```
#%% TP 12

#%% exo 3

import numpy as np
import matplotlib.pyplot as plt

def nuancier(n,p):
    M = np.zeros((n,p))
    for i in range(n):
        for j in range(p):
            M[i,j] = (i+j)/(n+p-2)
    return M
plt.imshow(nuancier(10,5),cmap='gray')
plt.show()

#%% exo 4

# Si besoin pour savoir quel est le répertoire courant :
# import os
# print(os.getcwd())

import matplotlib.image as mpimg

mon_image_RGB = mpimg.imread(r'C:\Users\fahel\Downloads\panda.jpg') # importer l'image
mon_image = mon_image_RGB[:,:,0] # pour avoir un simple tableau n x p
plt.imshow(mon_image, cmap='gray') # dessiner l'image
plt.show() # afficher l'image

panda = mon_image
#nombre de pixels par lignes
A=np.size(panda,0) #vaut 590
#nombre de pixels par colonne
B=np.size(panda,1) #vaut 800
#nombre de pixels au total
C=np.size(panda) #vaut 590 x 800 = 472000

#%% exo 5

def pixel_neg(p):
    return 1-p

def negatif(im):
    n = np.size(im,0)
    p = np.size(im,1)
    im_res = np.zeros((n,p))
    for i in range(n):
        for j in range(p):
            im_res[i,j] = pixel_neg(im[i,j])
    return im_res

plt.imshow(negatif(panda),cmap='gray')
plt.show()


#%% exo 6


def miroir(image):
    nb_lignes = np.size(image,0)
    nb_cols = np.size(image,1)
```

```
    res = np.zeros((nb_lignes,nb_cols))
    for i in range(nb_lignes):
        for j in range(nb_cols):
            res[i,j] = image[i,nb_cols-j-1]
    return res


plt.imshow(miroir(panda), cmap='gray')
plt.show()

##

def rotation90gauche(image):
    nb_lignes = np.size(image,0)
    nb_cols = np.size(image,1)
    image_rotation = np.zeros((nb_cols,nb_lignes))
    for i in range(nb_cols):
        for j in range(nb_lignes):
            image_rotation[i,j] = image[j,nb_cols-i-1]
    return image_rotation

plt.imshow(rotation90gauche(panda), cmap='gray')
plt.show()


#%% exo 7


N = 1000
dessin=np.zeros((N,N))

def abs_ord(k):
    return (k-N/2)*(4/N)

def disque(xa,ya,r,dessin):
  for i in range(N):
      x = abs_ord(i)
      for j in range(N):
        y = abs_ord(j)
        if (x-xa)**2 + (y-ya)**2 < r**2:
          dessin[i,j]=1-dessin[i,j]
  return dessin

disque(0,0,1,dessin)

plt.imshow(dessin, cmap='gray')
plt.show()
#%%

image = np.zeros((N,N))
sinus = np.sqrt(3)/2

disque(0,0,1,image)
disque(0,1,1,image)
disque(0,-1,1,image)
disque(sinus,-.5,1,image)
disque(sinus,.5,1,image)
disque(-sinus,.5,1,image)
disque(-sinus,-.5,1,image)

plt.imshow(image, cmap='gray')
```

```
plt.show()
```