Traitement d'images

Avant de commencer : uploader (téléverser) les 3 fichiers ci-dessous, qui doivent être présents dans le même dossier que ce notebook :

- voirP123tableau.py
- sioNoirEtBlanc.png
- image_degrade2.png

Puis **exécuter la cellule ci-dessous**

In [1]:
```
from voirP123tableau import *  #programme qui permet de voir
```

Tableaux (listes de listes)

Observer les sorties ci-dessous.

In [2]:
```
T = [[1, 2, 3], [4, 5, 6]]
L1 = T[0]
print(L1)
L2 = T[1]
print(L2)
print(T[0][2])
T[1][2] = 60
print(T)
L1[2] = 30
print(T)
print(L1)
```

```
[1, 2, 3]
[4, 5, 6]
3
[[1, 2, 3], [4, 5, 60]]
[[1, 2, 30], [4, 5, 60]]
[1, 2, 30]
```

Copier-coller le code ci-dessus dans la fenêtre de http://pythontutor.com/visualize.html#mode=display (http://pythontutor.com/visualize.html#mode=display) et exécuter pas à pas.

Nous allons manipuler des images pbm, pgm, ppm en les mettant sous forme de tableaux.

Images noir et blanc

L'image au format PBM sioNoirEtBlanc a été créée avec Notepad++ :

```
P1
14 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 1 0 1 1 1 1 1 0
0 1 0 0 0 0 1 0 1 0 0 0 1 0
0 1 0 0 0 0 1 0 1 0 0 0 1 0
0 1 1 1 1 0 1 0 1 0 0 0 1 0
0 0 0 0 1 0 1 0 1 0 0 0 1 0
0 0 0 0 1 0 1 0 1 0 0 0 1 0
0 1 1 1 1 0 1 0 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

NB : 14 car l'image comporte 14 colonnes ; 9 car elle comporte 9 lignes.

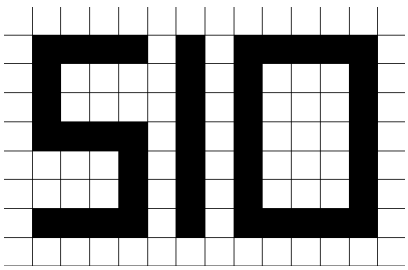
$14 \times 9 = 126$ donc c'est une image de 126 pixels.

Stockons cette image dans un tableau à deux dimensions :

In [3]:
```
imageP1 = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
           [0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0],
           [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0],
           [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0],
           [0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
           [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
           [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
           [0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0],
           [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

print(type(imageP1))
hauteur = len(imageP1)     # la hauteur est le nombre de lign
largeur = len(imageP1[0])  #P1[0] est la première ligne
print(largeur, hauteur)
```

```
<class 'list'>
14 9
```

In [4]:
```
voir_image('P1',imageP1)
```



Pour passer à la suite, fermer la fenêtre de visualisation.

Image en niveaux de gris

Voici une image au format PGM, à 10 nuances de gris :

```
P2
15 7
10
0   0   0  0   0  0  0  0  0  0  0  0  0  0  0
0  10  10  10  0  7  7  7  0  5  0  2  2  2  0
0   0   0  10  0  7  0  7  0  5  0  2  0  2  0
0  10  10  10  0  7  0  7  0  5  0  2  2  2  0
0  10   0   0  0  7  0  7  0  5  0  0  0  2  0
0  10  10  10  0  7  7  7  0  5  0  2  2  2  0
0   0   0  0   0  0  0  0  0  0  0  0  0  0  0
```

In [5]:
```
imageP2 = [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],[0,10,10,10,0,7,7,
voir_image('P2',imageP2,10)
```

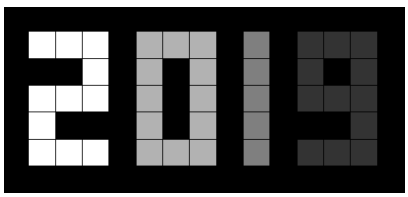


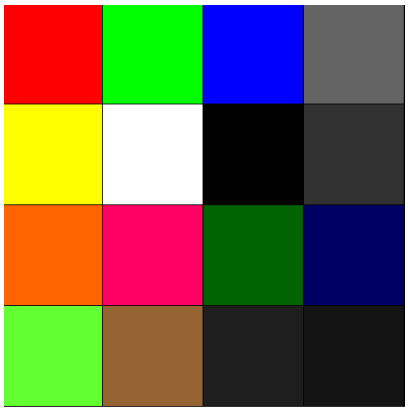
Image RVB

Voici une image au format PPM, de 4 pixels sur 4 (256 nuances).

