







Volé de là: <https://snt.ababsurdo.fr/la-photographie-numerique/pixelart/>

## SNT - PHOTOGRAPHIE NUMÉRIQUE - ACTIVITÉ

# PIXEL ART

La bibliothèque Pillow permet de manipuler, pixel par pixel, des images en Python. Chaque image est considérée comme un tableau, chaque case contenant un triplet de nombres (R, G, B).

	0	1	2
0			
1			

Par exemple, l'image ci-dessous représente une image de 3×2 pixels. Deux choses sont à noter sur les coordonnées :

- les coordonnées commencent à 0 (comme dans les ascenseurs, ce qui est courant en informatique) ;
- l'axe vertical est gradué de haut en bas (alors qu'habituellement en mathématiques, il est gradué de bas en haut).

Ainsi, la couleur du pixel de coordonnées (2, 0) (en haut à droite) est (236, 25, 32) (rouge).

Le programme Python suivant permet de dessiner le drapeau français de l'exemple ci-dessus.

```
from PIL import Image

image = Image.new('RGB', (3, 2))

image.putpixel((0, 0), (5, 20, 64))
image.putpixel((0, 1), (5, 20, 64))
image.putpixel((1, 0), (255, 255, 255))
image.putpixel((1, 1), (255, 255, 255))
image.putpixel((2, 0), (236, 25, 32))
image.putpixel((2, 1), (236, 25, 32))

image.save("image.png")
image.show()
```

Voici l'explication de chaque ligne :

- Chargement de la bibliothèque *pillow*, qui permet à Python de manipuler des images.

```
from PIL import Image
```

- Création d'une nouvelle image, de 3 pixels de large par 2 pixel de haut.

```
image = Image.new('RGB', (3, 2))
```

- Tracé d'un pixel de couleur (5, 20, 64) aux coordonnées (0, 1).

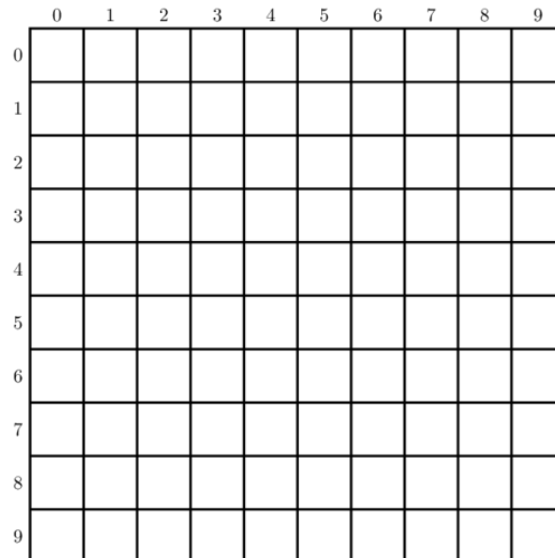
```
image.putpixel((0, 1), (5, 20, 64))
```

- Écriture de l'image dans le fichier *image.png*, puis affichage de l'image.

```
image.save("image.png")
image.show()
```

## 1. Pixel par pixel

1. Reproduisez le quadrillage suivant sur une feuille, et faites un dessin (une lettre, un smiley, etc.) en coloriant certaines cases. Votre dessin devra être composé d'au moins trois couleurs (en plus du blanc).



2. Copiez le code du drapeau de la france plus haut, et enregistrez-le dans un fichier appelé *dessin-NOM.py* (en remplaçant *NOM* par votre nom de famille).
3. Ouvrez ce fichier avec le logiciel Thonny, puis exécutez-le.

Si vous obtenez une erreur *ModuleNotFoundError: No module named < PIL >* : installez la bibliothèque Pillow.

4. Ouvrez le dossier qui contient votre fichier *.py* : un fichier *image.png* doit avoir été créé. Ouvrez cette image pour observer le drapeau français.
5. Modifiez le programme pour reproduire votre dessin, pixel par pixel. Remarques :
  1. Visitez ce site web pour déterminer le code RGB des différentes couleurs que vous souhaitez utiliser.  
→ <https://mdn.github.io/css-examples/tools/color-picker/>
  2. Par défaut, toute votre image sera noire, sauf les pixels que vous dessinez. Si vous voulez une image sur fond blanc, recopiez ces lignes juste avant de dessiner vos pixels :

```
for x in range(image.size[0]):
    for y in range(image.size[1]):
        image.putpixel((x, y), (255, 255, 255))
```

6. Écrivez bien, en commentaire de votre programme, le nom des deux membres du binôme.
7. Rendez le fichier *.py* sur Pronote.

## 2. Structures de contrôle

Plutôt que de définir les pixels un à un, il est possible d'utiliser des boucles. Voici quelques exemples. Quelques bandes

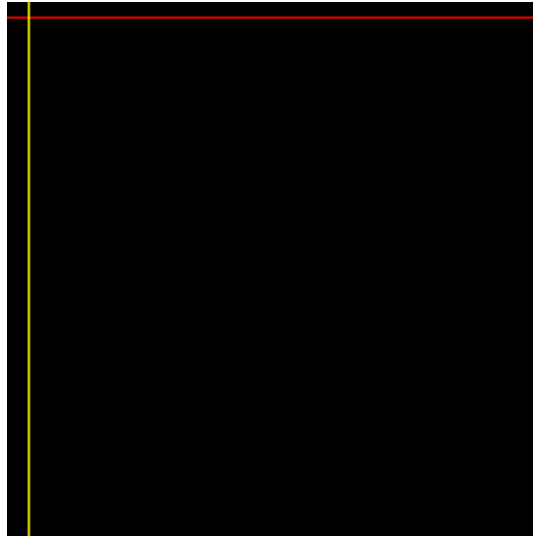
```
from PIL import Image

# On définit une image noire de 256 pixels par 256 pixels
image = Image.new("RGB", (256, 256))
```

```
# Pour chaque valeur de x, colorier le pixel de coordonnées (x, 7)
# Cela colorie toute la ligne de rang 7.
for x in range(256):
    image.putpixel((x, 7), (255, 0, 0))

# Pour chaque valeur de y, colorier le pixel de coordonnées (10, y)
# Cela colorie toute la ligne de rang 10.
for y in range(256):
    image.putpixel((10, y), (255, 255, 0))

image.save("image.png")
```



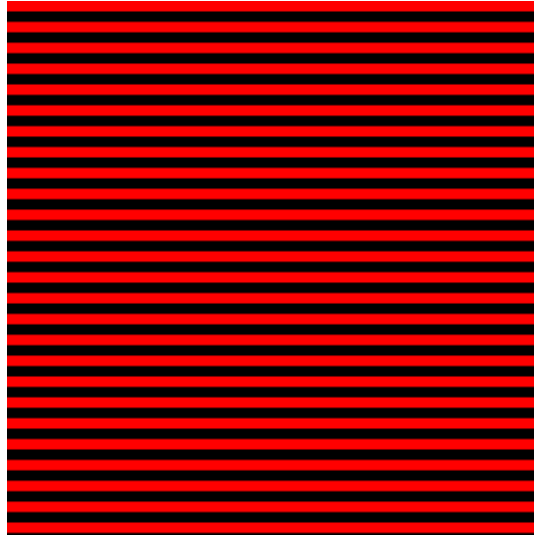
## Encore plus de bandes

```
from PIL import Image

image = Image.new("RGB", (256, 256))

for y in range(26):
    for x in range(256):
        for h in range(5):
            image.putpixel((x, 10*y+h), (255, 0, 0))

image.save("image.png")
```



## Entrelacs

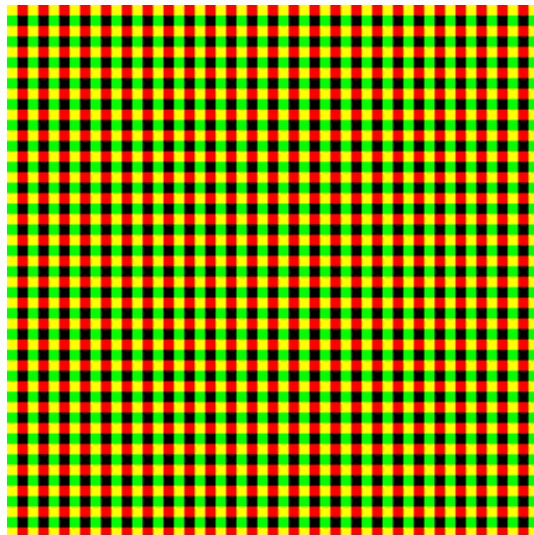
```
from PIL import Image

image = Image.new("RGB", (256, 256))

for y in range(26):
    for x in range(256):
        for h in range(5):
            image.putpixel((x, 10*y+h), (255, 0, 0))

for x in range(26):
    for y in range(256):
        for h in range(5):
            r, g, b = image.getpixel((10*x+h, y))
            image.putpixel((10*x+h, y), (r, 255, 0))

image.save("image.png")
```



