IMAGE PROCESSING WITH PYTHON

IMAGE PROCESSING TOOLS

IMAGE PROCESSING TOOLS

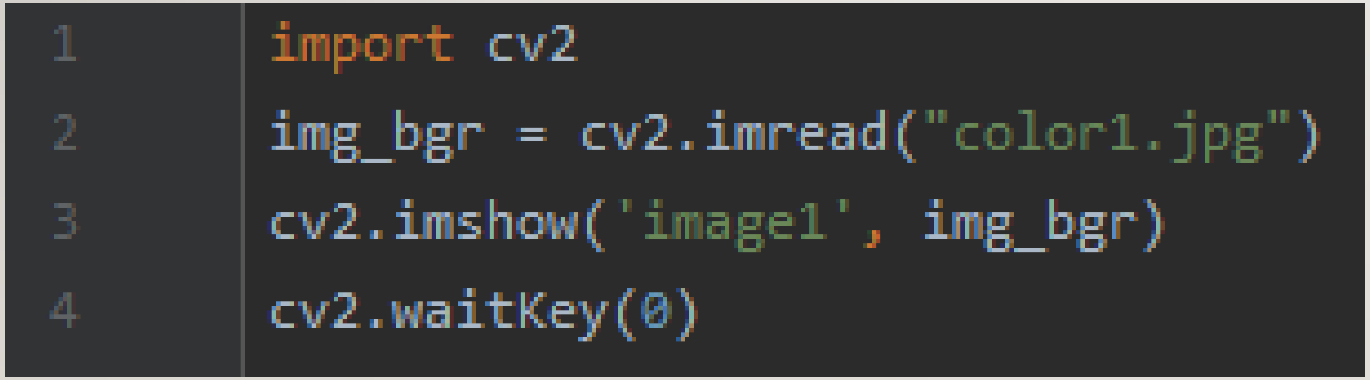
- **OpenCV**
  - Python Data Visualization library
- **PIL/pillow**
  - image processing library
- **Scikit-image**
  - image-processing library
- **NumPy**
  - scientific computing library
- **SciPy**
  - scientific computational library
- **Mahotas**
  - image processing operations
- **SimpleITK**
  - image registration and segmentation
- **Matplotlib**
  - data visualization lib

https://www.kdnuggets.com/2022/11/8-best-python-image-manipulation-tools.html

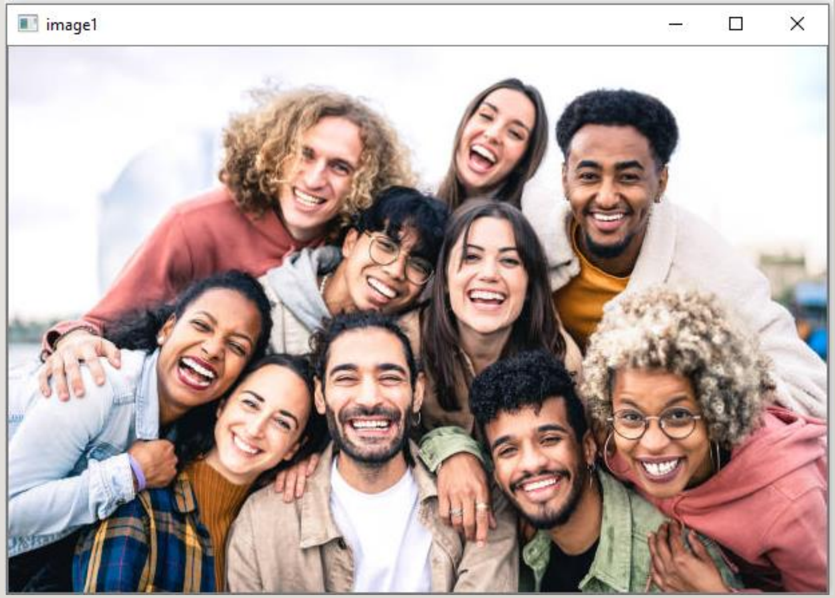
READ AND SHOW IMAGE

CV2 – CV2

- CV2 read and show image in blue-green-red mode

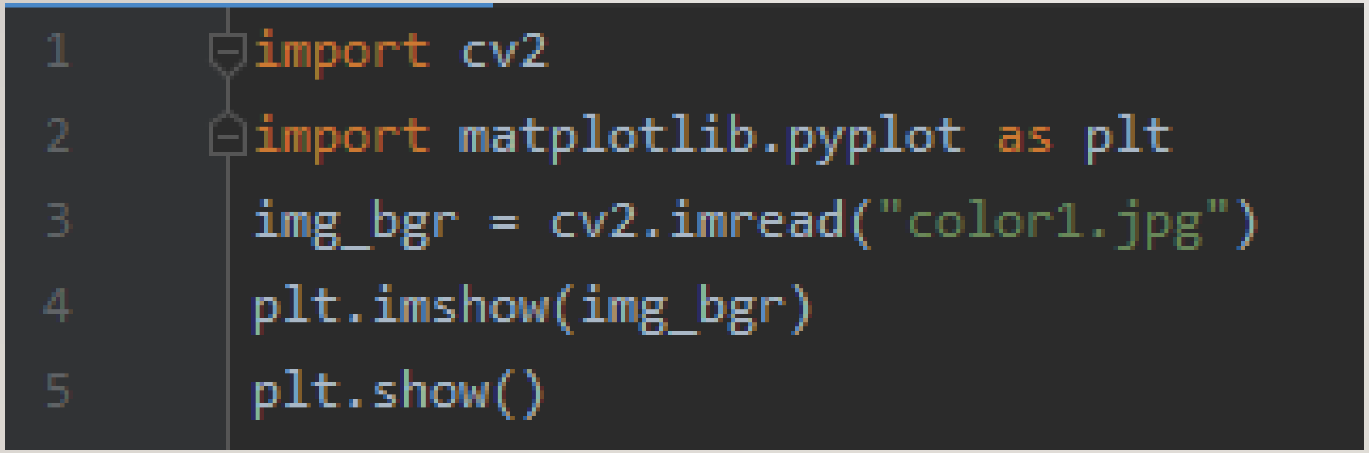


```python
import cv2
img_bgr = cv2.imread("color1.jpg")
cv2.imshow('image1', img_bgr)
cv2.waitKey(0)
```