```
P3
4 4
255
255  0   0     0  255  0      0   0  255    100 100 1
00
255 255  0    255 255 255     0   0   0      50  50
50
255  100 0    255  0  100     0  100  0      0    0 1
00
100 255  50   150  100 50     30  30  30     20  20
20
```

Pour chaque pixel, on a 3 nombres : la dose de Rouge, la dose de Vert, la dose de Bleu.

In [6]:
```
imageP3 = [[255,0,0,0,255,0,0,0,255,100,100,100],[255,255,0,2

voir_image('P3',imageP3,255)
```



Exercice 1 : inverser le noir et le blanc

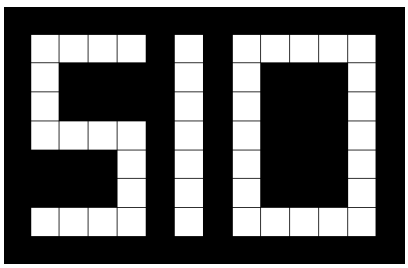
Nous allons modifier l'image noir et blanc.

Ecrire une fonction **inverseNoirEtBlanc(t)** qui prend en paramètre une image t (une liste de listes) et retourne une image où les pixels noirs sont devenus blanc et inversement.

In [7]:
```
image = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0],
         [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0],
         [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0],
         [0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
         [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
         [0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],
         [0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

def inverseNoirEtBlanc(t):
    i = 0
    while i < len(t):
        j = 0
        while j < len(t[0]):
            if t[i][j] == 0:
                t[i][j] = 1
            else:
                t[i][j] = 0
            j = j+1
        i = i+1
    return t


print(inverseNoirEtBlanc(image))
```

```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 0, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 1], [1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
1, 1, 0, 1], [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1], [1,
0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1], [1, 1, 1, 1, 0, 1,
0, 1, 0, 1, 1, 1, 0, 1], [1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
1, 0, 1], [1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1], [1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

In [8]:
```
voir_image('P1',inverseNoirEtBlanc(image))
```



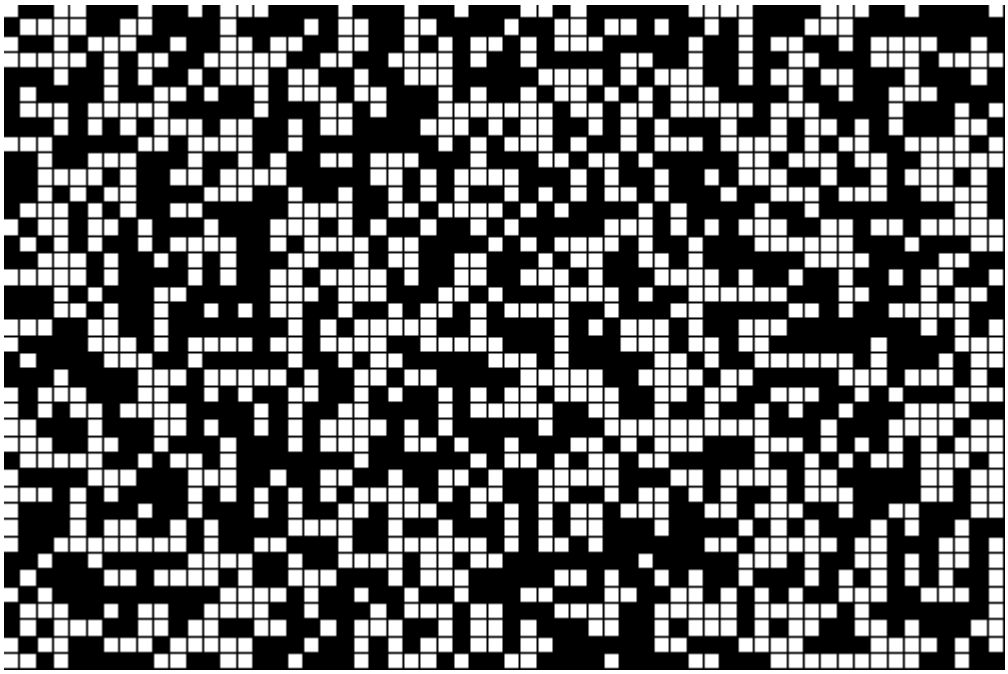
Images aléatoires

randint(0,1) fournit un entier aléatoire : 0 ou 1 (comme si on jouait à pile ou face avec une pièce équilibrée).

