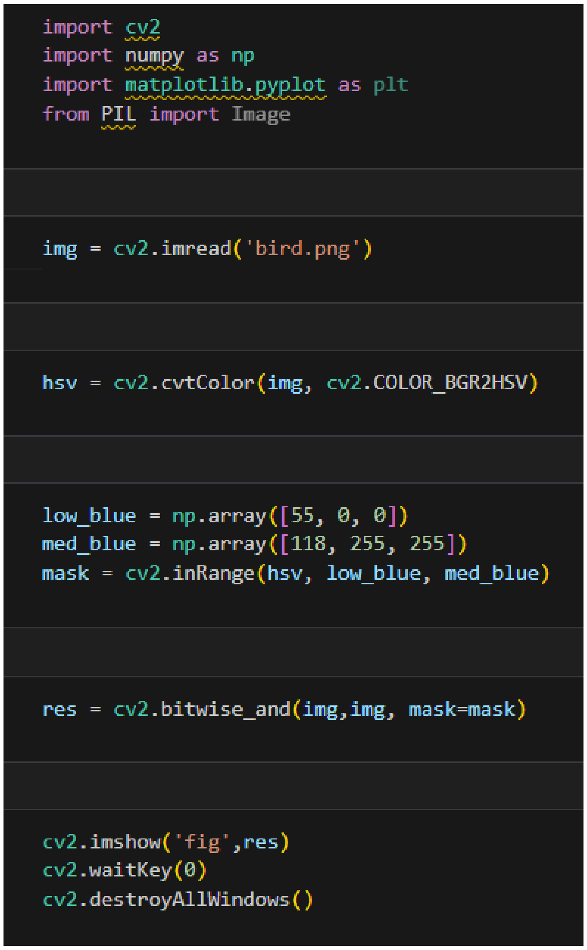
Part 1 :

Color Models of Digital Image Processing
RGB Model
HIS Model
Pseudo – Color Image Processing\



```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

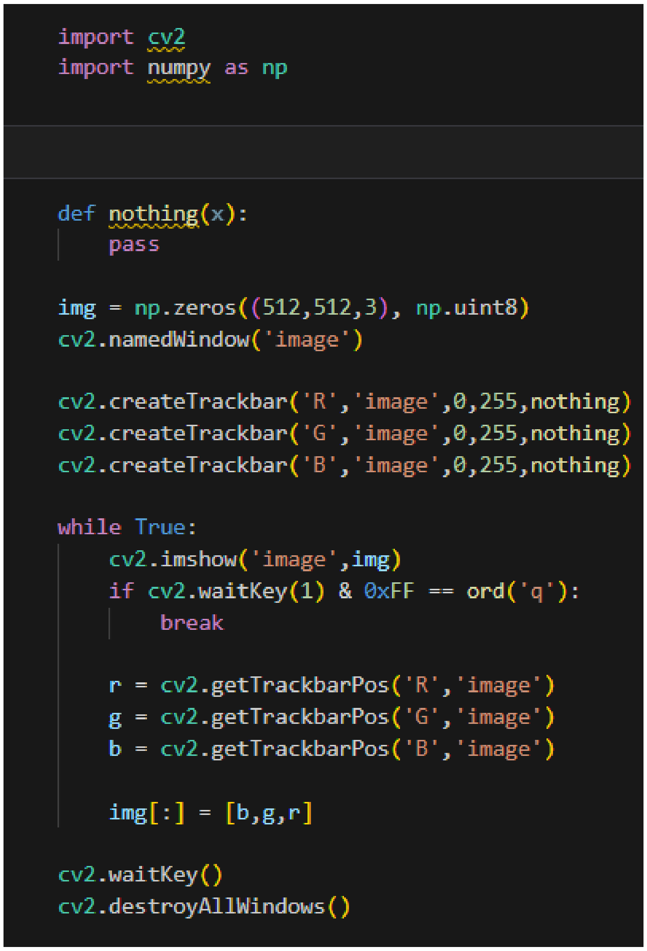
```
img = cv2.imread('bird.png')
```

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
low_blue = np.array([55, 0, 0])
med_blue = np.array([118, 255, 255])
mask = cv2.inRange(hsv, low_blue, med_blue)
```