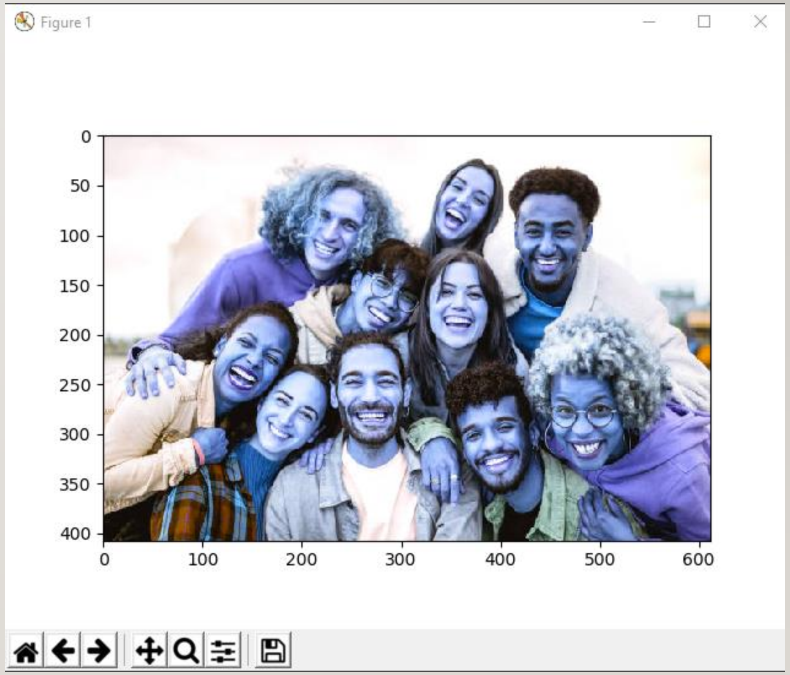


CV2 – MATPLOTLIB

- CV2 read image in blue-green-red format
- Matplotlib use red-green-blue format

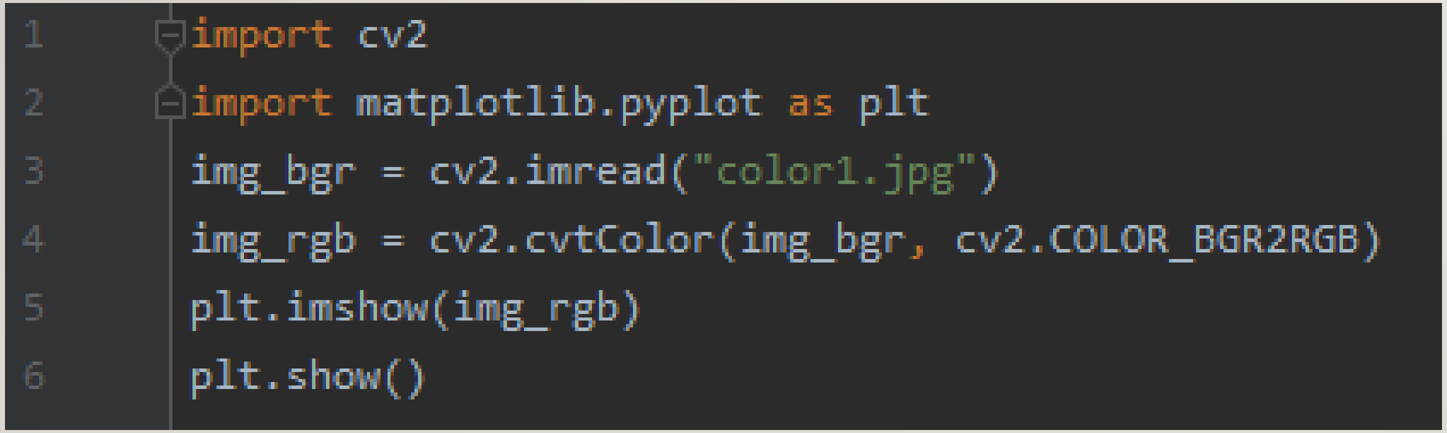


```python
import cv2
import matplotlib.pyplot as plt
img_bgr = cv2.imread("color1.jpg")
plt.imshow(img_bgr)
plt.show()
```

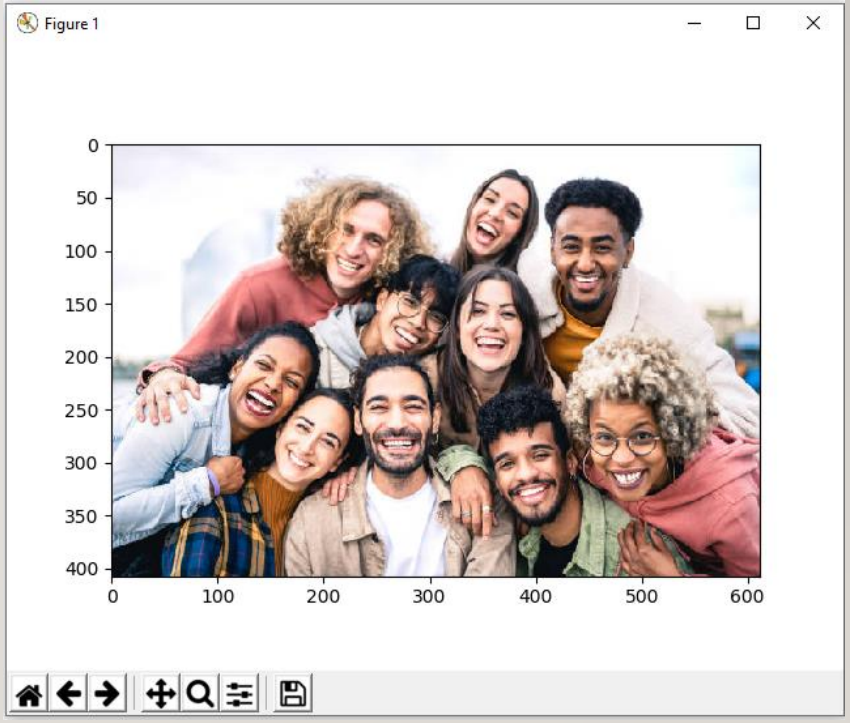


CV2 – MATPLOTLIB

- Convert BGR to RGB

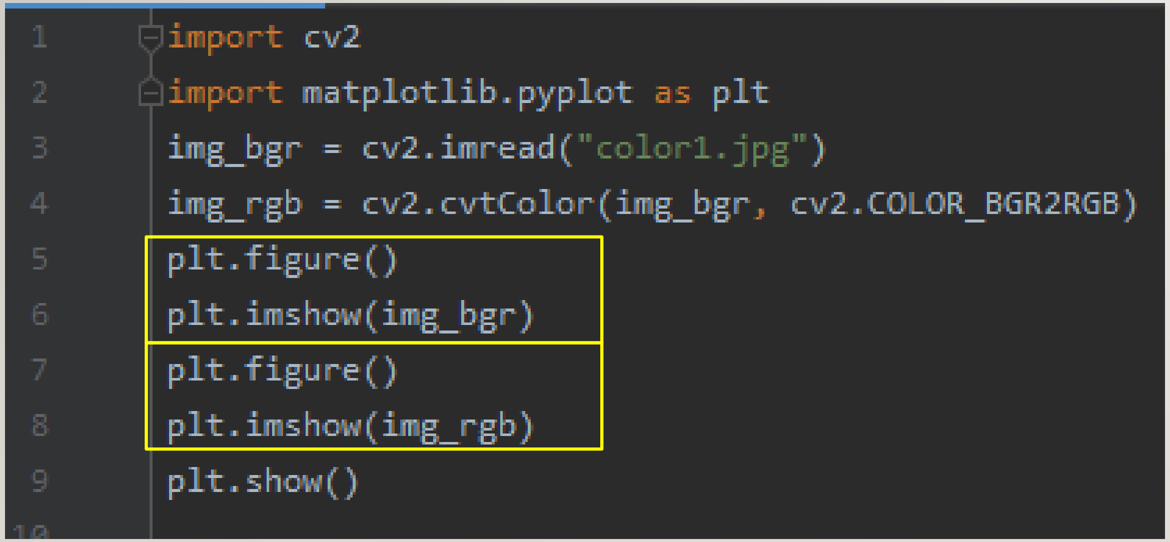


```python
import cv2
import matplotlib.pyplot as plt
img_bgr = cv2.imread("color1.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
plt.show()
```