Compléter le programme ci-dessous pour qu'il affiche une image aléatoire de 60 pixels sur 40.

In [9]:
```
from random import randint

largeur = 60
hauteur = 40

image_hasard = [[0]*largeur for k in range(hauteur)] # créati

i = 0

while i < len(image_hasard):
    j = 0
    while j < len(image_hasard[0]):
        image_hasard[i][j] = randint(0,1)
        j = j+1
    i = i+1

print(image_hasard)
voir_image('P1',image_hasard)
```

```
[[1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
1], [0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0], [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0], [0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,
1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 0, 0, 1], [0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1], [1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
1, 0, 0, 1, 1, 0], [1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
```



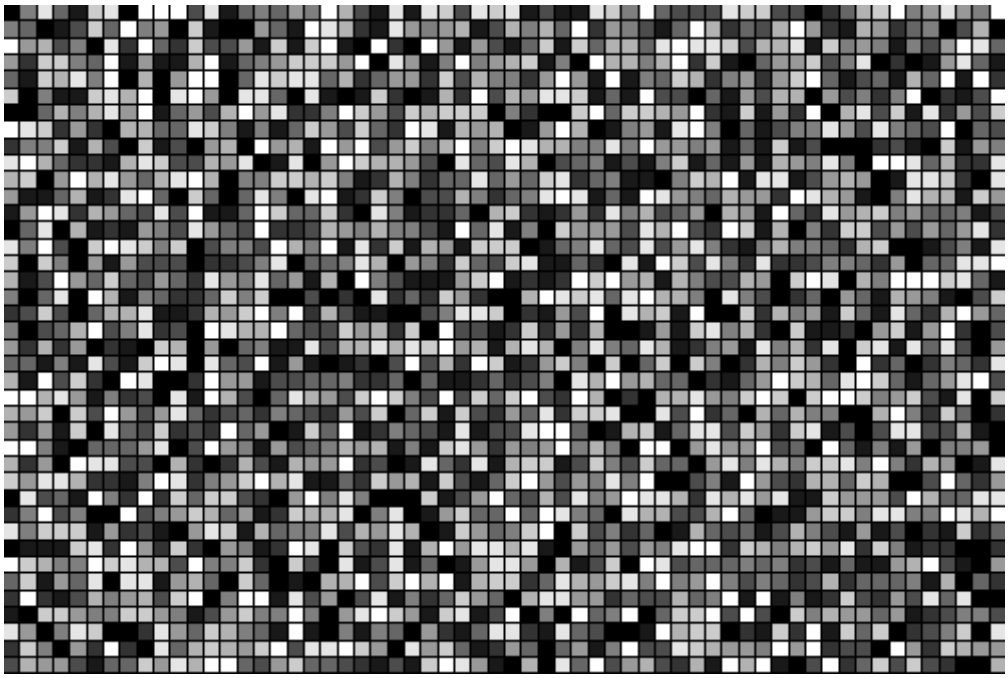
Exercice 2

En s'inspirant du programme ci-dessus, créer une image aléatoire en niveaux de gris.

Si on choisit 10 niveaux de gris, randint(0,10) donnera un entier aléatoire entre 0 et 10 (0 et 10 compris).

