FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES

CARRERA: COMPUTACIÓN **ASIGNATURA**: APRENDIZAJE DE MÁQUINA

NRo. PRÁCTICA:1 **TÍTULO PRÁCTICA**: Procesamiento de Imágenes con Numpy

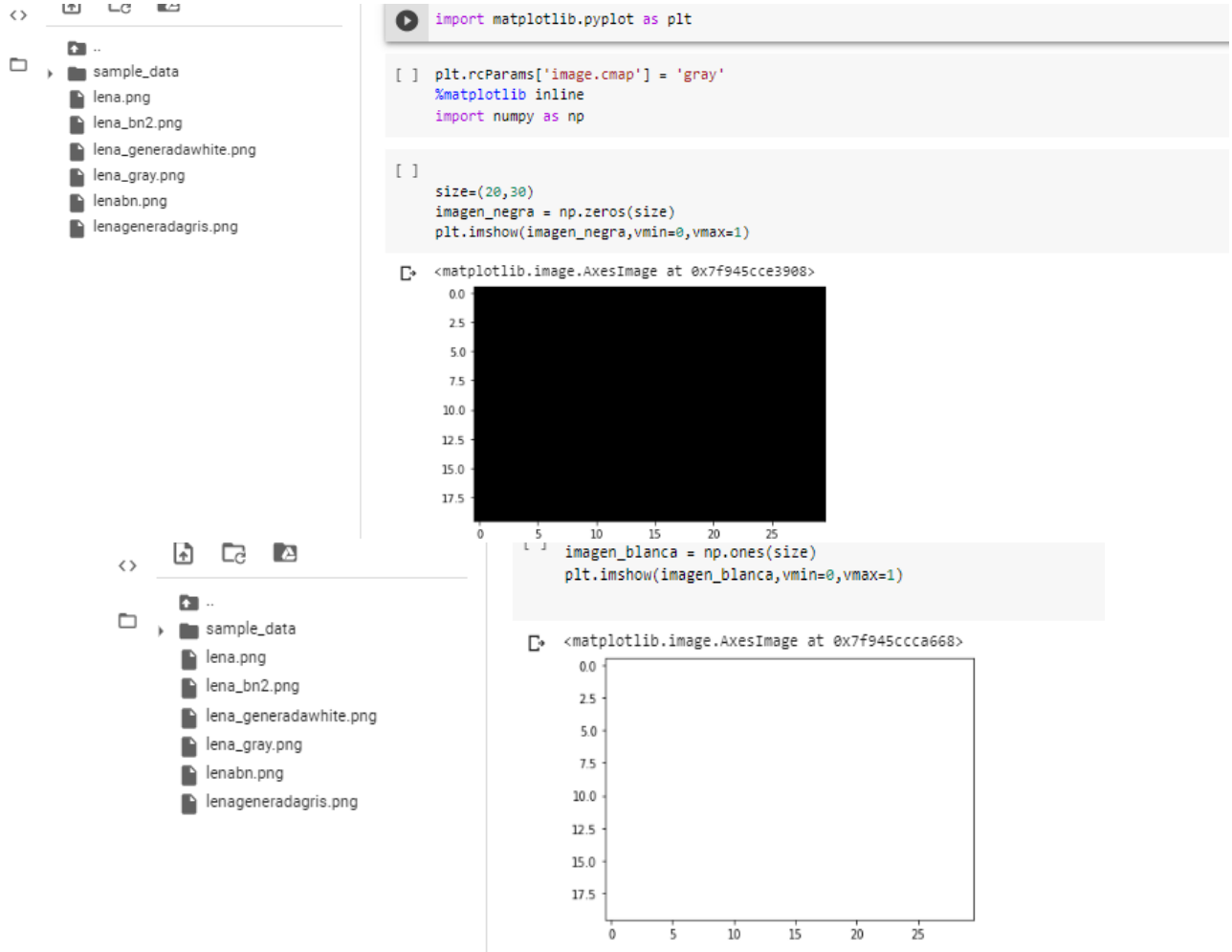
OBJETIVOS:

- Desarrollar las actividades con lo explicado en clases y utilizar las respectivas librerías.
- Aprender a modificar, crear, etc. Las respectivas imágenes con lo solicitado.

ACTIVIDADES POR DESARROLLAR (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

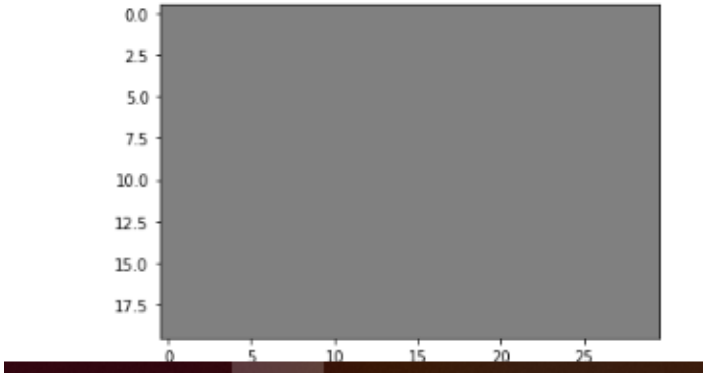
1. Ingresar a Google colab
2. Subir el archivo GPL1
3. Agregar las respectivas imágenes LENA y LENA_GRAY para que el programa no falle
4. Realizar las respectivas actividades indicadas en el PDF
5. Realizar histogramas
6. Capturar los resultados obtenidos

RESULTADO(S) OBTENIDO(S):



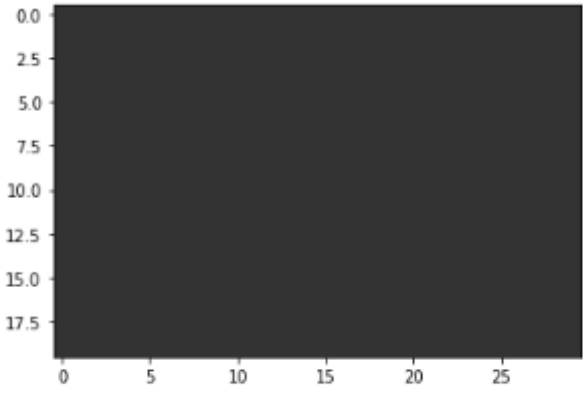
```
[ ]  plt.figure()
     imagen_gris = np.ones(size)*0.5
     plt.imshow(imagen_gris,vmin=0,vmax=1)
```

```
<matplotlib.image.AxesImage at 0x7f945cc305f8>
```



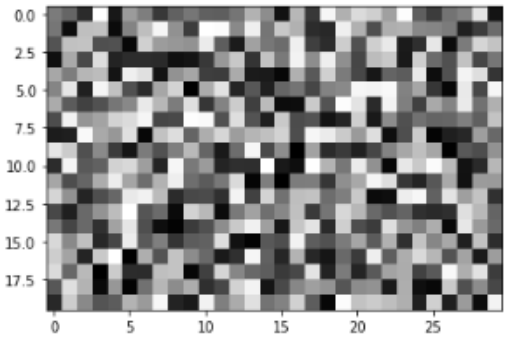

```
[ ]  imagen_gris_oscuro = np.ones(size)*0.2
     plt.imshow(imagen_gris_oscuro,vmin=0,vmax=1)
```

```
<matplotlib.image.AxesImage at 0x7f945cc18320>
```



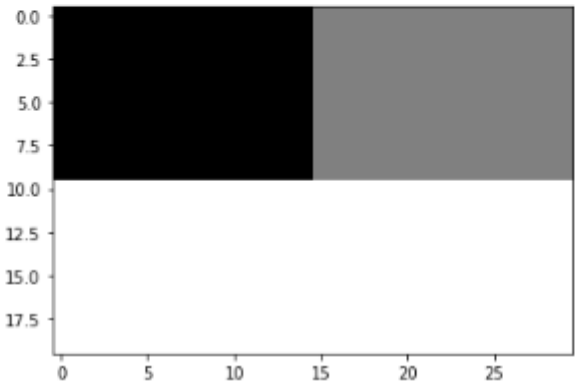

```
[ ]  plt.figure()
     imagen_aleatoria = np.random.rand(size[0],size[1])
     plt.imshow(imagen_aleatoria,vmin=0,vmax=1)
```

```
<matplotlib.image.AxesImage at 0x7f945cb7e320>
```