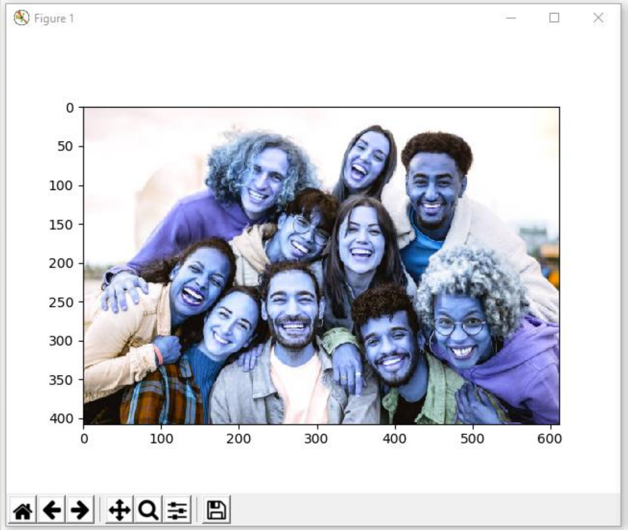


CV2 – MATPLOTLIB

- Show two figure



```
import cv2
import matplotlib.pyplot as plt
img_bgr = cv2.imread("color1.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(img_bgr)
plt.figure()
plt.imshow(img_rgb)
plt.show()
```



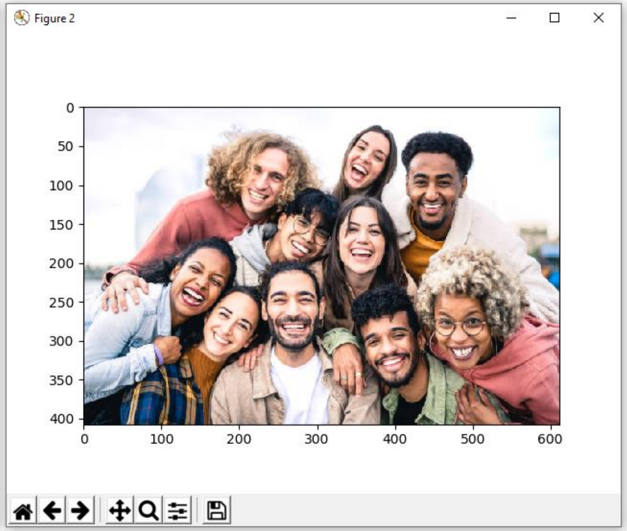
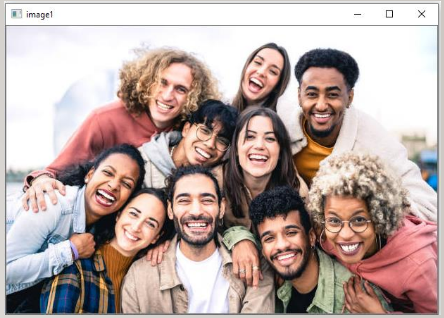
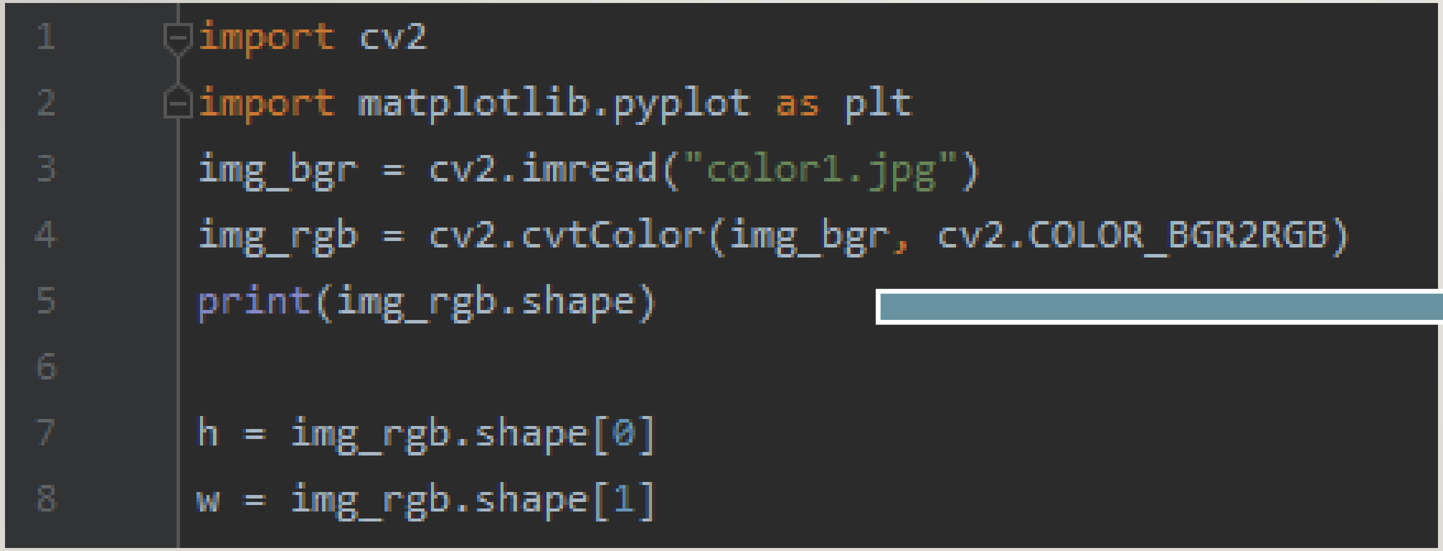


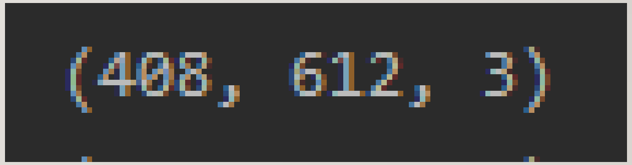
IMAGE SHAPE



- Image store with 3D array (height, width, color_channel)
- Dimension of image stored in image.shape



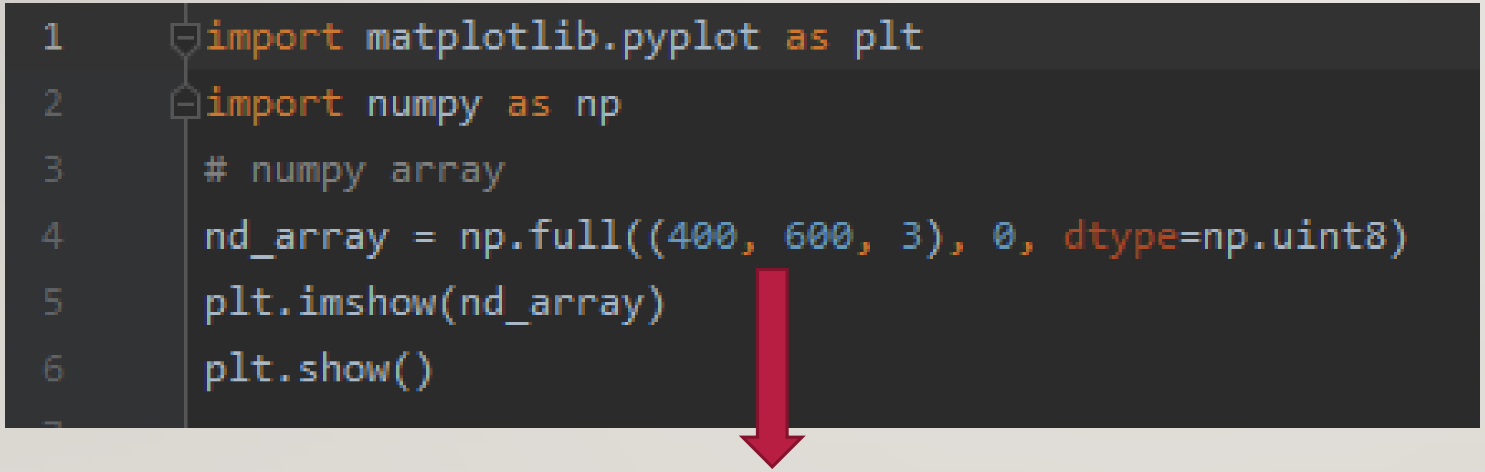
```python
import cv2
import matplotlib.pyplot as plt
img_bgr = cv2.imread("color1.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
print(img_rgb.shape)

h = img_rgb.shape[0]
w = img_rgb.shape[1]
```



```
(408, 612, 3)
```

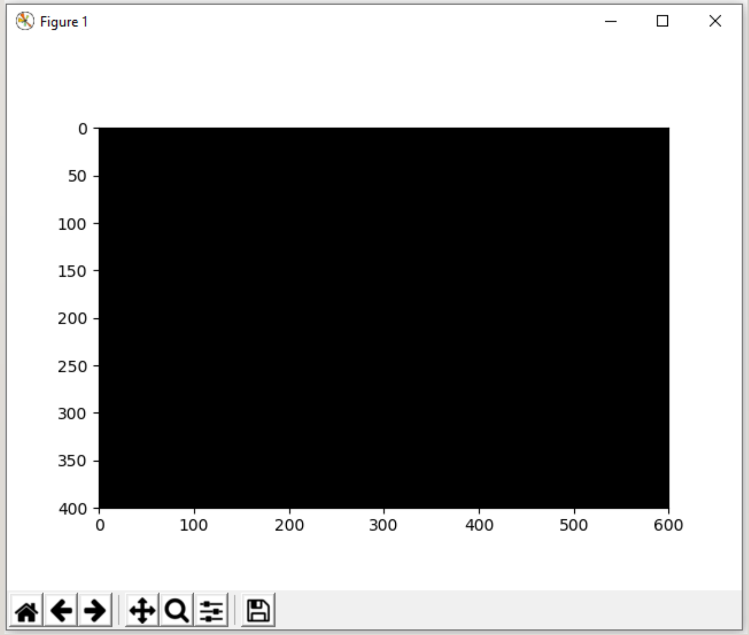
CREATE NUMPY ARRAY AS A IMAGE

- **numpy.full(shape, fill_value, dtype)**
  - Return a new array with the same shape and type as a given array filled with a fill_value.

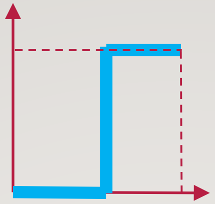


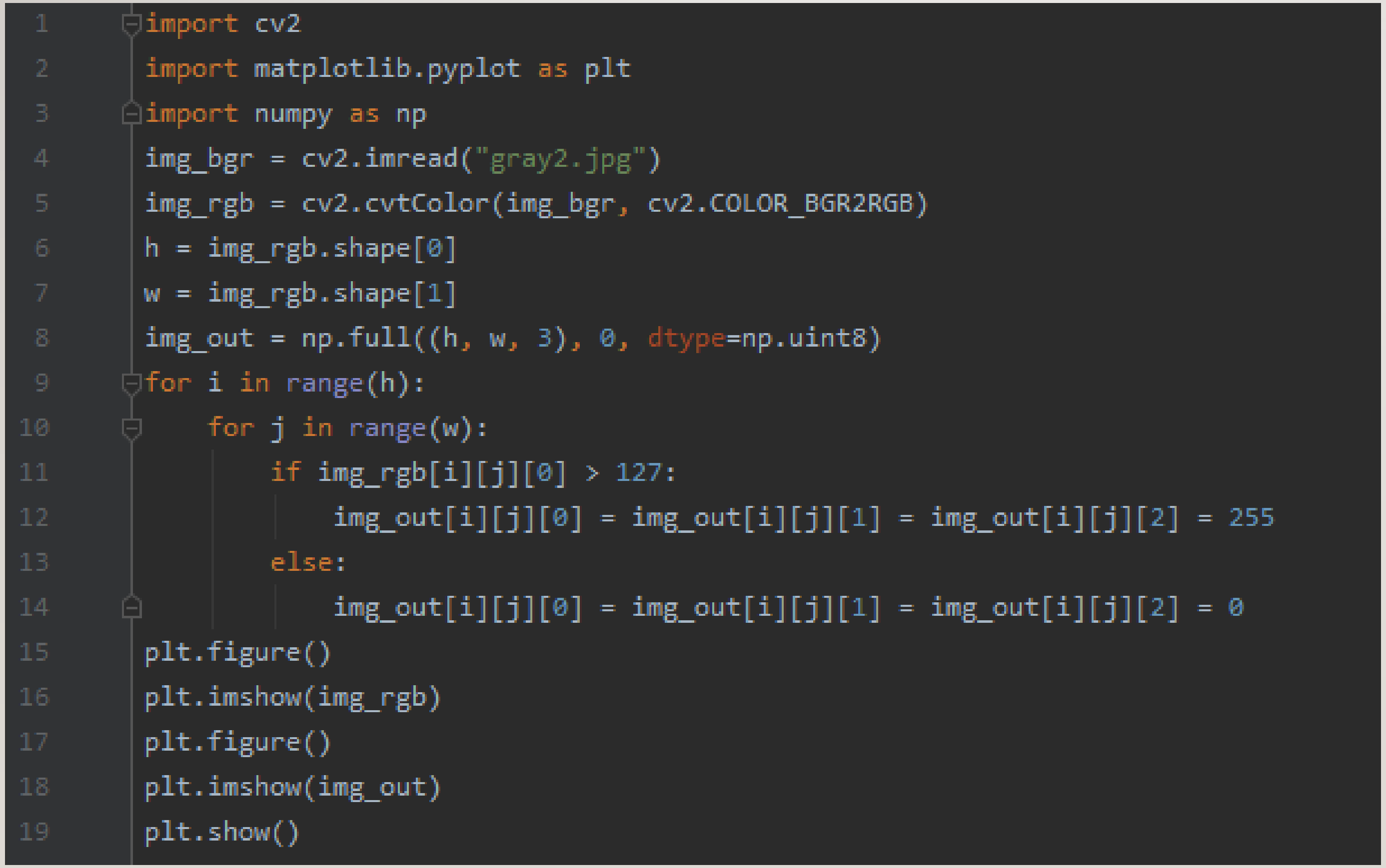
```python
import matplotlib.pyplot as plt
import numpy as np
# numpy array
nd_array = np.full((400, 600, 3), 0, dtype=np.uint8)
plt.imshow(nd_array)
plt.show()
```

(H,W,dim)