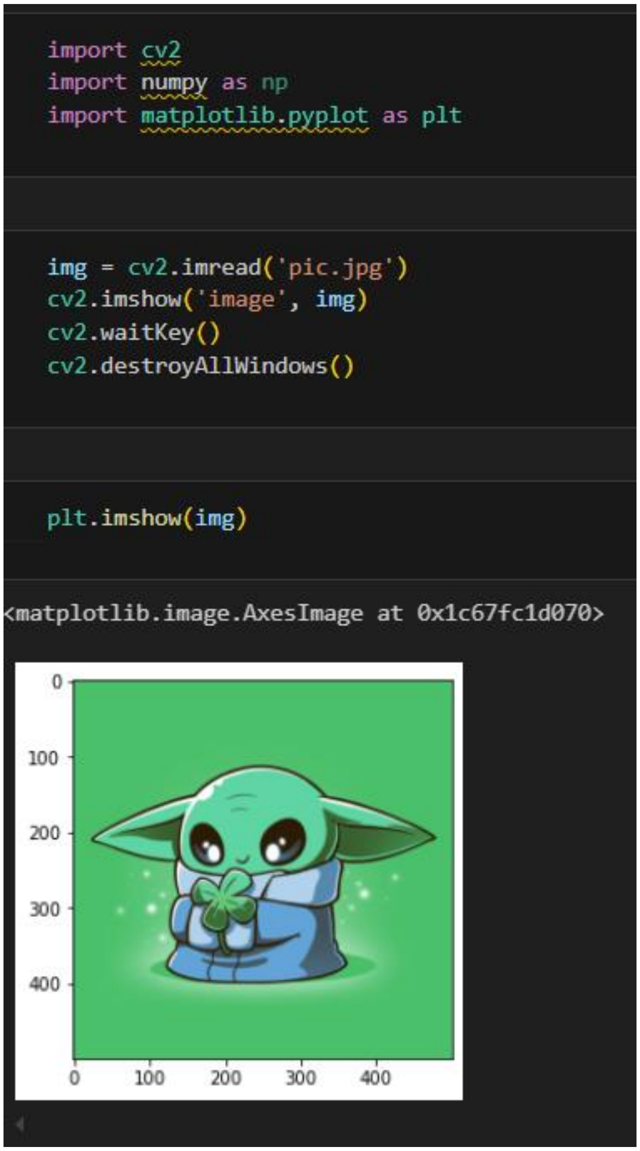
```
res = cv2.bitwise_and(img,img, mask=mask)
```

```
cv2.imshow('fig',res)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
import cv2
import numpy as np
```

```
def nothing(x):
    pass

img = np.zeros((512,512,3), np.uint8)
cv2.namedWindow('image')

cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

while True:
    cv2.imshow('image',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    r = cv2.getTrackbarPos('R','image')
    g = cv2.getTrackbarPos('G','image')
    b = cv2.getTrackbarPos('B','image')

    img[:] = [b,g,r]

cv2.waitKey()
cv2.destroyAllWindows()
```

