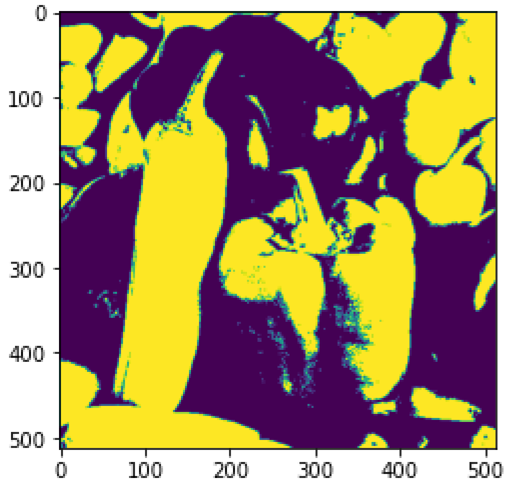


In [64]:
```
import cv2

originalImage = cv2.imread(r'C:/Users/hp/Downloads\peppers.bmp')
grayImage = cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY)
(thresh, blackAndWhiteImage) = cv2.threshold(grayImage, 127, 255, cv2.THRESH_BINA
cv2.imwrite('blackAndWhiteImage.png', blackAndWhiteImage)
cv2.imwrite('grayImage.png', grayImage)
```

Out[64]:
```
True
```

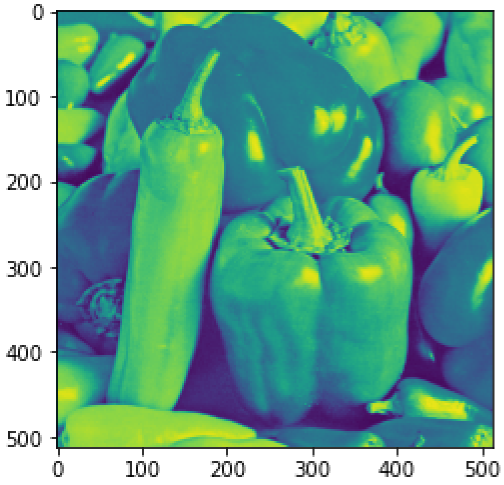
In [65]:
```
plt.imshow(blackAndWhiteImage)
```

Out[65]:
```
<matplotlib.image.AxesImage at 0x27f9a238160>
```



In [66]:
```
plt.imshow(grayImage)
```

Out[66]:
```
<matplotlib.image.AxesImage at 0x27f9bd89be0>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: