Rép 1 & 2 & 3

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

#-------------------------------------------------------------------------

path_img = 'D:\Enseignements\Maamar\TP-02\img\color-img.png'
img = cv.imread(path_img)
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow("Converted Image",gray)
cv.imwrite('D:\Enseignements\Maamar\TP-02\img\gray-img.png', gray)

cv.waitKey(0)
cv.destroyAllWindows()
#-------------------------------------------------------------------------
```

4- Créer une image avec une dimension 200 * 200 et tel que les valeurs de tous les pixels sont égaux à zéro, affiche la, et créer son histogramme.

```
#-------------------------------------------------------------------------
img = np.zeros((200,200) , np.uint8)
cv.imshow("img",img)
plt.hist(img.ravel() , 256 ,[0,255])
plt.show()
cv.waitKey(0)
cv.destroyAllWindows()
```

5- Dans le chapitre 2 du cours page 7 appliquer une transformation simple sur l'image f(x ,y)

En utilisant la formule g(x,y) = $\sqrt{f(x,y)}$ Afficher les deux images avant et après la transformation

```
#-------------------------------------------------------------------------
# Transformation simples
path_img = 'D:\Enseignements\Maamar\TP-02\img\im-02.png'
img_02 = cv.imread(path_img)
img_02 = img_02 / 255
new_gray = np.sqrt(img_02)
cv.imshow("Image Ameliorée ",new_gray)
cv.waitKey(0)
cv.destroyAllWindows()
#-------------------------------------------------------------------------
```

6- Transformer l'image f(x,y) en binaire en utilisant un seuil s = 127

```
#-------------------------------------------------------------------------------
seuil = 127
im_b = cv.threshold(gray, seuil, 255, cv.THRESH_BINARY)[1]
cv.imshow("Image Binaire  ",im_b)
cv.waitKey(0)
cv.destroyAllWindows()
#-------------------------------------------------------------------------------
```

7- Afficher l'histogramme de l'image suivante :

$$\begin{vmatrix} 0,1,1,1,0 \\ 1,3,3,2,1 \\ 1,3,2,3,1 \\ 1,2,3,3,1 \\ 0,1,1,1,0 \end{vmatrix}$$

```
# ------------------------ Creer Histogramme avec pendas -----------------------------
import pandas as pd
data = {
    'Niveau de gris': [0, 1, 2, 3],
    'Frequence Pixels': [0.16, 0.48, 0.12, 0.24]
        }

# Creation de DataFrame de data
df = pd.DataFrame(data)
# Creation de bar chart avec Matplotlib
plt.bar(df['Niveau de gris'], df['Frequence Pixels'], width=0.5, color="orange")
plt.title('Exemple')
plt.xlabel('Niveau de gris')
plt.ylabel('Frequence Pixels')
for i, value in enumerate(df['Frequence Pixels']):
    plt.text(df['Niveau de gris'][i], value, str(value), ha='center', va='bottom')

plt.grid(True)
plt.show()
```

8- Donner l'histogramme de l'image gray de la question 03 et réaliser l'égalisation de ce dernier en affichant l'histogramme égalisé, Afficher l'image après l'égalisation

```
#---------------------------------------- Egalization ----
hist = cv.calcHist([gray], [0], None, [256], [0, 256])
plt.plot(hist)
plt.savefig('img_non_eg.png')
img_non_eg = cv.imread('img_non_eg.png')
plt.show()
# ------------------ Egalisation --------------------
img_eg = cv.equalizeHist(gray)
hist = cv.calcHist([img_eg], [0], None, [256], [0, 256])
plt.plot(hist)
plt.savefig('img_eg.png')
newImg = cv.imread('img_eg.png')
plt.show()
#-------------------------------------------------------------
```

9-

```
from scipy.ndimage import gaussian_filter

def gaussian_filtre(image, sigma=1):
    if image.ndim == 2:  # si image niveau de gris
        return gaussian_filter(image, sigma=sigma)
    elif image.ndim == 3:  # si image couleur
        return np.stack([gaussian_filter(image[:, :, i], sigma=sigma) for i in range(3)], axis=2)
    else:
        raise ValueError("Image must be either 2D (grayscale) or 3D (RGB).")

image = plt.imread('D:\Enseignements\Maamar\TP-02\img\gray-img.png')
sigma_value = 2  # Adjust sigma value for more or less blur
filtered_image = gaussian_filtre(image, sigma=sigma_value)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title("Originale Image")
plt.imshow(image)
plt.axis('off')
plt.subplot(1, 2, 2)
plt.title(f"Filtre Gaussian avec (sigma={sigma_value})")
plt.imshow(filtered_image)
plt.axis('off')
plt.show()
```

```python
def convolve2D(image, kernel, padding=0, strides=1):
    kernel = np.flipud(np.fliplr(kernel))
    xKernShape = kernel.shape[0]
    yKernShape = kernel.shape[1]
    xImgShape = image.shape[0]
    yImgShape = image.shape[1]

    xOutput = int(((xImgShape - xKernShape + 2 * padding) / strides) + 1)
    yOutput = int(((yImgShape - yKernShape + 2 * padding) / strides) + 1)
    output = np.zeros((xOutput, yOutput))

    if padding != 0:
        imagePadded = np.zeros((image.shape[0] + padding*2, image.shape[1] + padding*2))
        imagePadded[int(padding):int(-1 * padding), int(padding):int(-1 * padding)] = image
        print(imagePadded)
    else:
        imagePadded = image
    for y in range(image.shape[1]):
        if y > image.shape[1] - yKernShape:
            break
        if y % strides == 0:
            for x in range(image.shape[0]):
                if x > image.shape[0] - xKernShape:
                    break
                try:
                    if x % strides == 0:
                        output[x, y] = (kernel * imagePadded[x: x + xKernShape, y: y + yKernShape]).sum()
                except:
                    break
    return output
```

```python
def processImage(image):
  image = cv.imread(image)
  image = cv.cvtColor(src=image, code=cv.COLOR_BGR2GRAY)
  return image

from PIL import Image as im
image = processImage('D:\Enseignements\Maamar\TP-02\img\conv.png')
kernel = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])
output = convolve2D(image, kernel, padding=2)
cv.imwrite('2DConvolved.jpg', output)
image = im.fromarray(output)
image.show(image)
```