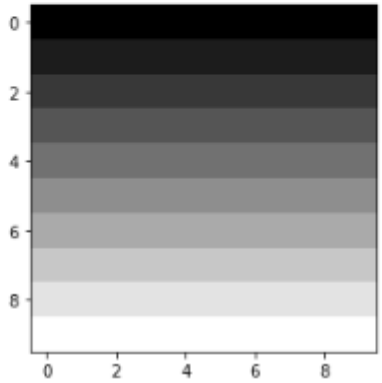
```
[ ]  A = np.zeros((10,15))
     B = np.ones((10,30))*0.5
     C = np.ones((10,30))

     B[:,:-15] = A

     D = np.concatenate((B ,C), axis = 0)
     plt.imshow(D, vmin = 0, vmax= 1)
```

```
<matplotlib.image.AxesImage at 0x7f94592226d8>
```



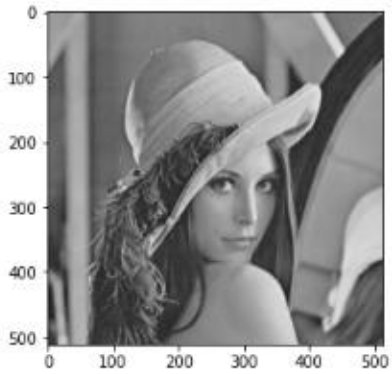

```
[ ]  A = np.ones((10,10))
     B = np.linspace(0,1,10)
     C = B*A
     #IMPLEMENTAR
     plt.imshow(C.transpose())
```

```
<matplotlib.image.AxesImage at 0x7f945cac3f60>
```



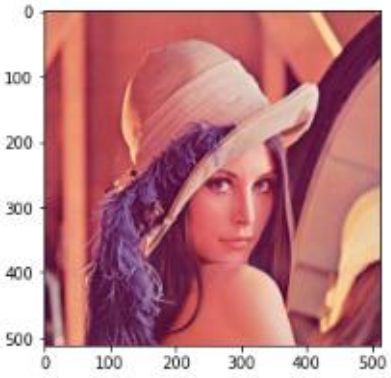
```
[ ] from skimage import io
    image=io.imread("lena_gray.png")/255.0
    print("Dimension de la imagen")
    print(image.shape)
    plt.imshow(image,vmin=0,vmax=1)
```

```
Dimension de la imagen
(512, 512)
<matplotlib.image.AxesImage at 0x7f945ca261d0>
```



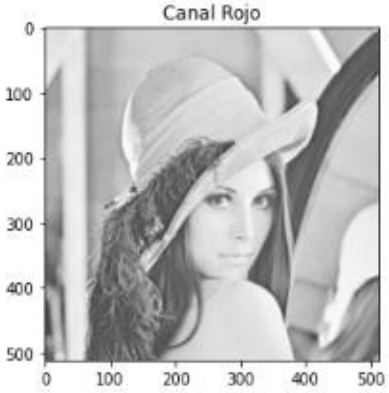

```
[ ] lenargb=io.imread("lena.png")/255.0
    plt.imshow(lenargb)
    print("Dimension de la imagen ")
    print(lenargb.shape)
```

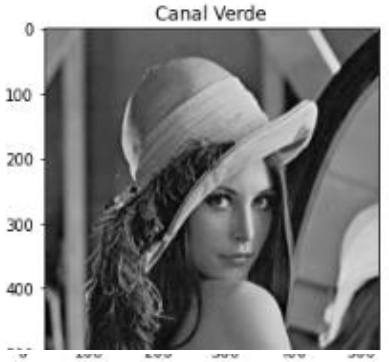
```
Dimension de la imagen
(512, 512, 3)
```

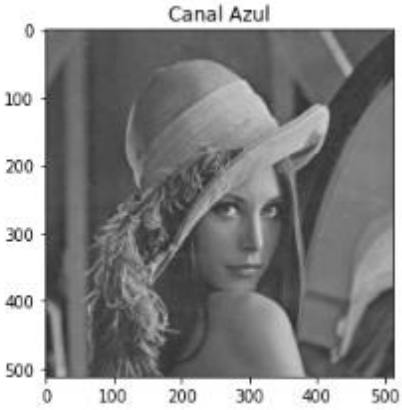


```
[ ]  plt.imshow(lenargb[:,:,0],vmin=0,vmax=1)
     plt.title("Canal Rojo")
     plt.figure()
     plt.imshow(lenargb[:,:,1],vmin=0,vmax=1)
     plt.title("Canal Verde")
     plt.figure()
     plt.imshow(lenargb[:,:,2],vmin=0,vmax=1)
     plt.title("Canal Azul")
```

```
Text(0.5, 1.0, 'Canal Azul')
```

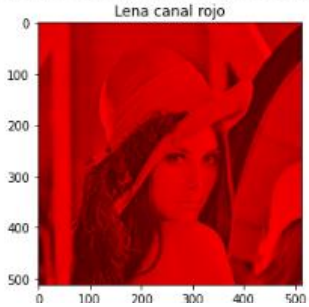






```
[ ] lenared=np.copy(lenargb)
    lenared[:,:,1]=0
    lenared[:,:,2]=0
    plt.title("Lena canal rojo")
    plt.imshow(lenared)
```

```
<matplotlib.image.AxesImage at 0x7f945c884cc0>
```

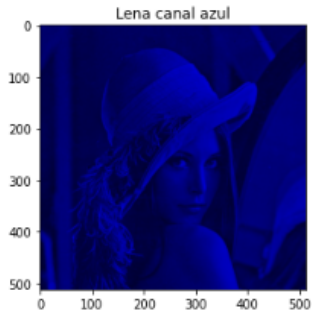


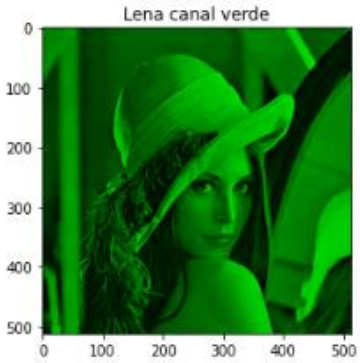

```
[ ] lenaredgreen=np.copy(lenargb)
    lenaredgreen[:,:,2]=0
    plt.title("Lena sin canal azul")
    plt.imshow(lenaredgreen)
```

```
<matplotlib.image.AxesImage at 0x7f945c85f208>
```

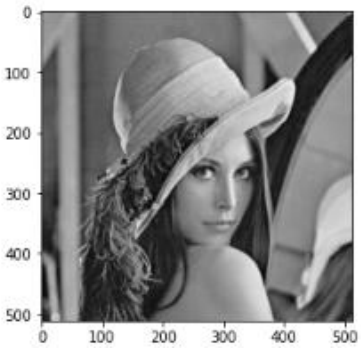

```
[ ] lena_blue=np.copy(lenargb)
    lena_blue[:,:,0]=0
    lena_blue[:,:,1]=0
    plt.title("Lena canal azul")
    plt.imshow(lena_blue)
    plt.figure()
    lena_green=np.copy(lenargb)
    lena_green[:,:,0]=0
    lena_green[:,:,2]=0
    plt.title("Lena canal verde")
    plt.imshow(lena_green)
    plt.figure()
    suma_blue_green = lena_blue+lena_green
    plt.imshow(suma_blue_green)
    plt.title("Lena canal Azul y Verde")
    plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

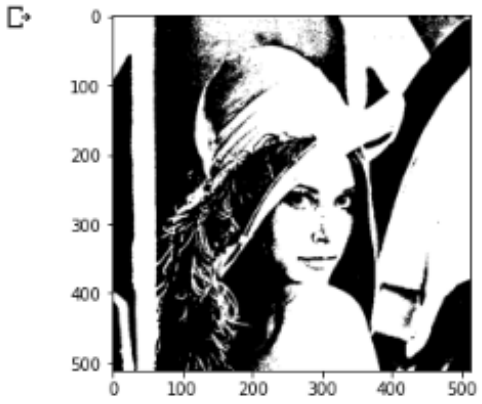


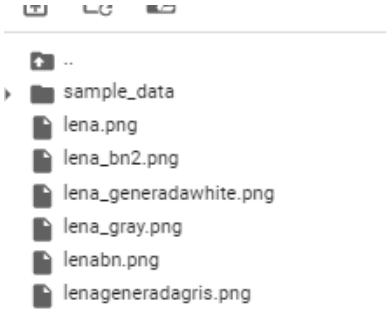



```
[ ]  #Ejercicio: Convertir la imagen de lena color a escala de grises
     from skimage import img_as_ubyte
     lena_gris = (lenargb[:,:,0]+lenargb[:,:,1]+lenargb[:,:,2])/3
     plt.imshow(lena_gris)
     io.imsave("lenageneradagris.png",img_as_ubyte(lena_gris))
```



```
[ ]  from skimage import io
     from skimage import img_as_ubyte
     h,w=lena_gris.shape # obtenemos el tamaño de la imagen original
     lena_white_black=np.zeros((h,w)) # creamos una matriz donde generar la imagen
     pM=np.mean(lena_gris)
     for i in range(h):
      for j in range(w):
        if(lena_gris[i,j]>pM):
          lena_white_black[i,j]=1
        else:
            lena_white_black[i,j]=0

      # guardar ese promedio en el pixel i,j de la imagen generada
     plt.imshow(lena_white_black)
     io.imsave("lena_generadawhite.png",img_as_ubyte(lena_white_black))
```

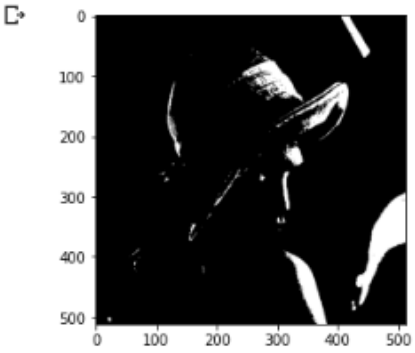





```
[ ]  #Convertir la imagen original en blanco y negro
     from PIL import Image
     image_file = Image.open("lena.png") # open colour image
     image_file = image_file.convert('1') # convert image to black and white
     image_file.save('lenabn.png')
```

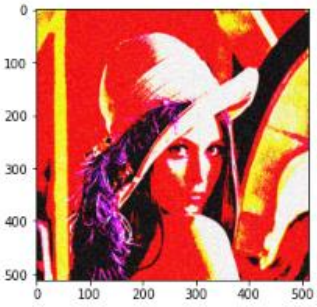
```
[ ]  from PIL import Image
     img = Image.open('lena.png')
     thresh = 200
     fn = lambda x : 255 if x > thresh else 0
     r = img.convert('L').point(fn, mode='1')
     r.save('lena_bn2.png')
```

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

     #Pixels higher than this will be 1. Otherwise 0.
     THRESHOLD_VALUE = 200

     #Load image and convert to greyscale
     img = Image.open("lena.png")
     img = img.convert("L")

     imgData = np.asarray(img)
     thresholdedData = (imgData > THRESHOLD_VALUE) * 1.0

     plt.imshow(thresholdedData)
     plt.show()
```



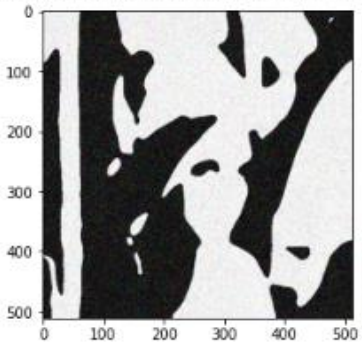

```
[ ]  from scipy import ndimage
     f = 10
     c = 256
     lena_img = np.copy(lenargb)
     points = c*np.random.random((2,f**2))
     lena_img[(points[0]).astype(np.int),(points[1].astype(np.int))] = 1
     lena_img = ndimage.gaussian_filter(lena_img, sigma=1/(4.*f))
     m = (lena_img > lena_img.mean()).astype(np.float)
     m += 0.1 * lena_img
     lena_histogram = m + 0.2*np.random.randn(*m.shape)
     """
     h, bedges = np.histogram(lena_img, bins=60)
     bcenter = 0.5*(bedges[:-1]+bedges[1:])
     bin_lena = lena_histogram > 0.5
     """
     plt.imshow(lena_histogram)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7f945c06b208>
```



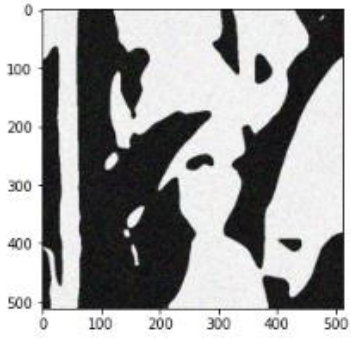
```
[ ] from scipy import ndimage
    f = 10
    c = 256
    lena_img = np.copy(lenargb)
    points = c*np.random.random((2,f**2))
    lena_img[(points[0]).astype(np.int),(points[1].astype(np.int))] = 1
    lena_img = ndimage.gaussian_filter(lena_img, sigma=c/(4.*f))
    m = (lena_img > lena_img.mean()).astype(np.float)
    m += 0.1 * lena_img
    lena_histogram = m + 0.2*np.random.randn(*m.shape)
    h, bedges = np.histogram(lena_img, bins=60)
    bcenter = 0.5*(bedges[:-1]+bedges[1:])
    bin_lena = lena_histogram > 0.5
    plt.imshow(lena_histogram)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7f945bfc4da0>
```



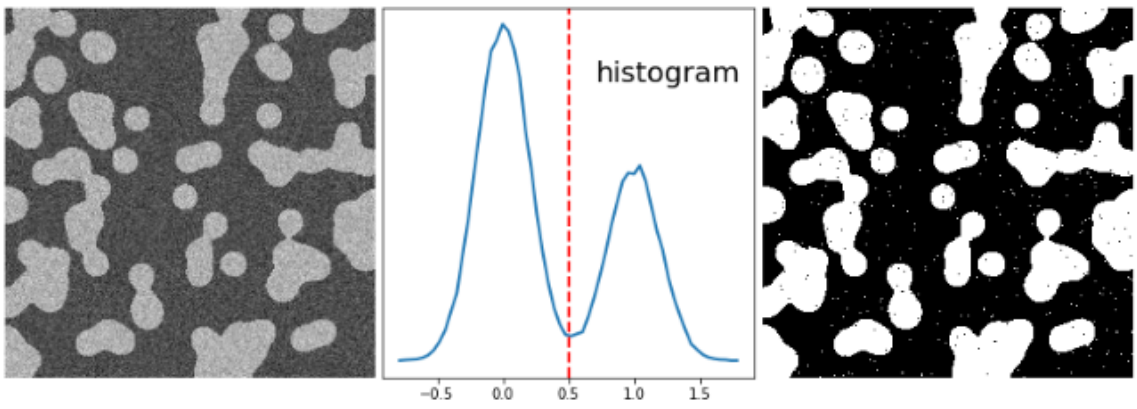

```
[ ] from scipy import ndimage
    f = 10 #valor n
    c = 256 #valor rgb
    lena_o = np.copy(lenargb) #se toma la foto de lena.png
    points = c*np.random.random((2, f**2))
    lena_o[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
    lena_o = ndimage.gaussian_filter(lena_o, sigma=c/(4.*f)) #metodo gaussiano
    mask = (lena_o > lena_o.mean()).astype(np.float)
    mask += 0.1 * lena_o
    histogram = mask + 0.2*np.random.randn(*mask.shape)
    plt.imshow(histogram)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7f945bf170b8>
```

