

# Correction\_Cours\_Images

December 14, 2019

## 0.1 Exercice 2 : Images bitmap au format PBM

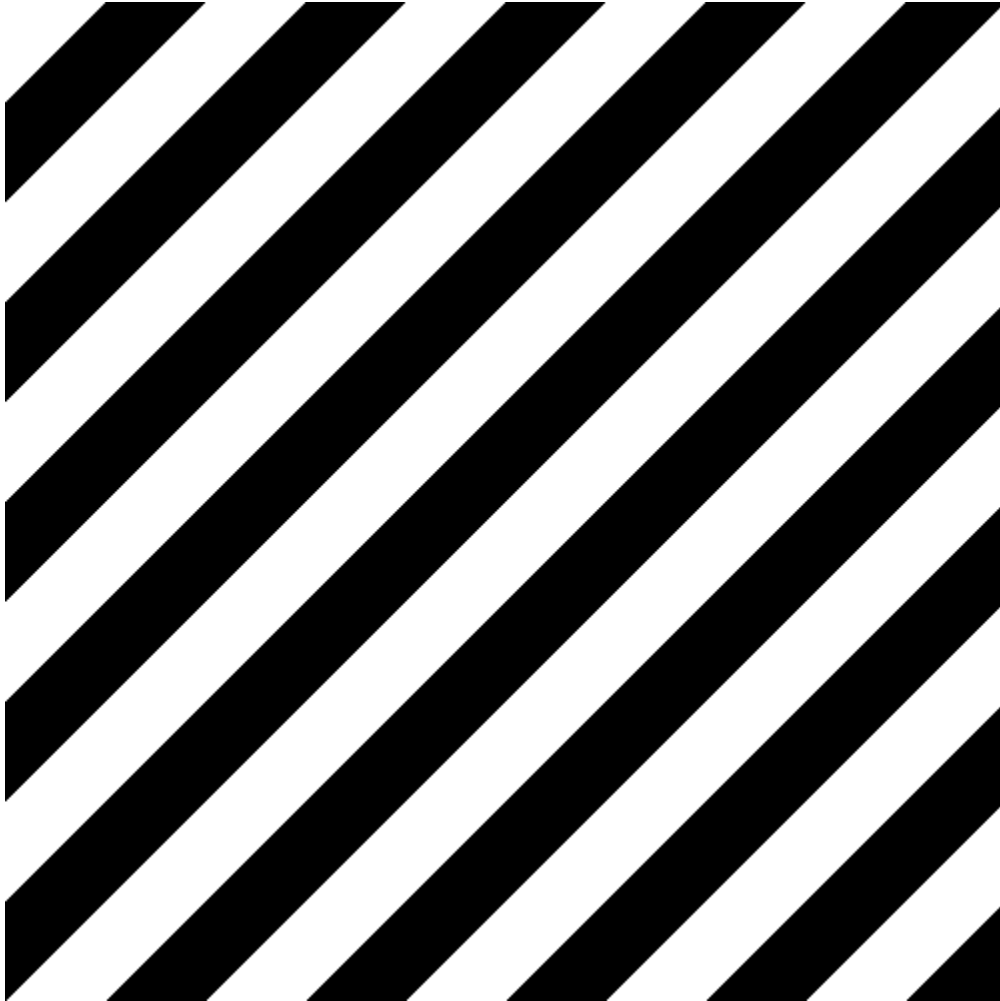
### 0.1.1 Une première solution

```
In [25]: L, H = 500, 500
         f = open('bandes.pbm', 'w')
         f.write('P1\n')
         f.write(str(L) + ' ' + str(H) + '\n')
         for lig in range(H):
             for col in range(L):
                 if (lig // 50 + col // 50)%2 == 0 and lig % 50 + col % 50 <= 49 \
                     or (lig // 50 + col // 50)%2 == 1 and lig % 50 + col % 50 > 49 :
                     f.write('0') #blanc
                 else:
                     f.write('1') #noir
             f.write('\n')
         f.close()
```

### 0.1.2 Une solution plus compacte

```
In [26]: L, H = 500, 500
         f = open('bandes.pbm', 'w')
         f.write('P1\n')
         f.write(str(L) + ' ' + str(H) + '\n')
         for lig in range(H):
             for col in range(L):
                 if ((lig + col) // 50)%2 == 1:
                     f.write('1') #noir
                 else:
                     f.write('0') #blanc
             f.write('\n')
         f.close()
```

```
In [27]: #affichage avec wand module de Python de binding pour ImageMagick
         def conversion(imagesource,imagebut, formatbut):
             import wand
             from wand.image import Image
             from wand.display import display
```



```
#on convertit l'image inversée en PNG avec wand
impbm = wand.image.Image(filename = imagesource)
impng = impbm.convert(formatbut)
impng.save(filename = imagebut)

pbm2png('bandes.pbm', 'bandes-pbm.png')
```

## 0.2 Exercice 3 : Images bitmap au format PGM

```
In [28]: def inverser_couleurs_pbm(source, but):
"""Lit le fichier pbm source et le recopie dans le fichier pgm
but en inversant les couleurs"""
#ouverture de fichiers en lecture pour source et écriture pour but
f = open(source, 'r')
g = open(but, 'w')
#on recopie l'en-tête (les deux premières lignes)
```

```

for k in range(2):
    lig = f.readline()
    g.write(lig)
#pour les lignes codant les pixels on inverse chaque valeur de pixel
for lig in f:
    for caractere in lig:
        if caractere == '0':
            g.write('1')
        elif caractere == '1':
            g.write('0')
        else:
            g.write(caractere)
g.close()
f.close()

```

In [29]: `inverser_couleurs_pbm('damier.pbm', 'damier-inverse.pbm')`

In [30]: `conversion('damier-inverse.pbm', 'damier-inverse.png', 'png')`

### 0.3 Exercice 3 Images en niveaux de gris au format PGM

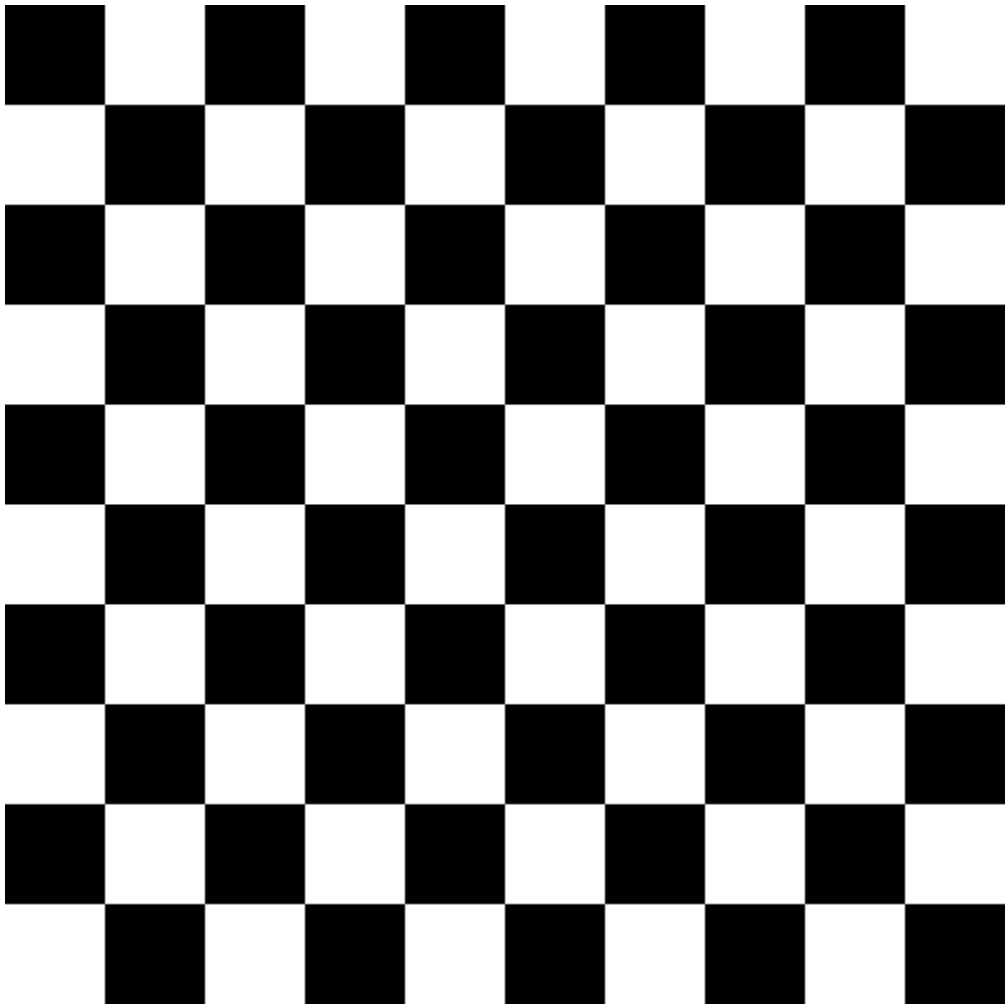
```

In [31]: def rechercher_pixel_ppm(source, lig, col):
    """Retourne la valeur du pixel en ligne lig et colonne col
    dans le fichier image source au format PGM"""
    f = open(source, 'r')
    for k in range(3 + lig + 1):
        f.readline()
    liste_pixels = f.readline().split()
    f.close()
    return int(liste_pixels[col])

def flip_ppm(source, but):
    """Retourne un fichier but au format PGM obtenu en appliquant une symétrie
    d'axe vertical (flip) à l'image du fichier source au format PGM"""
    #ouverture de fichiers en lecture pour source et écriture pour but
    f = open(source, 'r')
    g = open(but, 'w')
    #on recopie l'en-tête (les trois premières lignes)
    for k in range(3):
        lig = f.readline()
        g.write(lig)
    for lig in f:
        liste_pixels = lig.split()
        g.write(' '.join(liste_pixels[::-1]) + '\n')
    g.close()
    f.close()

def flop_ppm(source, but):

```



```

"""Retourne un fichier but au format PGM obtenu en appliquant une symétrie
d'axe horizontal (flop) à l'image du fichier source au format PGM"""
#ouverture de fichiers en lecture pour source et écriture pour but
f = open(source, 'r')
g = open(but, 'w')
#on recopie l'en-tête (les trois premières lignes)
for k in range(3):
    lig = f.readline()
    g.write(lig)
liste_lignes = f.readlines()
for lig in reversed(liste_lignes):
    g.write(lig)
g.close()
f.close()

```

```

In [32]: flip_ppm('lena.ppm', 'lena-flip.ppm')
         flop_ppm('lena.ppm', 'lena-flop.ppm')

```

```

In [33]: conversion('lena.ppm', 'lena.png', 'png')
         conversion('lena-flip.ppm', 'lena-flip.png', 'png')
         conversion('lena-flop.ppm', 'lena-flop.png', 'png')

```

### 0.3.1 lena.png

### 0.3.2 lena-flip.png

### 0.3.3 lena-flop.png

## 0.4 Exercice 3 Fichiers binaires et images bitmap

```

In [34]: ## Imports des modules

```

```

from PIL import Image
import numpy

```

```

##Exercice 3 Question 1

```

```

def transformation(i, j, n):
    """Retourne les coordonnées (colonne, ligne) de l'image du pixel de
coordonnées (i, j) par la transformation du photomaton"""
    if i % 2 == 0:
        if j % 2 == 0:
            return (i // 2, j // 2)
        else:
            return (i // 2, (j + n) // 2)
    else:
        if j % 2 == 0:
            return ((i + n) // 2, j // 2)
        else:

```









```
return ((i + n) // 2, (j + n) // 2)
```

```
##Exercice 3 Question 2
```

```
def copie(tableau):  
    """Retourne la copie profonde d'un tableau à deux dimensions"""  
    return numpy.array([[tableau[j][i] for i in range(len(tableau[j]))]  
        for j in range(len(tableau))], dtype = 'uint8')  
  
def photomaton(tableau):  
    """Retourne le tableau calculé obtenu après application de la transformation  
    du photomaton au tableau passé en paramètre"""  
    tableau2 = copie(tableau)  
    n = len(tableau)  
    for i in range(n):  
        for j in range(n):  
            (i2, j2) = transformation(i, j, n)  
            tableau2[i2][j2] = tableau[i][j]  
    return tableau2
```

```
##Exercice 3 Question 3
```

```
def photomaton_iterer(source, k):  
    im = Image.open(source)  
    nom, extension = source.split('.')  
    tab = numpy.asarray(im)  
    for iteration in range(1, k + 1):  
        tab = photomaton(tab)  
        im = Image.fromarray(tab)  
        im.save(nom + '-photomaton-iteration-' + str(iteration) + '.' + extension)
```

```
In [35]: photomaton_iterer('lenagray-256.png', 8)
```

0.4.1 Itération 1

0.4.2 Itération 2

0.4.3 Itération 3

0.4.4 Itération 4

0.4.5 Itération 5

0.4.6 Itération 6

0.4.7 Itération 7

0.4.8 Itération 8