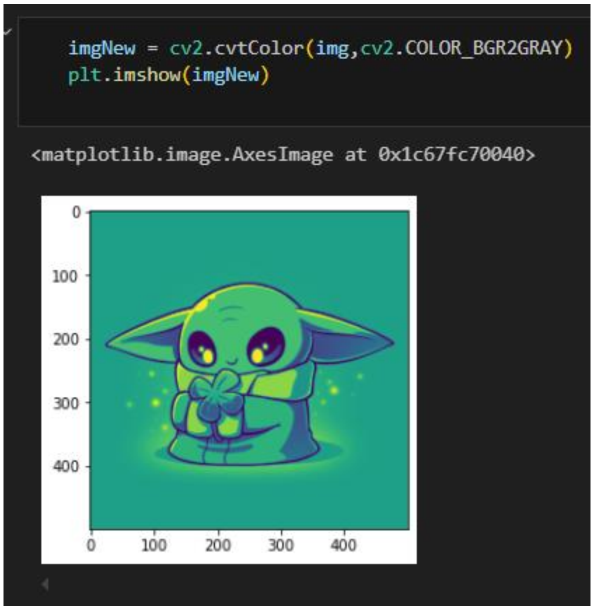


```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
img = cv2.imread('pic.jpg')
cv2.imshow('image', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

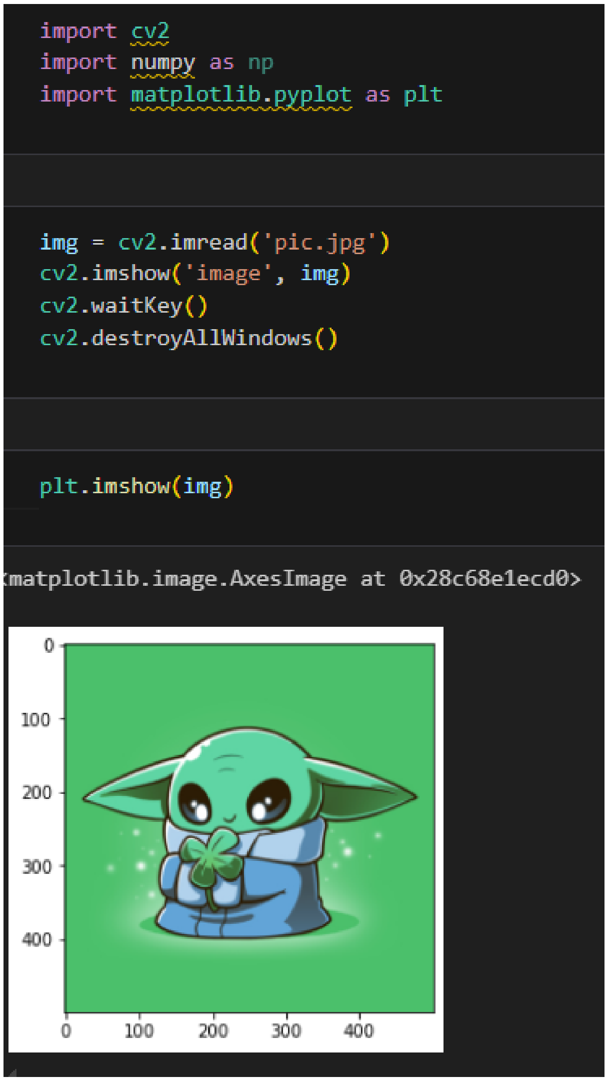
```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x1c67fc1d070>
```



```
imgNew = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
plt.imshow(imgNew)
```

```
<matplotlib.image.AxesImage at 0x1c67fc70040>
```

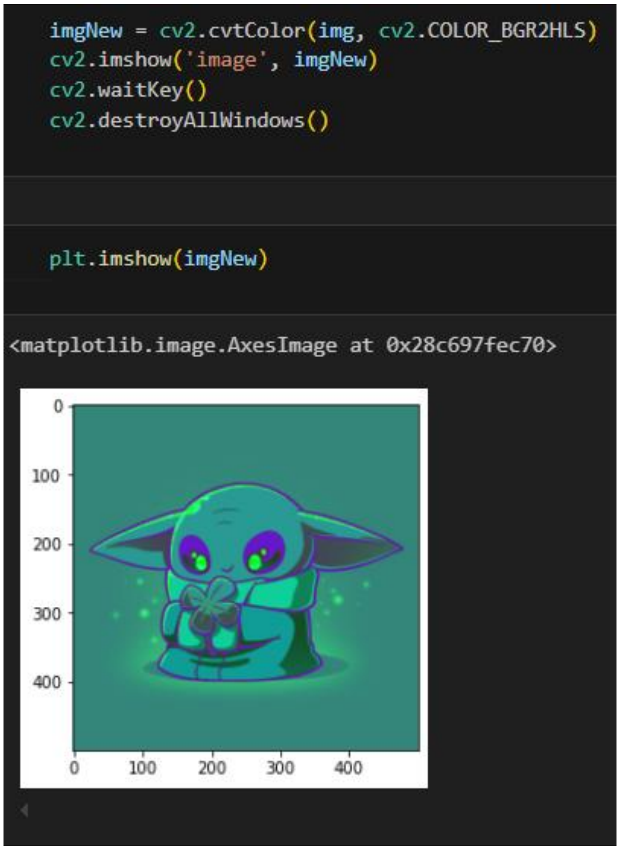


