Presque comme Snapchat…. (Partie 1)

Les réseaux sociaux tels que Snapchat proposent des filtres permettant entre autre de modifier les visages en y ajoutant des accessoires ou des effets.

Comment peut-on faire ce type de traitements en utilisant le langage Python ?

I. Comprendre les images numériques

- *D'après vous que se passe-t-il si on zoome sur l'œil du chat ? ……*

Bilan : Les points observés sont appelés pixels. On dit que l'image est matricielle car en Mathématiques une matrice désigne un tableau de nombres.
Remarque : On parle aussi de « carte de points » (de l'anglais « bitmap »)

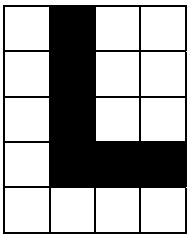
1) Un tableau de nombres ?

a) Cas d'une image noir et blanc

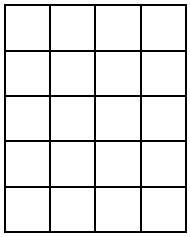
- *Proposer une méthode permettant de coder une image représentant la lettre **L**.*

***INFO :** Dans un ordinateur, seuls des 0 et des 1 (c'est-à-dire des bits) peuvent être stockés ou interprétés.*

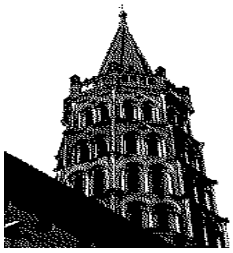
Image de la lettre L



Solution proposée



Bilan : Une image numérique est une image acquise, créée, traitée ou stockée sous forme binaire (suite de 0 et de 1).



Exemple de format :

Format pbm : Format d'images en noir et blanc inventé dans les années 1980 pour pouvoir permettre la transmission d'images par message électronique.

←Image contenue dans le fichier Sernin.pbm

INFO : En informatique, un **format de données** est la façon dont est représenté (codé) un type de données (comme les images), sous forme d'une suite de bits.

b) Cas d'une image dite en niveau de gris

Ci-contre l'image contenue dans le fichier **Sernin.pgm**

INFO: Le format **pgm** est un format d'images en niveau de gris mis au point en 1988.