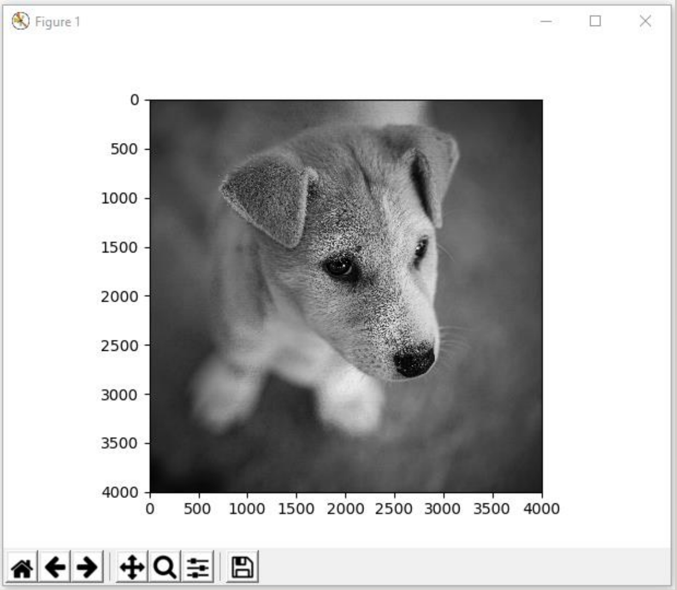


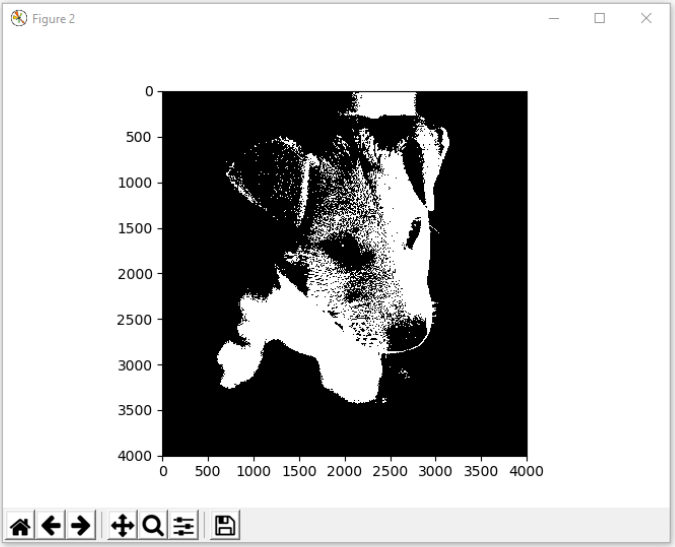
THRESHOLDING



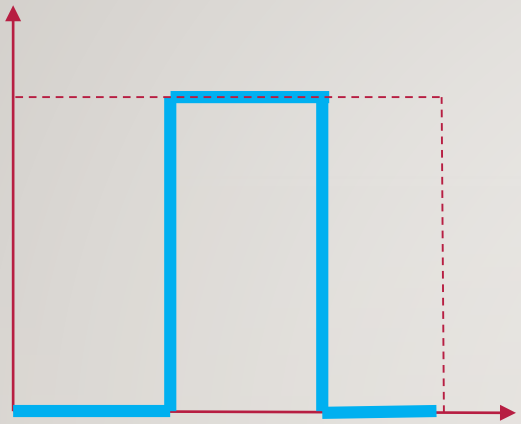


```python
import cv2
import matplotlib.pyplot as plt
import numpy as np
img_bgr = cv2.imread("gray2.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
h = img_rgb.shape[0]
w = img_rgb.shape[1]
img_out = np.full((h, w, 3), 0, dtype=np.uint8)
for i in range(h):
    for j in range(w):
        if img_rgb[i][j][0] > 127:
            img_out[i][j][0] = img_out[i][j][1] = img_out[i][j][2] = 255
        else:
            img_out[i][j][0] = img_out[i][j][1] = img_out[i][j][2] = 0
plt.figure()
plt.imshow(img_rgb)
plt.figure()
plt.imshow(img_out)
plt.show()
```