In [10]:
```
from random import randint

largeur = 60
hauteur = 40

image_hasard = [[0]*largeur for k in range(hauteur)] # créati

for i in range(len(image_hasard)):
    for j in range(len(image_hasard[0])):
        image_hasard[i][j] = randint(0,10)

print(image_hasard)
voir_image('P2',image_hasard,10)
```

```
[[1, 4, 2, 7, 2, 10, 2, 1, 8, 6, 5, 4, 0, 9, 6, 4, 10, 2, 8,
10, 7, 2, 1, 0, 2, 8, 4, 0, 10, 2, 5, 6, 5, 8, 0, 9, 6, 1,
1, 7, 3, 8, 7, 6, 4, 5, 6, 7, 5, 1, 9, 8, 6, 7, 7, 7, 5, 5,
10, 0], [4, 2, 1, 2, 0, 1, 5, 10, 2, 4, 8, 0, 9, 10, 7, 8,
7, 8, 2, 9, 6, 0, 0, 7, 5, 1, 9, 5, 9, 2, 10, 5, 1, 3, 8, 7,
2, 6, 4, 5, 9, 1, 1, 8, 2, 0, 4, 9, 4, 4, 9, 2, 3, 10, 1, 7,
6, 9, 2, 6], [10, 7, 0, 2, 0, 7, 1, 10, 8, 6, 0, 5, 6, 2, 8,
3, 9, 9, 6, 2, 4, 9, 4, 10, 0, 8, 4, 1, 3, 8, 10, 6, 5, 4,
9, 3, 9, 10, 1, 5, 5, 3, 8, 2, 9, 2, 6, 6, 0, 3, 4, 4, 4, 6,
6, 4, 8, 6, 7, 8], [5, 3, 6, 9, 9, 8, 1, 1, 0, 6, 5, 0, 2,
4, 10, 5, 0, 7, 5, 7, 7, 1, 6, 4, 0, 2, 4, 10, 8, 2, 3, 7,
7, 5, 6, 5, 8, 9, 8, 3, 5, 2, 2, 2, 6, 2, 2, 4, 3, 0, 1, 9,
10, 2, 8, 2, 8, 9, 8, 1], [3, 2, 2, 6, 4, 9, 4, 2, 5, 3, 3,
6, 2, 9, 9, 2, 2, 6, 2, 8, 5, 5, 10, 2, 4, 10, 5, 1, 0, 9,
6, 8, 4, 5, 7, 2, 4, 4, 8, 7, 7, 5, 1, 7, 4, 6, 4, 10, 3, 1
0, 8, 4, 0, 7, 4, 1, 7, 6, 8, 0], [3, 0, 9, 10, 10, 3, 7, 7,
9, 3, 6, 7, 1, 9, 0, 7, 10, 8, 0, 4, 3, 1, 0, 5, 4, 7, 3, 3,
0, 4, 6, 8, 5, 0, 3, 4, 3, 3, 3, 3, 8, 8, 8, 8, 6, 9, 4, 2,
5, 4, 9, 3, 6, 6, 0, 7, 6, 5, 5, 6], [7, 10, 4, 6, 5, 2, 0,
```



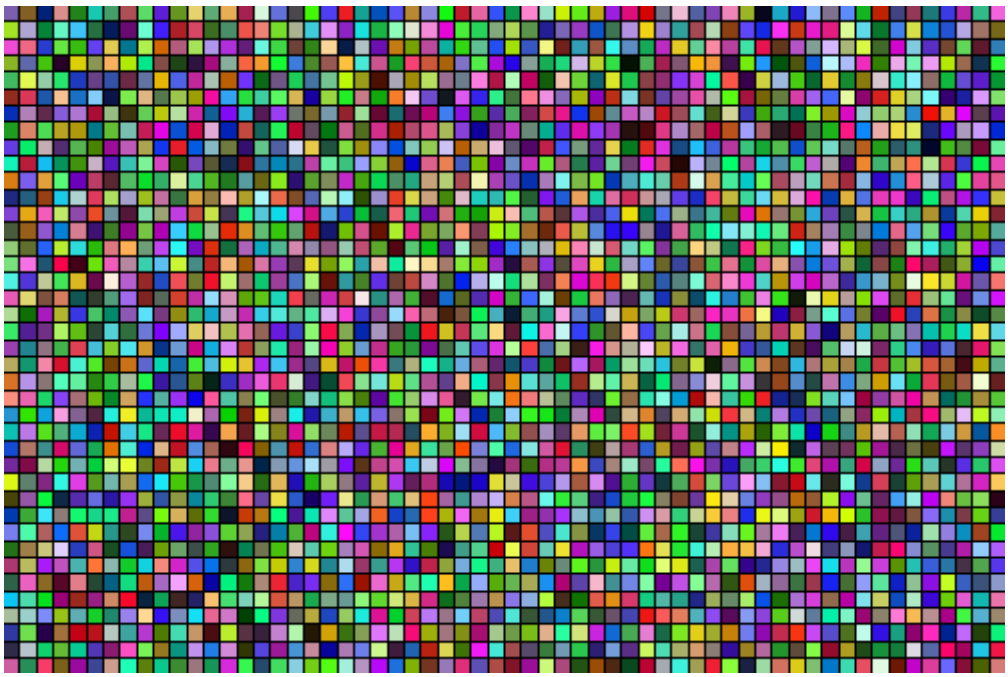
Exercice 3

Créer une image aléatoire RVB :

In [11]:
```
from random import randint

largeur = 60
hauteur = 40

image_hasard = [[0]*3*largeur for k in range(hauteur)] # créa

for i in range(len(image_hasard)):
    for j in range(len(image_hasard[0])):
        image_hasard[i][j] = randint(0,255)

voir_image('P3',image_hasard,255)
```