```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
img = cv2.imread('pic.jpg')
cv2.imshow('image', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
plt.imshow(img)
```

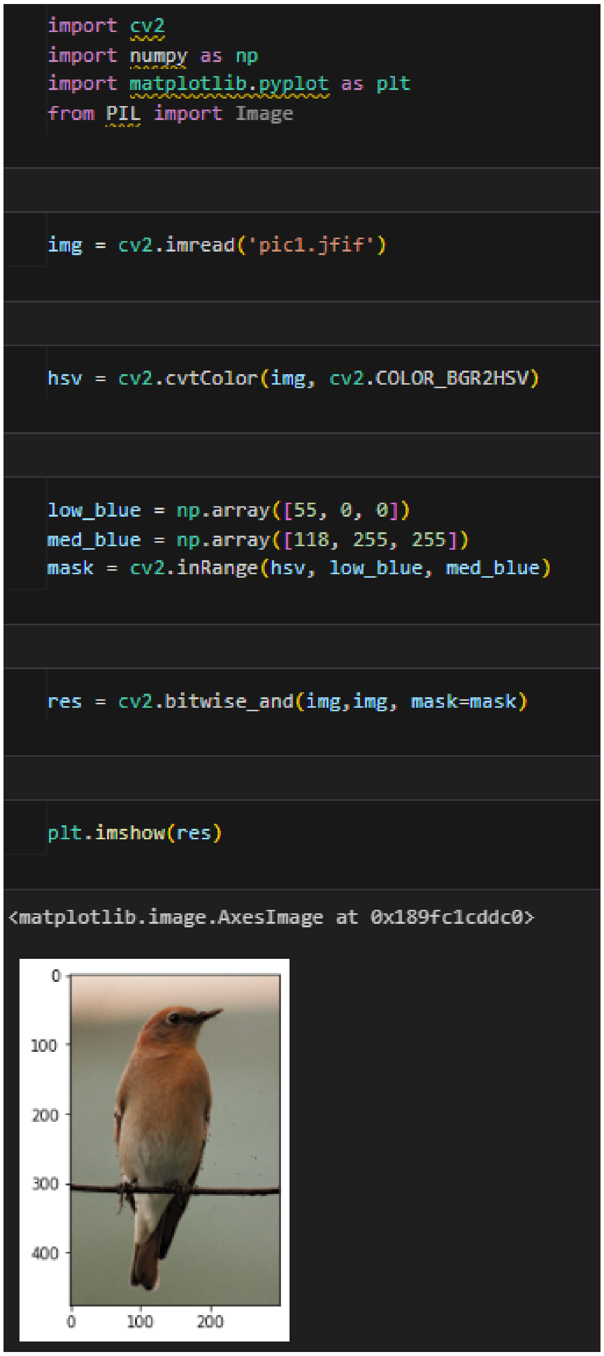
```
<matplotlib.image.AxesImage at 0x28c68e1ecd0>
```



```
imgNew = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
cv2.imshow('image', imgNew)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
plt.imshow(imgNew)
```

```
<matplotlib.image.AxesImage at 0x28c697fec70>
```



```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

```
img = cv2.imread('pic1.jfif')
```

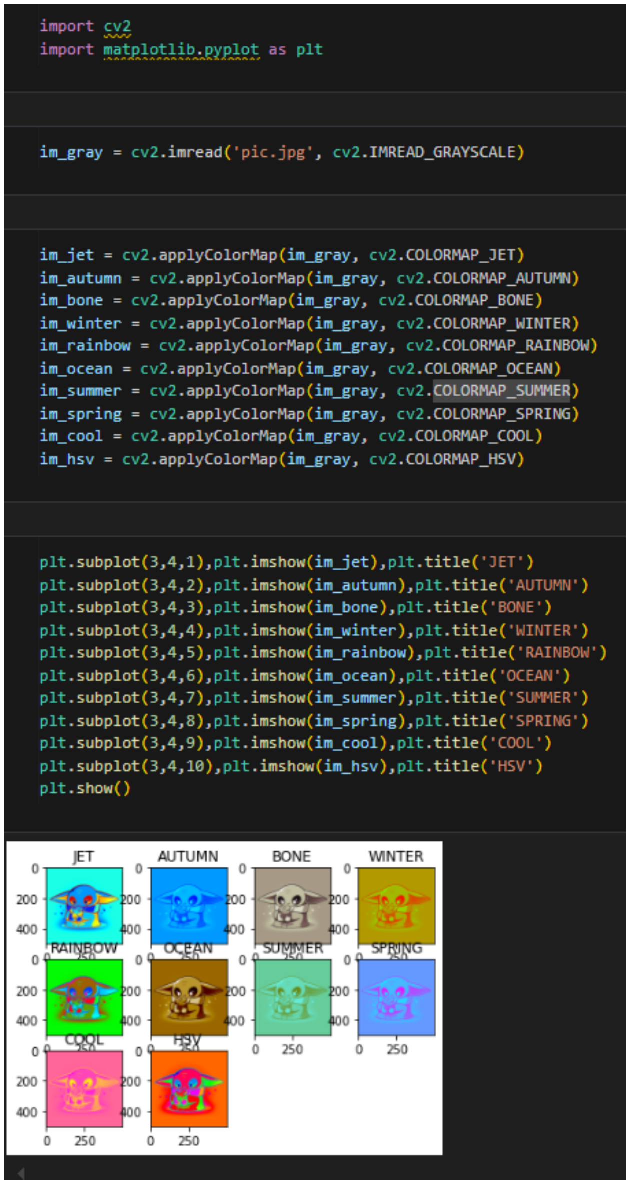
```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
low_blue = np.array([55, 0, 0])
med_blue = np.array([118, 255, 255])
mask = cv2.inRange(hsv, low_blue, med_blue)
```

```
res = cv2.bitwise_and(img,img, mask=mask)
```

```
plt.imshow(res)
```

```
<matplotlib.image.AxesImage at 0x189fc1cddc0>
```



```
import cv2
import matplotlib.pyplot as plt
```

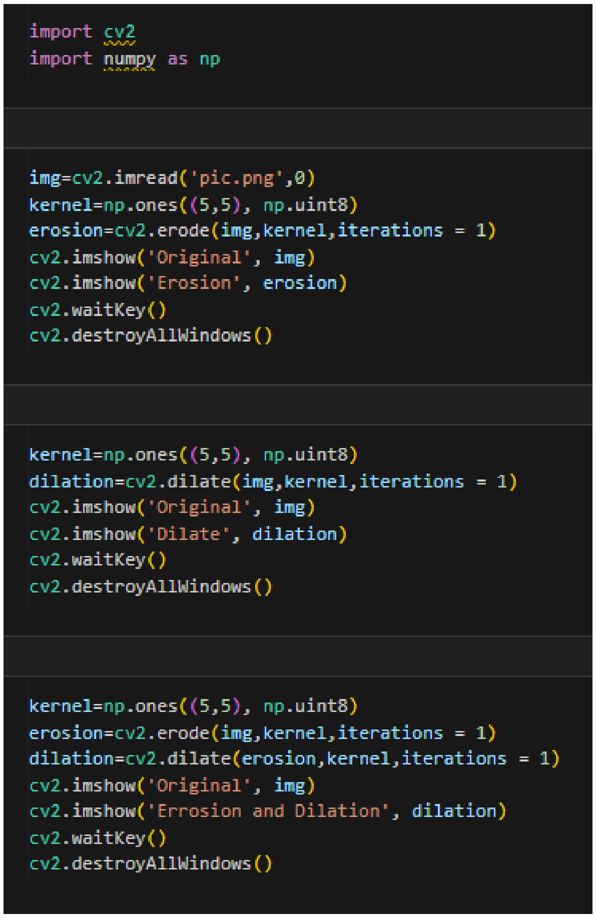
```
im_gray = cv2.imread('pic.jpg', cv2.IMREAD_GRAYSCALE)
```

```
im_jet = cv2.applyColorMap(im_gray, cv2.COLORMAP_JET)
im_autumn = cv2.applyColorMap(im_gray, cv2.COLORMAP_AUTUMN)
im_bone = cv2.applyColorMap(im_gray, cv2.COLORMAP_BONE)
im_winter = cv2.applyColorMap(im_gray, cv2.COLORMAP_WINTER)
im_rainbow = cv2.applyColorMap(im_gray, cv2.COLORMAP_RAINBOW)
im_ocean = cv2.applyColorMap(im_gray, cv2.COLORMAP_OCEAN)
im_summer = cv2.applyColorMap(im_gray, cv2.COLORMAP_SUMMER)
im_spring = cv2.applyColorMap(im_gray, cv2.COLORMAP_SPRING)
im_cool = cv2.applyColorMap(im_gray, cv2.COLORMAP_COOL)
im_hsv = cv2.applyColorMap(im_gray, cv2.COLORMAP_HSV)
```

```
plt.subplot(3,4,1),plt.imshow(im_jet),plt.title('JET')
plt.subplot(3,4,2),plt.imshow(im_autumn),plt.title('AUTUMN')
plt.subplot(3,4,3),plt.imshow(im_bone),plt.title('BONE')
plt.subplot(3,4,4),plt.imshow(im_winter),plt.title('WINTER')
plt.subplot(3,4,5),plt.imshow(im_rainbow),plt.title('RAINBOW')
plt.subplot(3,4,6),plt.imshow(im_ocean),plt.title('OCEAN')
plt.subplot(3,4,7),plt.imshow(im_summer),plt.title('SUMMER')
plt.subplot(3,4,8),plt.imshow(im_spring),plt.title('SPRING')
plt.subplot(3,4,9),plt.imshow(im_cool),plt.title('COOL')
plt.subplot(3,4,10),plt.imshow(im_hsv),plt.title('HSV')
plt.show()
```