## Opération sur les coordonnées

Ici, la couleur de chaque pixel (la « quantité » de rouge, de vert, de bleu) est définie par une fonction de ses coordonnées  $x$  et  $y$ .

```

from PIL import Image

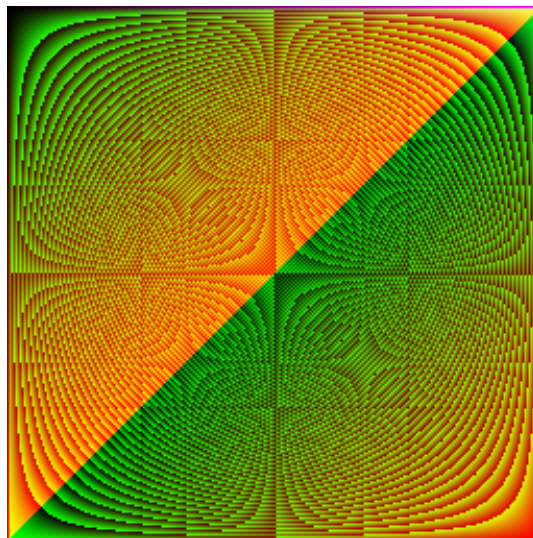
image = Image.new("RGB", (256, 256))

for x in range(256):
    for y in range(256):
        rouge = (x + y) % 256
        vert = (y * x) % 256
        bleu = round(x / (y + 1)) % 256

        image.putpixel((x, y), (rouge, vert, bleu))

image.save("image.png")

```



La « quantité » de rouge, vert et bleu doit être un nombre entier compris entre 0 et 255. Pour nous assurer cela :

- la fonction `round()` permet d'arrondir le nombre à l'entier le plus proche ;
- le modulo `%` permet de « ramener » dans l'intervalle `[0;255]` des nombres trop grands ou trop petits.

## Hasard

On peut aussi définir des coordonnées ou couleurs au hasard. Par exemple, `random.randint(0, 255)` donnera un nombre aléatoire entre 0 et 255.

Le programme suivant produit une « marche aléatoire » : un pixel se déplace aléatoirement sur le dessin, en laissant une trace colorée.

```

from PIL import Image
import random

image = Image.new("RGB", (256, 256))

x = round(255 / 2)
y = round(255 / 2)
rouge = round(255 / 2)
vert = round(255 / 2)
bleu = round(255 / 2)

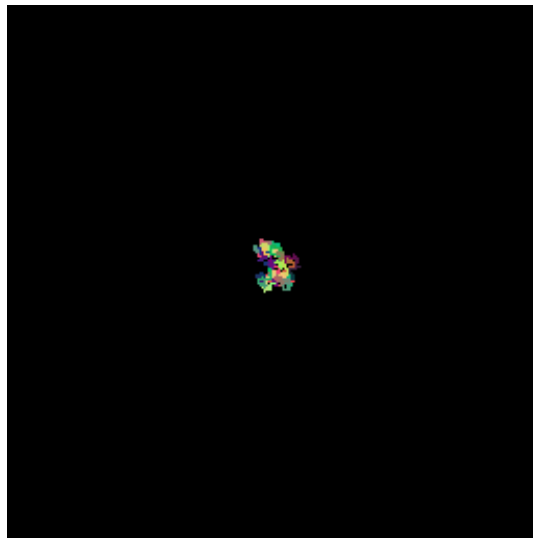
for i in range(1000):
    direction = random.choice(["gauche", "droite", "haut", "bas"])
    if direction == "gauche":

```

```
x = (x - 1) % 256
elif direction == "droite":
    x = (x + 1) % 256
elif direction == "haut":
    y = (y - 1) % 256
else:
    y = (y + 1) % 256

rouge = (rouge + 2) % 256
vert = (vert - 1) % 256
if random.randint(0, 1) == 0:
    bleu = (bleu + 3) % 256
else:
    bleu = (bleu - 3) % 256
image.putpixel((x, y), (rouge, vert, bleu))

image.save("image.png")
```



## À vous !

- Faites une jolie image en utilisant tous ces outils.
- Écrivez bien, en commentaire de votre programme, le nom des deux membres du binôme.
- Rendez le fichier `.py` sur Pronote.