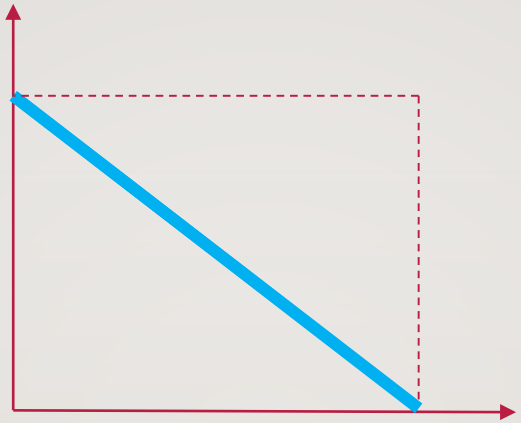




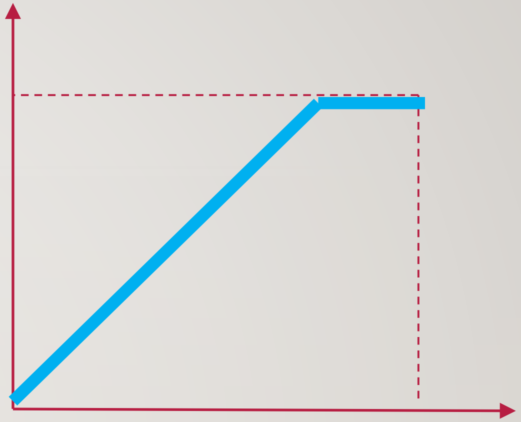
Apply to this threshold function



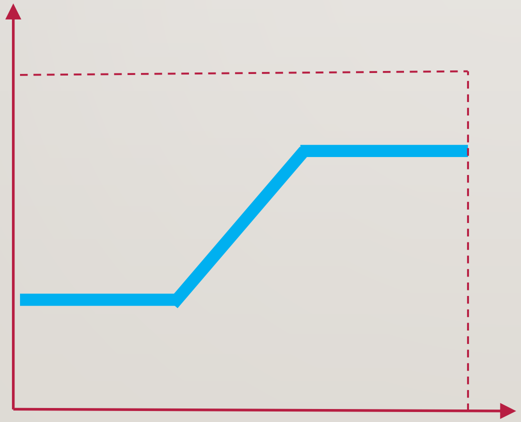
thresholding



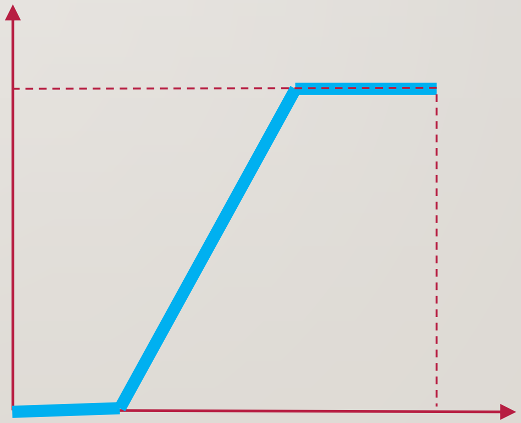
negative



Multiply by constant



clipping

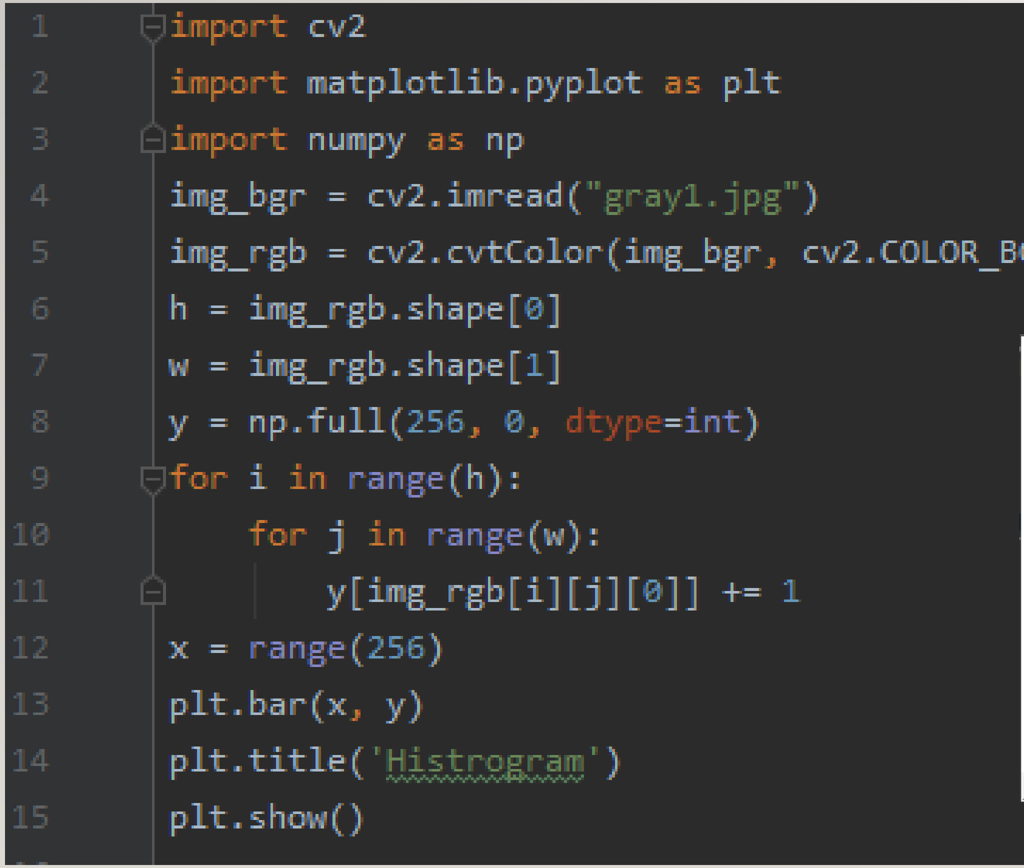


clipping

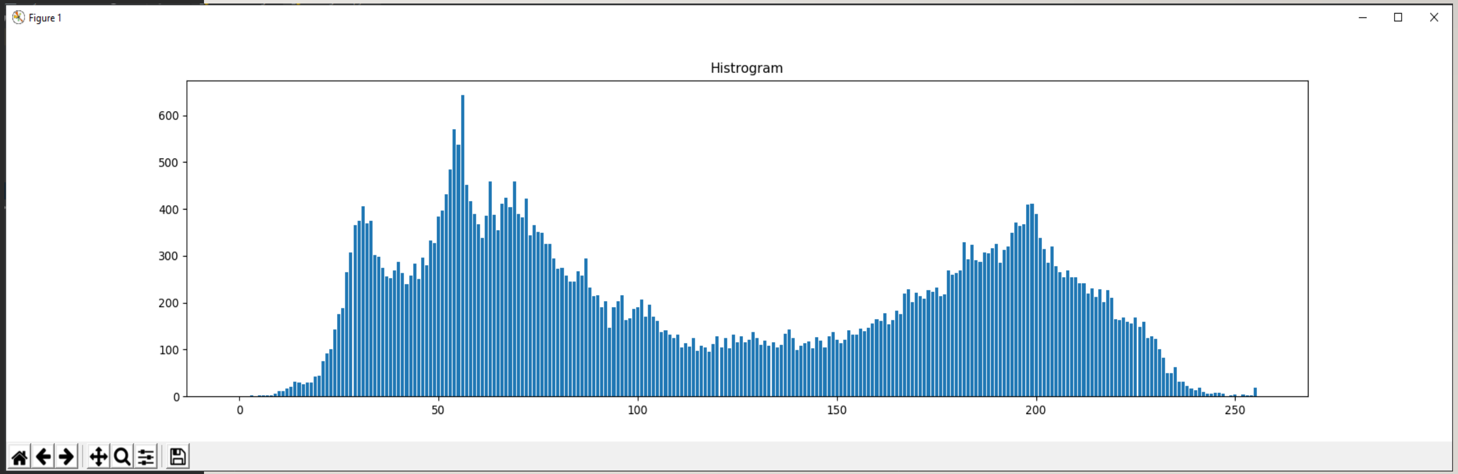
Apply to this grayscale image

- Addition
- Subtraction
- Histogram
- Automatic Thresholding

HISTROGRAM

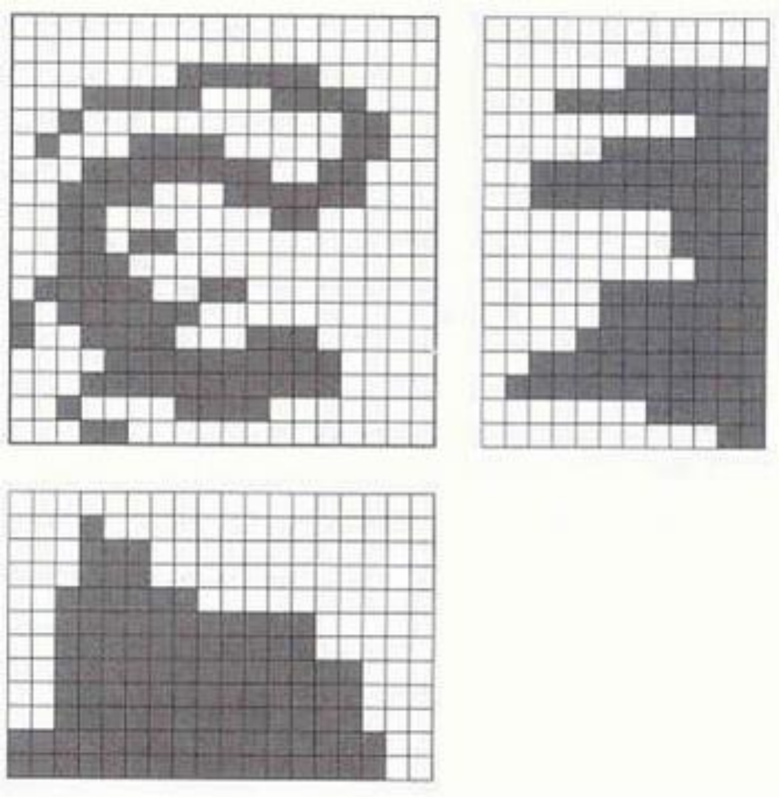


```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img_bgr = cv2.imread("gray1.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
h = img_rgb.shape[0]
w = img_rgb.shape[1]
y = np.full(256, 0, dtype=int)
for i in range(h):
    for j in range(w):
        y[img_rgb[i][j][0]] += 1
x = range(256)
plt.bar(x, y)
plt.title('Histrogram')
plt.show()
```