In [12]:
```
len(image_hasard[0])
```

Out[12]: 180



Exercice 4

Observer l'image :

Ecrire le programme qui affiche cette image (20 pixels sur 2) :

In [13]:
```
largeur = 20
hauteur = 2

image = [[0]*largeur for k in range(hauteur)] # création d'un

# creation de la 1ere ligne
for j in range(len(image[0])):
    image[0][j] = 20-j
# creation de la 2ere ligne
for j in range(len(image[0])):
    image[1][j] = j
print(image)
voir_image('P2',image,20)
```

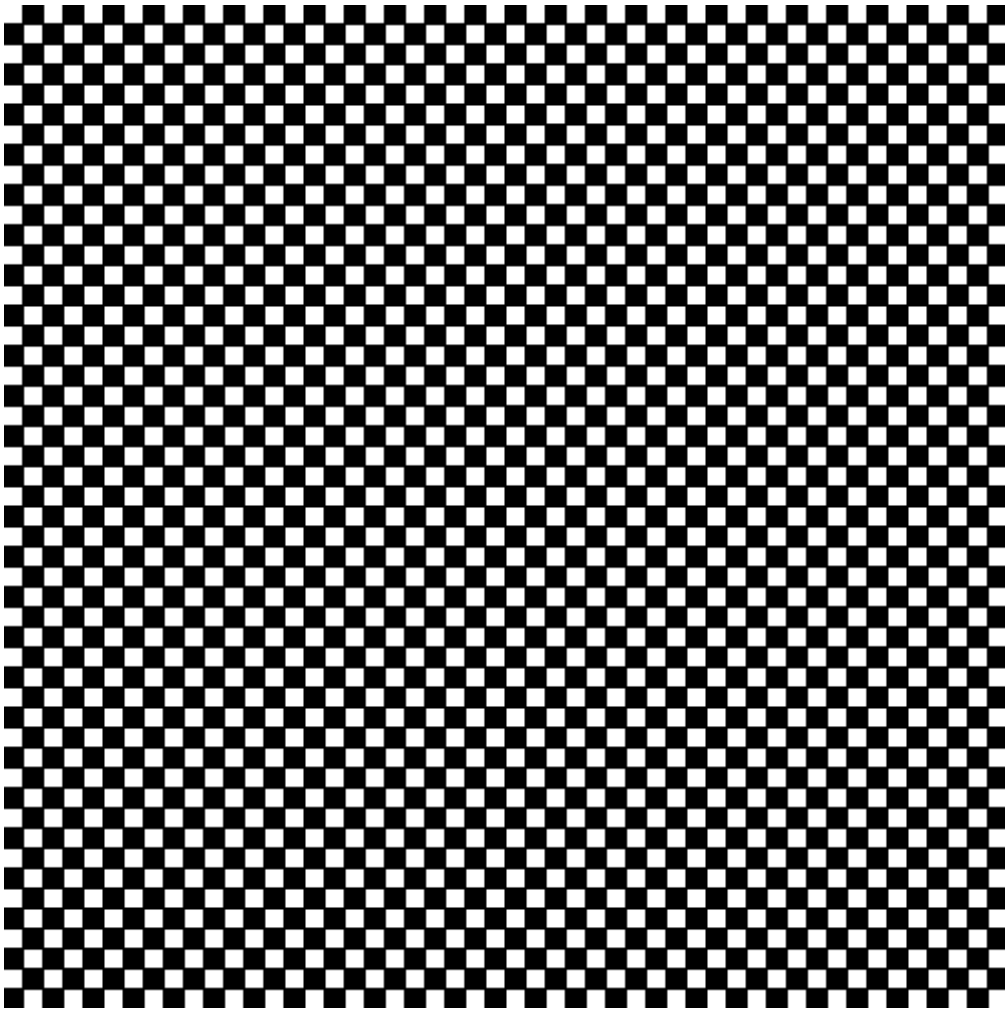
```
[[20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5,
4, 3, 2, 1], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19]]
```

Exercice 5

Ecrire une fonction **damier(n)** qui affiche un damier de n cases sur n.

Pour n=8, on verra un plateau de jeu d'échecs.

In [14]:
```
def damier(n):
    jeu = [[0]*n for k in range(n)] # creation d'un damier bl
    for i in range(n):
        for j in range(n):
            jeu[i][j] = (i+j)%2
    return jeu

voir_image('P1',damier(50))
```



In [ ]: