Informatique PCSI
Corrigé TP 7 suite

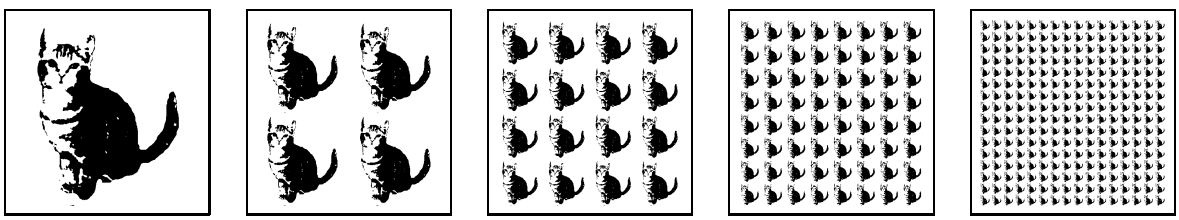
Exercice 1

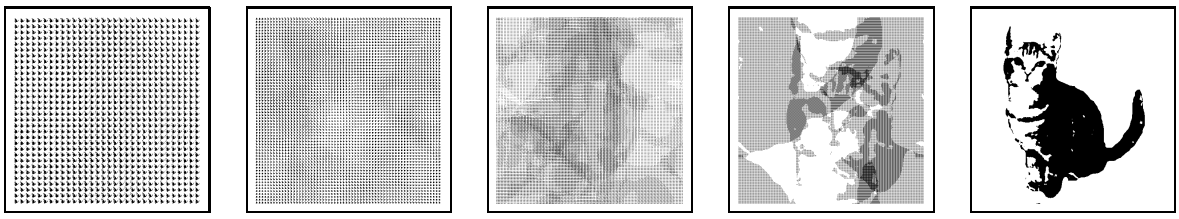
```
def echange_lignes(img): # matrice carrée
    n = len(img) # n pair
    for k in range(1, n//2):
        for i in range(k, n-1-k, 2):
            img[i], img[i+1] = img[i+1], img[i]

m = lire_fichier_pbm('chat.pbm')
for i in range(1, 10):
    echange_lignes(m)
    ecrire_fichier_pbm("echange" + str(i) + ".pbm", m)
```

Exercice 2

```
def photomaton(img, k):
    n = len(img)
    mat1 = [[img[i][j] for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            if i % 2 == 0 and j % 2 == 0:
                img[i//2][j//2] = mat1[i][j]
            elif i % 2 == 1 and j % 2 == 0:
                img[(n+i)//2][j//2] = mat1[i][j]
            elif i % 2 == 0 and j % 2 == 1:
                img[i//2][(n+j)//2] = mat1[i][j]
            else:
                img[(n+i)//2][(n+j)//2] = mat1[i][j]
    ecrire_fichier_pbm("photomaton"+str(k)+".pbm", img)
    if k < 9:
        photomaton(img, k+1)

m = lire_fichier_pbm('chat.pbm')
photomaton(m, 1)
```

Avec une image de 512 par 512 pixels, on retrouve la même image après neuf exécutions.





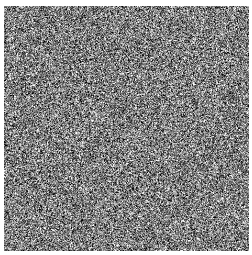
Exercice 3

1. La fonction `hasard`.

On importe la fonction `randint` du module `random`. L'expression `randint(0, 1)` prend la valeur aléatoire 0 ou 1.

```
def hasard(n, p):
    from random import randint
    m = [[0 for j in range(p)] for i in range(n)]
    for i in range(n):
        for j in range(p):
            m[i][j] = randint(0, 1)
    return m

a = hasard(512, 512) # dimensions de l'image chat.pbm
ecrire_fichier_pbm("alea.pbm", a)
```

On obtient une image aléatoire comme celle-ci :



2. La fonction de chiffrement.

On définit une matrice `m` avec les bonnes dimensions remplies de 0. Ensuite on utilise l'opérateur *ou exclusif* entre chaque nombre de la matrice `img` et le nombre correspondant de la matrice `cle` pour obtenir la valeur de `m[i][j]`.

```
def chiffre(img, cle):
    n = len(img)
    p = len(img[0])
    m = [[0 for j in range(p)] for i in range(n)]
    for i in range(n):
        for j in range(p):
            m[i][j] = img[i][j] ^ cle[i][j]
    return m

m = lire_fichier_pbm('chat.pbm')
c = chiffre(m, a)
ecrire_fichier_pbm("image_chiffrée.pbm", c)
```

L'image chiffrée obtenue ressemble à une image aléatoire.

Il s'agit d'un chiffrement symétrique. Pour déchiffrer le résultat, on utilise la clé de chiffrement avec le même programme et on écrit simplement :

```
d = chiffre(c, a)
ecrire_fichier_pbm("image_déchiffrée.pbm", d)
```

On obtient une image identique à l'image initiale. On peut vérifier que le contenu des deux fichiers `image1.pbm` et `image_déchiffrée.pbm` est exactement le même.

3. On reprend la fonction `chiffre` qui sert aussi à déchiffrer et on remplace l'opérateur *ou exclusif* par l'opérateur *ou*.

```
def devoile(img, cle):
    n = len(img)
    p = len(img[0])
    m = [[0 for j in range(p)] for i in range(n)]
    for i in range(n):
        for j in range(p):
            m[i][j] = img[i][j] | cle[i][j]
    return m

dv = devoile(c, a)
ecrire_fichier_pbm("image_dévoilée.pbm", dv)
```

On obtient une image qu'on dit « dévoilée ». Elle laisse apparaître l'image déchiffrée.