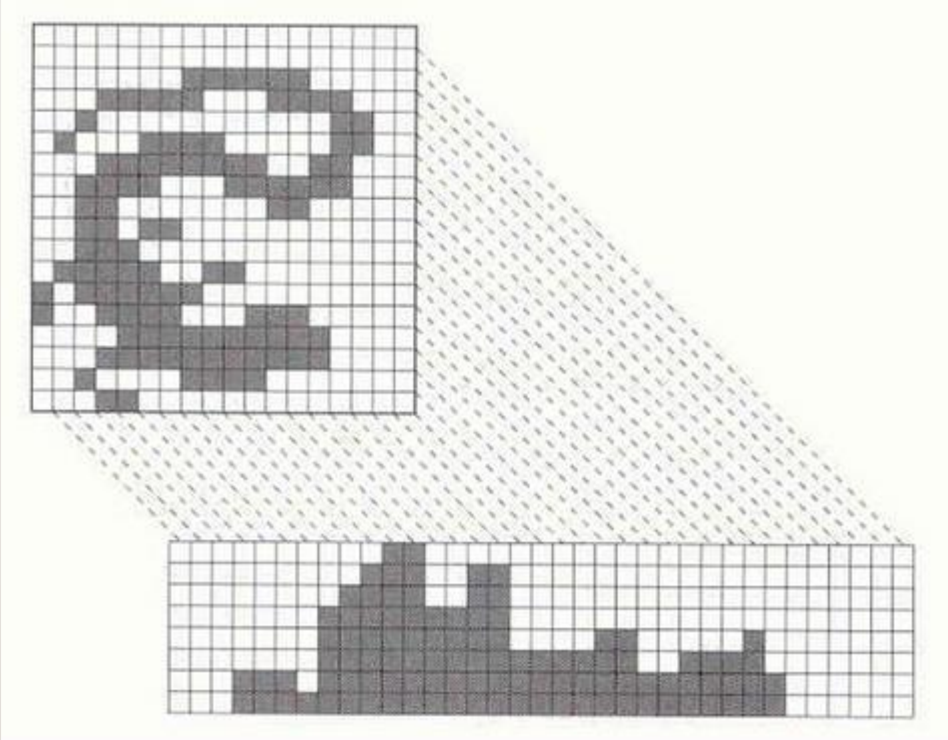


Apply to this algorithm

Horizontal projection

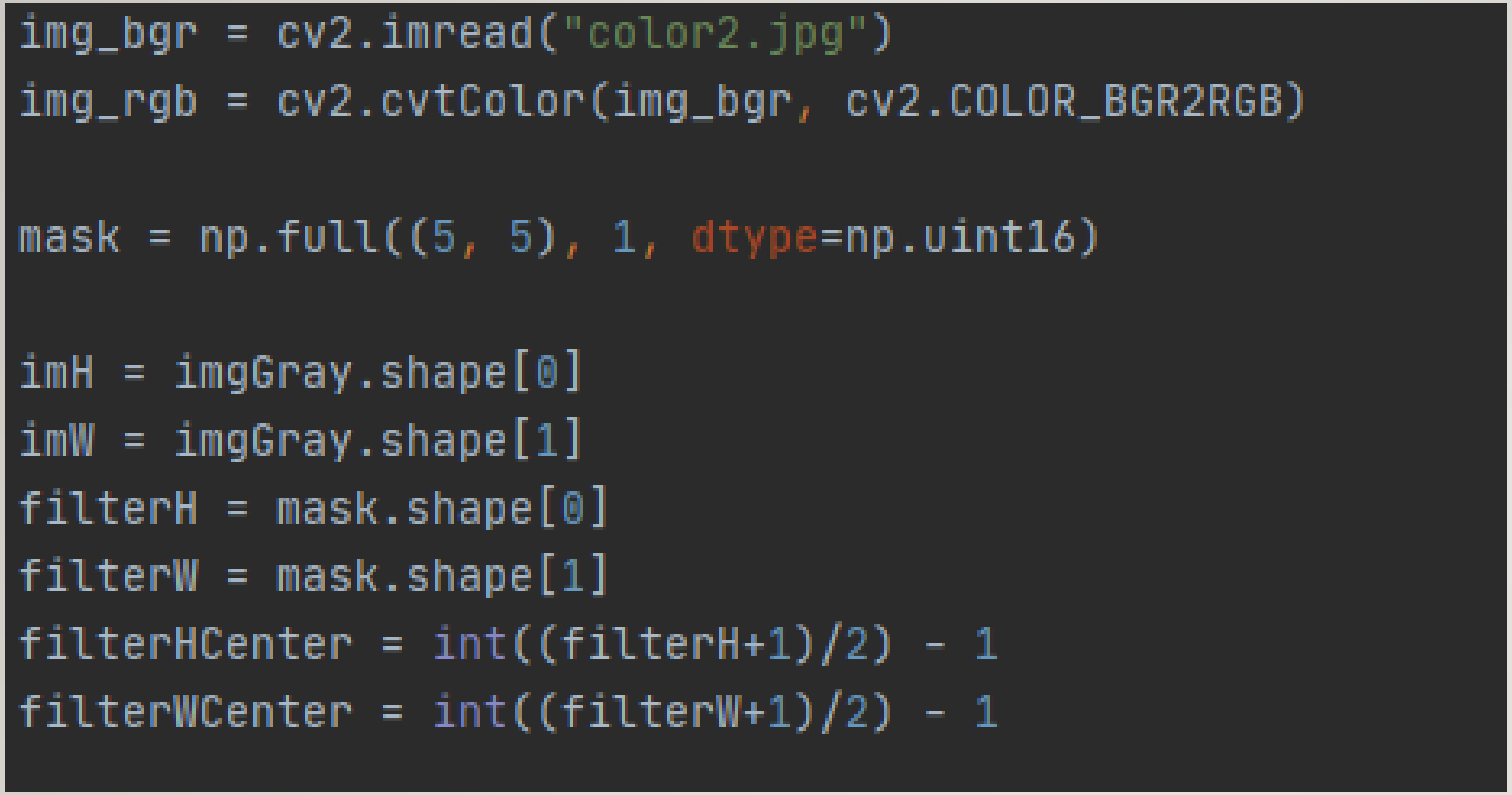


Vertical projection

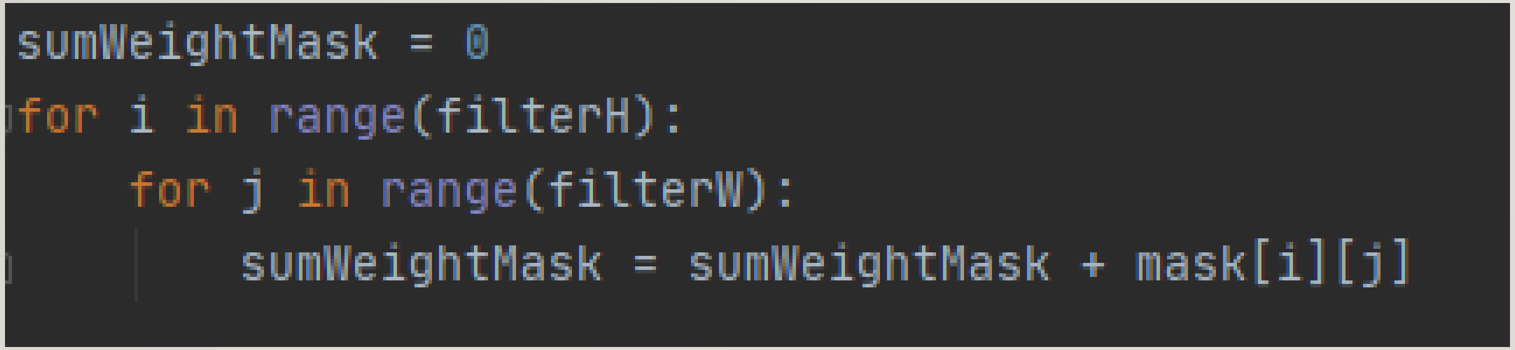


Diagonal projection

APPLY MASK TO IMAGE

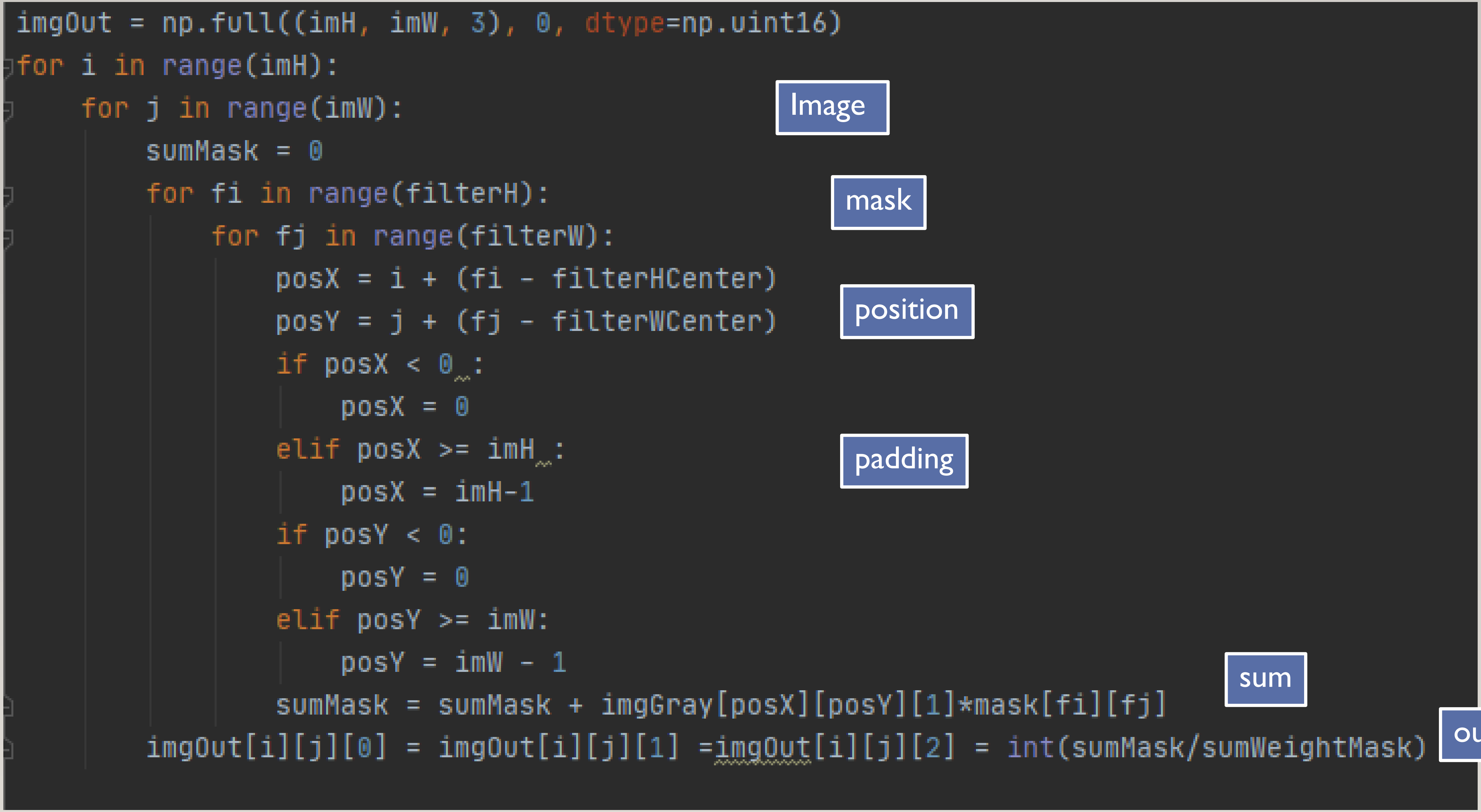


```python
img_bgr = cv2.imread("color2.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

mask = np.full((5, 5), 1, dtype=np.uint16)

imH = imgGray.shape[0]
imW = imgGray.shape[1]
filterH = mask.shape[0]
filterW = mask.shape[1]
filterHCenter = int((filterH+1)/2) - 1
filterWCenter = int((filterW+1)/2) - 1
```



```python
sumWeightMask = 0
for i in range(filterH):
    for j in range(filterW):
        sumWeightMask = sumWeightMask + mask[i][j]
```

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



```python
imgOut = np.full((imH, imW, 3), 0, dtype=np.uint16)
for i in range(imH):
    for j in range(imW):
        sumMask = 0
        for fi in range(filterH):
            for fj in range(filterW):
                posX = i + (fi - filterHCenter)
                posY = j + (fj - filterWCenter)
                if posX < 0 :
                    posX = 0
                elif posX >= imH :
                    posX = imH-1
                if posY < 0:
                    posY = 0
                elif posY >= imW:
                    posY = imW - 1
                sumMask = sumMask + imgGray[posX][posY][1]*mask[fi][fj]
        imgOut[i][j][0] = imgOut[i][j][1] =imgOut[i][j][2] = int(sumMask/sumWeightMask)
```

Image

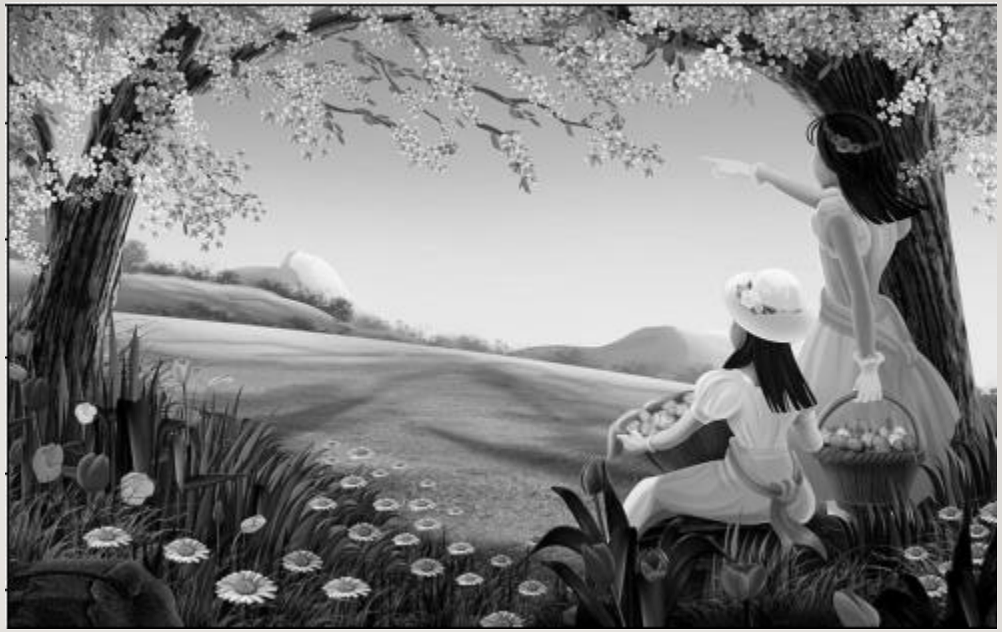
mask

position

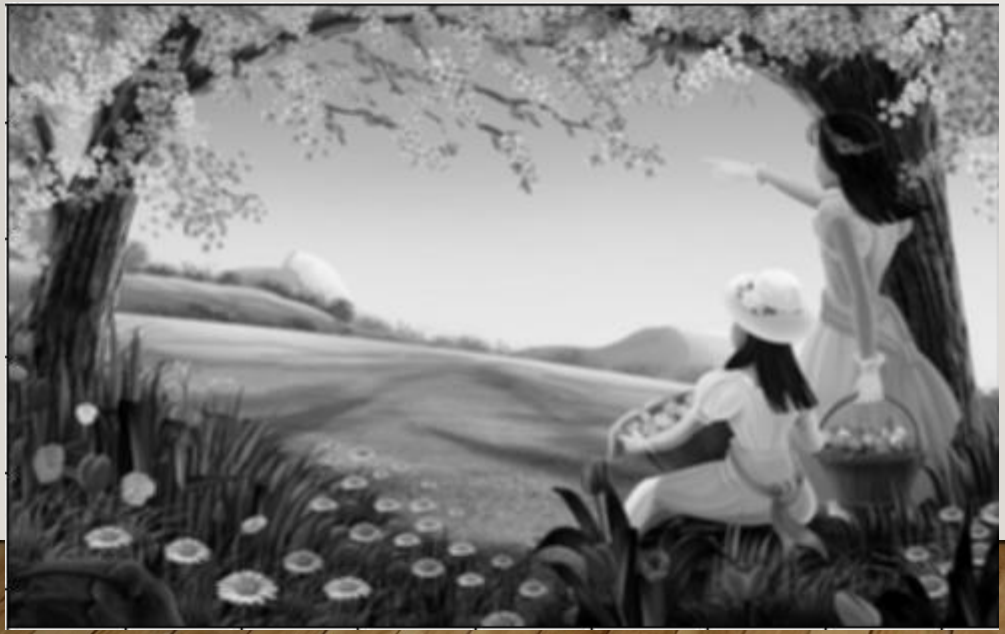
padding

sum

output



Mean 5x5



Mean 11x11