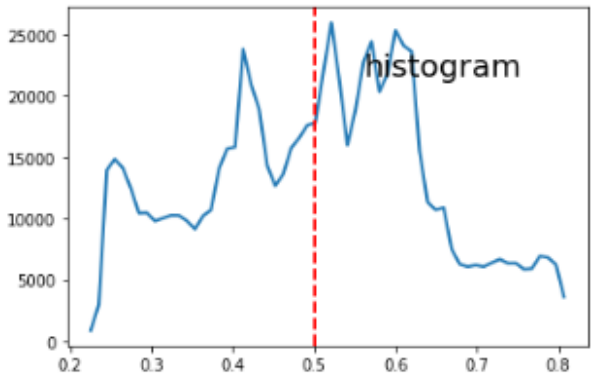


```
]  from scipy import ndimage
   import matplotlib.pyplot as plt

   np.random.seed(1)
   n = 10
   l = 256
   im = np.zeros((l, l))
   points = l*np.random.random((2, n**2))
   im[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
   im = ndimage.gaussian_filter(im, sigma=l/(4.*n))

   mask = (im > im.mean()).astype(np.float)

   mask += 0.1 * im

   img = mask + 0.2*np.random.randn(*mask.shape)

   hist, bin_edges = np.histogram(img, bins=60)
   bin_centers = 0.5*(bin_edges[:-1] + bin_edges[1:])

   binary_img = img > 0.5

   plt.figure(figsize=(11,4))

   plt.subplot(131)
   plt.imshow(img)
   plt.axis('off')
   plt.subplot(132)
   plt.plot(bin_centers, hist, lw=2)
   plt.axvline(0.5, color='r', ls='--', lw=2)
   plt.text(0.57, 0.8, 'histogram', fontsize=20, transform = plt.gca().transAxes)
   plt.yticks([])
   plt.subplot(133)
   plt.imshow(binary_img, cmap=plt.cm.gray, interpolation='nearest')
   plt.axis('off')
```

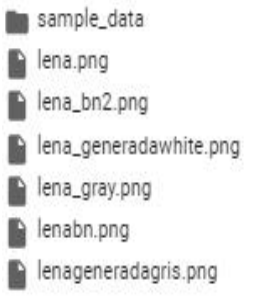
```
plt.subplots_adjust(wspace=0.02, hspace=0.3, top=1, bottom=0.1, left=0, right=1)
plt.show()
```



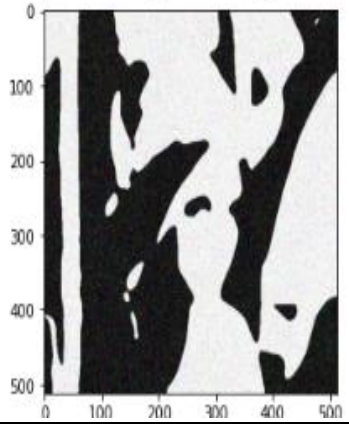
```
[ ] from scipy import ndimage
    import matplotlib.pyplot as plt
    f = 10 #valor n
    c = 256 #valor rgb
    lena_img = np.copy(lenargb) #se toma la foto de lena.png
    points = c*np.random.random((2,f**2))
    lena_img[(points[0]).astype(np.int),(points[1].astype(np.int))] = 1
    lena_img = ndimage.gaussian_filter(lena_img, sigma=c/(4.*f)) #metodo gaussiano
    m = (lena_img > lena_img.mean()).astype(np.float)
    m += 0.1 * lena_img
    lena_histogram = m + 0.2*np.random.randn(*m.shape)
    h, bedges = np.histogram(lena_img, bins=60)
    bcenter = 0.5*(bedges[:-1]+bedges[1:])
    bin_lena = lena_histogram > 0.5
    plt.plot(bcenter, h, lw=2)
    plt.axvline(0.5, color='r', ls='--', lw=2) #linea punteada del histograma
    plt.text(0.57, 0.8, 'histogram', fontsize=20, transform = plt.gca().transAxes) #histogrma
    plt.show()
    plt.imshow(lena_histogram)
```




```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
<matplotlib.image.AxesImage at 0x7f945c6100f0>
```




```
Clipping input data to the valid range for imshow with RGB data (
<matplotlib.image.AxesImage at 0x7f945c6100f0>
```



CONCLUSIONES:

- Esta práctica nos llenó de más conocimientos en ver cómo puede variar una imagen a través de su escala de colores desde blanco a amarillo, también en cómo realizar una segmentación basada en un histograma y aparte los links que están dentro del PDF nos ayudó en comprender mucho mejor y poder realizar este trabajo.

RECOMENDACIONES

- Seguir dándonos talleres para poder obtener un buen aprendizaje y que a su vez siga enviando mas links de libros, artículos ya que estos nos ayudan en tener un mayor aprendizaje y entendimiento, aunque también depende de uno como estudiante querer superarse.
- Siga tomando preguntas a través de kahoot y otras aplicaciones ya que nos refuerza la clase y así se sabe si comprendimos o no.

***Nombre de estudiante*:** Ángel Cevallos – Andrea Rigchac

***Firma de estudiante*:** Ángel Cevallos – Andrea Rigchac