Part 2:

Simple Thresholding
Segmentation through Manual Thresholding
Segmentation through OpenCV Thresholding

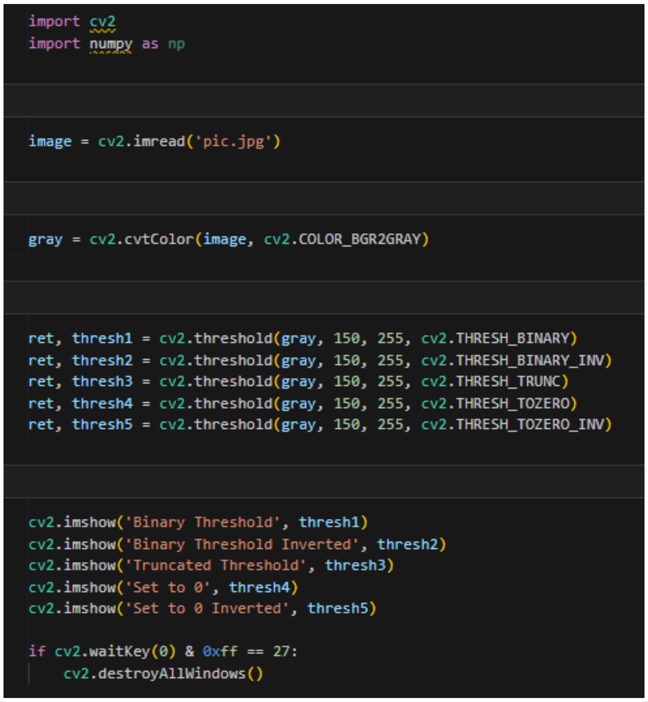


```
import cv2
import numpy as np
```

```
img=cv2.imread('pic.png',0)
kernel=np.ones((5,5), np.uint8)
erosion=cv2.erode(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Erosion', erosion)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
kernel=np.ones((5,5), np.uint8)
dilation=cv2.dilate(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Dilate', dilation)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
kernel=np.ones((5,5), np.uint8)
erosion=cv2.erode(img,kernel,iterations = 1)
dilation=cv2.dilate(erosion,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Errosion and Dilation', dilation)
cv2.waitKey()
cv2.destroyAllWindows()
```



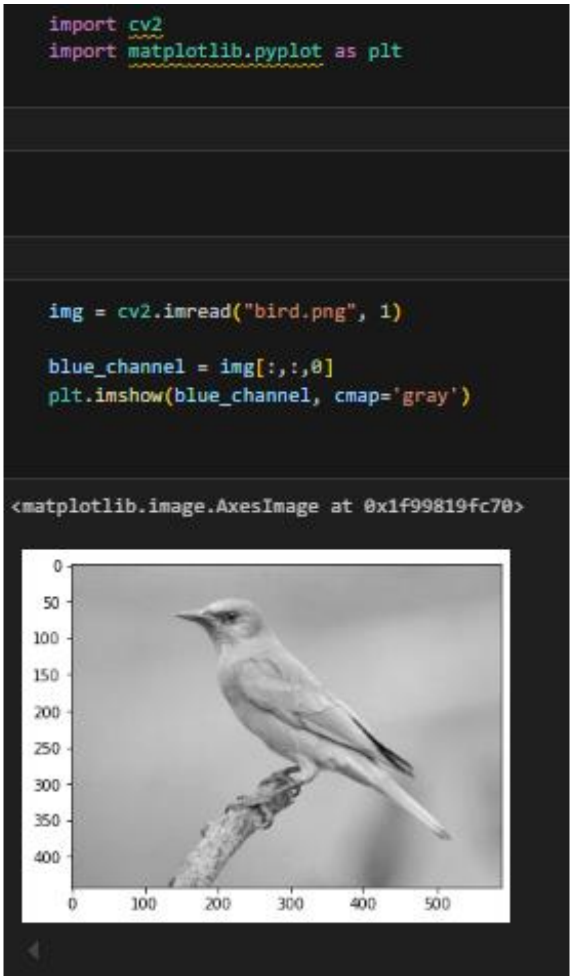
```
import cv2
import numpy as np
```

```
image = cv2.imread('pic.jpg')
```

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
ret, thresh1 = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(gray, 150, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(gray, 150, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(gray, 150, 255, cv2.THRESH_TOZERO_INV)
```

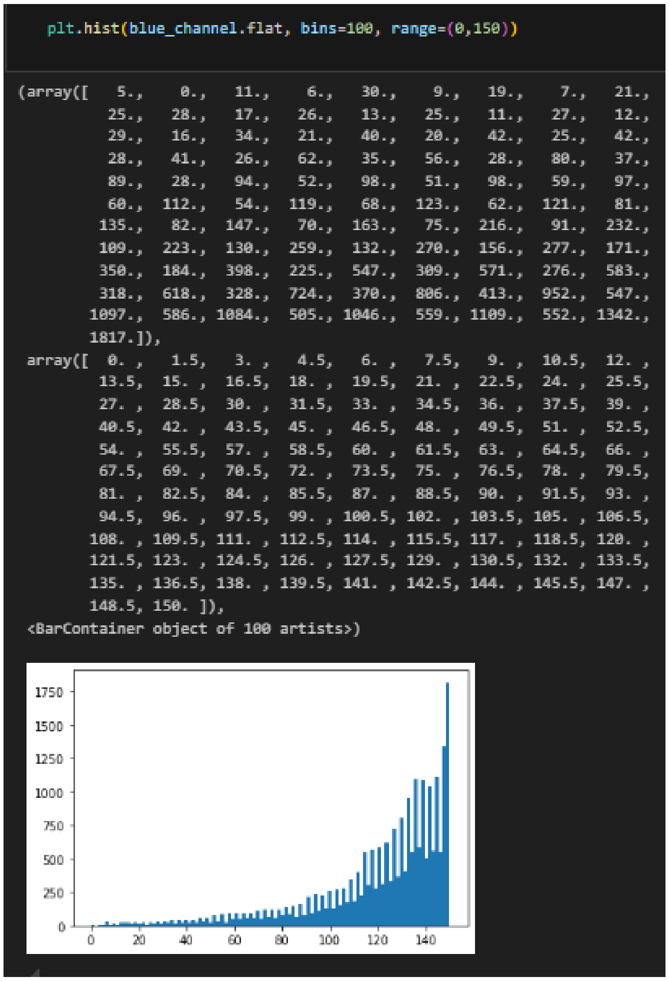
```
cv2.imshow('Binary Threshold', thresh1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4)
cv2.imshow('Set to 0 Inverted', thresh5)

if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```



```
import cv2
import matplotlib.pyplot as plt
```

