```
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 25 14:43:45 2022

@author: dconduche

Programme 1er semestre de PCSI :
Les images servent de support à la présentation de manipulatic
de tableaux à deux dimensions.
 * Algorithmes de rotation,
 * de réduction ou d'agrandissement.
TP:Modification d'une image par convolution : flou, détection
"""

import matplotlib.pyplot as plt
import numpy as np
plt.set_cmap('gray')


def rotation(a):
    """Tourne une image de +90° (+π/2) dans le sens trigo
    a -- tableau numpy
    retourne une image b tournée de +π/2
    """
    n, p = a.shape
    b = np.empty((p, n), dtype=type(a[0, 0]))
    for i in range(n):
        for j in range(p):
            b[p-j-1, i] = a[i, j]
    return b

"""Explication :
Image  « a » de départ (indices (i, j)) : (n,p) = (3,2)
(0,0)   (0,1)
(1,0)   (1,1)
(2,0)   (2,1)
Image « b » d'arrivée :
(0,1)  (1,1)  (2,1)
(0,0)  (1,0)  (2,0)

=> Le pixel a[i, j] va dans b[p-j-1, i]

ligne 25 : on récupère la taille du tableau contenant l'image.
 n lignes, p colonnes (comme A ∈ Mnp(R))

ligne 26 : taille : on tourne l'image, donc p lignes et n colc
```

```
Plusieurs type de tableau peuvent exister pour coder une image
(unint8 : 0..255; float64: 0...1.). Il est donc nécessaire de
le type d'un élément de a, pour fixer le type des données dans
(donné par « dtype »).

ligne 27-28 : on parcourt les lignes (i) et colonnes (j) de a.

ligne 29 : Le pixel a[i, j] se retrouve en position [p-1-j, i]
 [cf schéma au debut du paragraphe]
"""



def baisseReso(a, N):
    """baisse la résolution
    a -- tableau numpy
    N -- coté du carré de pixel.
    retourne une image b de résolution //N**2
    """
    n, p = a.shape
    b = a[:n//N, :p//N]  # Astuce pour créer b, avec le même d
    for i in range(n//N):
        for j in range(p//N):
            b[i, j] = a[i*N:(i+1)*N, j*N:(j+1)*N].sum()/N**2
    return b

"""Explication :
ligne 66 : on récupère la taille du tableau contenant l'image.
 n lignes, p colonnes (comme A ∈ Mnp(R))

ligne 67 : taille : on divise le nombre de pixels par N**2. Do
sera de taille (n//N, p//N).
Pourquoi partir de a :  En créant b à partir de a,
 on est certain d'avoir le même dtype dans le tableau numpy.

ligne 68-69 : on parcourt les lignes (i) et colonnes (j) de b.

ligne 70 : dans le pixel b[i,j], on met la moyenne des pixel d
 de taille (N,N) dans a, en partant du pixel a[i*N, j*N] en ha
 [Faire un dessin ! Testez pour i, j, N petits]

Remarque : Si n et p (taille de l'image) ne sont pas des multi
on ignore ce qui dépasse.
Ainsi, pour diviser par 2 la taille d'une image de taille n=7
et p=5 colonnes, python fait comme si l'image était n=6 et p=4
lignes et colonnes ne sont pas lues.
```

```
"""


a = plt.imread('js.jpg')  # votre image noir et blanc 2 dimens
N = 10
print('baisse de résoltuion avec N de', N)
b = baisseReso(a, N)
plt.imshow(b, interpolation='none')
plt.show()


b = rotation(a)
plt.imshow(b)
plt.show()


a = np.random.rand(20, 20)  # Tableau aléatoire :
# intensité lumineuse entre 0 (noir) et 1 (blanc, intensité mo
a[a > .5] = 1   # On prend les indices tels que a > .5, et on
a[a <= .5] = 0  # Idem pour les indices tels que a <= .5
plt.imshow(a, interpolation='none')  # interpolation : pour ur
  # sinon, plt floute pour adoucir les bords (testez en enlevo
plt.show()
plt.imshow(baisseReso(a, 4))
plt.show()
plt.imshow(rotation(a))
plt.show()
```