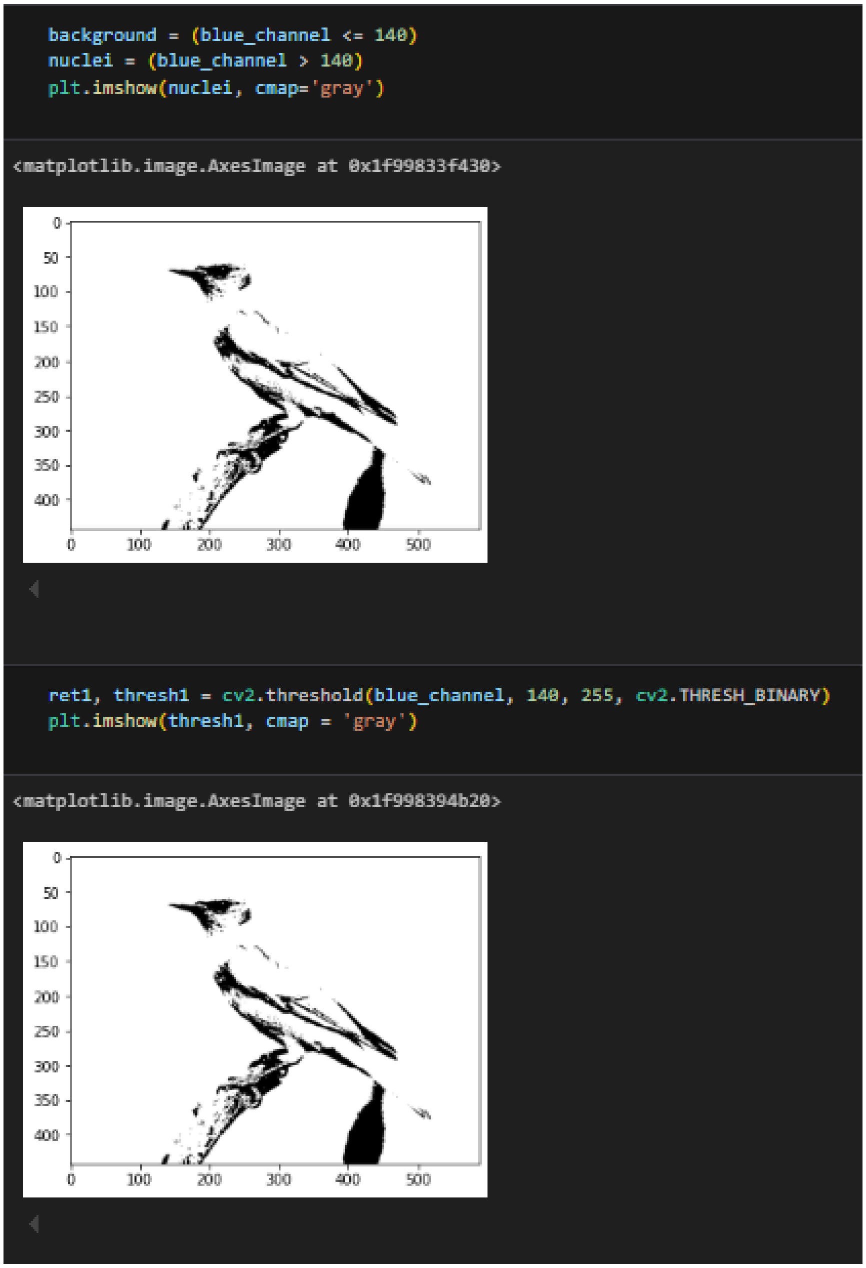
```
img = cv2.imread("bird.png", 1)

blue_channel = img[:,:,0]
plt.imshow(blue_channel, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1f99819fc70>
```



```
plt.hist(blue_channel.flat, bins=100, range=(0,150))
```

```
(array([   5.,    0.,   11.,    6.,   30.,    9.,   19.,    7.,   21.,
          25.,   28.,   17.,   26.,   13.,   25.,   11.,   27.,   12.,
          29.,   16.,   34.,   21.,   40.,   20.,   42.,   25.,   42.,
          28.,   41.,   26.,   62.,   35.,   56.,   28.,   80.,   37.,
          89.,   28.,   94.,   52.,   98.,   51.,   98.,   59.,   97.,
          60.,  112.,   54.,  119.,   68.,  123.,   62.,  121.,   81.,
         135.,   82.,  147.,   70.,  163.,   75.,  216.,   91.,  232.,
         109.,  223.,  130.,  259.,  132.,  270.,  156.,  277.,  171.,
         350.,  184.,  398.,  225.,  547.,  309.,  571.,  276.,  583.,
         318.,  618.,  328.,  724.,  370.,  806.,  413.,  952.,  547.,
        1097.,  586., 1084.,  505., 1046.,  559., 1109.,  552., 1342.,
        1817.]),
 array([  0. ,   1.5,   3. ,   4.5,   6. ,   7.5,   9. ,  10.5,  12. ,
         13.5,  15. ,  16.5,  18. ,  19.5,  21. ,  22.5,  24. ,  25.5,
         27. ,  28.5,  30. ,  31.5,  33. ,  34.5,  36. ,  37.5,  39. ,
         40.5,  42. ,  43.5,  45. ,  46.5,  48. ,  49.5,  51. ,  52.5,
         54. ,  55.5,  57. ,  58.5,  60. ,  61.5,  63. ,  64.5,  66. ,
         67.5,  69. ,  70.5,  72. ,  73.5,  75. ,  76.5,  78. ,  79.5,
         81. ,  82.5,  84. ,  85.5,  87. ,  88.5,  90. ,  91.5,  93. ,
         94.5,  96. ,  97.5,  99. , 100.5, 102. , 103.5, 105. , 106.5,
        108. , 109.5, 111. , 112.5, 114. , 115.5, 117. , 118.5, 120. ,
        121.5, 123. , 124.5, 126. , 127.5, 129. , 130.5, 132. , 133.5,
        135. , 136.5, 138. , 139.5, 141. , 142.5, 144. , 145.5, 147. ,
        148.5, 150. ]),
 <BarContainer object of 100 artists>)
```



```
background = (blue_channel <= 140)
nuclei = (blue_channel > 140)
plt.imshow(nuclei, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x1f99833f430>
```

```
ret1, thresh1 = cv2.threshold(blue_channel, 140, 255, cv2.THRESH_BINARY)
plt.imshow(thresh1, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x1f998394b20>
```