Voici un extrait d'un fichier image enregistré dans le format **pgm**. Le choix a été fait de coder le blanc par 255 et le noir par 0.

```
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255   0   0   0   0   0   0 255 255 255 122 122 122 122 122 122 255 255
255   0   0   0   0   0   0 255 255 255 122 122 122 122 122 122 255 255
255   0   0 255 255 255 255 255 255 255 255 255 122 122 255 255 255 255
255   0   0 255 255 255 255 255 255 255 255 255 122 122 255 255 255 255
255   0   0   0   0   0   0 255 255 255 255 255 122 122 255 255 255 255
255   0   0   0   0   0   0 255 255 255 255 255 122 122 255 255 255 255
255 255 255 255 255   0   0 255 255 255 255 255 122 122 255 255 255 255
255 255 255 255 255   0   0 255 255 255 255 255 122 122 255 255 255 255
255   0   0   0   0   0   0 255 255 255 255 255 122 122 255 255 255 255
255   0   0   0   0   0   0 255 255 255 255 255 122 122 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

- *Combien de niveaux de gris est-il possible d'obtenir dans ce cas-là ?*
- *Sur combien de bits est stockée la « couleur »?*

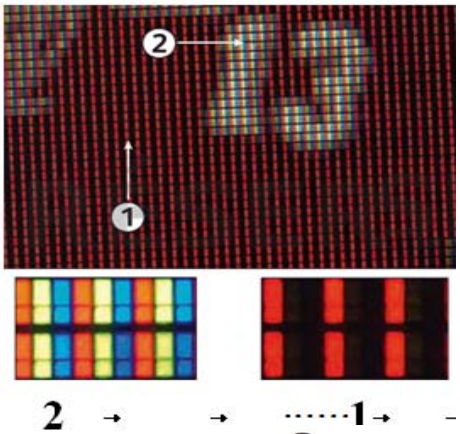
- *Quelle est l'image stockée dans ce fichier ?*

- La définition d'une **image numérique** correspond au nombre de pixels qui composent l'**image** en hauteur (axe vertical) et en largeur (axe horizontal). Quelle est la définition de l'image précédente ?

c) Cas d'une image couleur

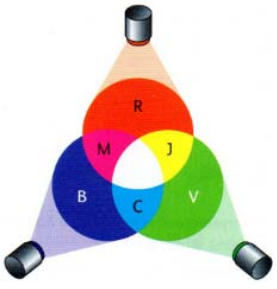
Le principe physique

Voici un agrandissement de l'image d'un écran LCD (Liquid Cristal Display)
On peut distinguer les **pixels** qui la composent.



Si on agrandit encore l'image, on peut voir que chaque pixel est subdivisé en 3 rectangles (un rouge, un vert et un bleu) : ce sont les **sous-pixels.**

http://www.ostralo.net/3_animations/swf/couleurs_ecran.swf



- *Compléter le tableau ci-dessous :*

sous-pixels allumés	**B** **V** **R**	**B** ■ **R**	■ **V** **R**	**B** **V** ■	■ ■ **R**
couleur perçue	blanc ①				rouge ②

<u>Bilan :</u> L'œil et le cerveau réalisent une synthèse additive des couleurs. Toute lumière colorée peut être reproduite en superposant, dans certaines proportions, trois faisceaux lumineux de couleurs rouge, verte et bleue (nos yeux ne sont sensibles qu'à ces trois couleurs).

Codage des couleurs

Chaque pixel est codé par trois valeurs (rouge, vert et bleu).

Exemple :

255 **0** **0**	**0** **255** **0**	**0** **0** **255**
255 **255** **0**	**255** **255** **255**	**0** **0** **0**

donne

Rouge	**Vert**	**Bleu**
Jaune	**Blanc**	**Noir**

- *En vous aidant de l'exemple donné ci-dessus, indiquer la valeur associée à chaque sous-pixel dans le cas proposé.*

Rouge	**Magenta**	**Cyan**

- *Sachant que la profondeur de couleur est le nombre de bit associés à chaque pixel, indiquer la profondeur de couleur dans le cas traité.*
- *Que se passe-t-il quand les trois couleurs ont la même valeur (par exemple 125,125,125)*
- *Combien de couleurs peuvent être codées en associant 3 octets à chaque pixel ?*

<u>Bilan : Codage des couleurs</u>
Un pixel est composé de trois sous-pixels (rouge, vert et bleu). On parle de système RVB (ou RGB en anglais). On codera un pixel à l'aide d'un triplet de valeur. La 1ère valeur donnant l'intensité du rouge, la 2ème du vert et la 3ème du bleu. La valeur de l'intensité lumineuse associée à chaque couleur est souvent comprise entre 0 et 255 (256 valeurs possibles), soit environ 16,8 millions de couleurs possibles.

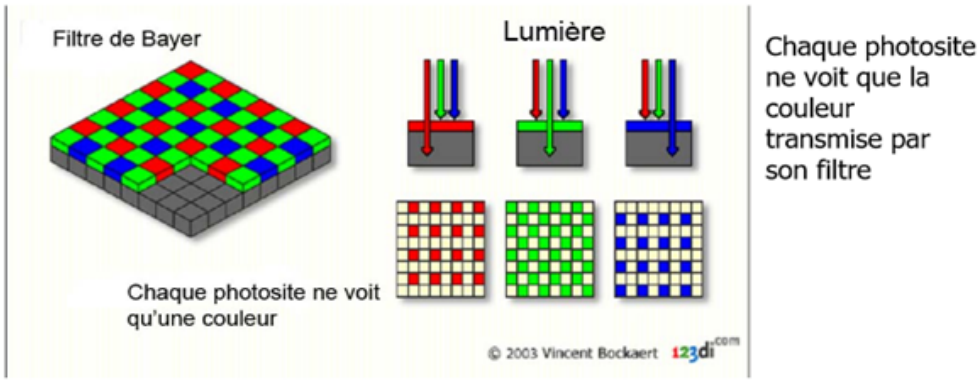
Partie 3 : Presque comme Snapchat….

Comment l'image est capturée

1) Le capteur de l'appareil photo

Les appareils photo numérique sont équipés de capteurs composés d'une matrice d'éléments qui réagissent à la quantité de lumière qu'ils reçoivent. Ce sont les photosites.

La surface du capteur est recouverte d'un filtre (filtre de Bayer)



Les filtres de bayer font en sorte qu'un photosite ne reçoive qu'une couleur primaire. Comme dans notre œil, il y a plus d'éléments sensibles au vert qu'au rouge ou au bleu

2) Les photosites

Rectangle pris en photo	Photosites activés	
		Lorsqu'on photographie un rectangle blanc tous les photosites sont activés
		Lorsqu'on photographie un rectangle vert seuls les photosites sensibles au vert sont activés
		Lorsqu'on photographie un rectangle rouge seuls les photosites sensibles au rouge sont activés
		Lorsqu'on photographie un rectangle bleu seuls les photosites sensibles au bleu sont activés

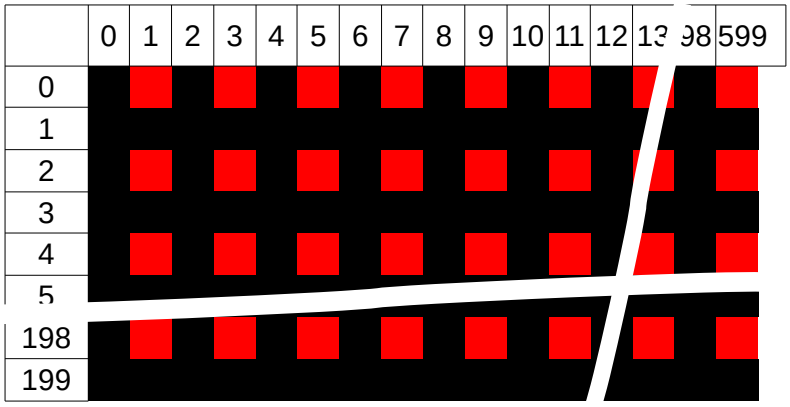
Compléter le tableau pour les rectangles suivants	
JAUNE	Lorsqu'on photographie ce rectangle les photosites activés sont ceux sensibles : o Au rouge o Au bleu o Au vert
MAGENTA	Lorsqu'on photographie ce rectangle les photosites activés sont ceux sensibles : o Au rouge o Au bleu o Au vert
CYAN	Lorsqu'on photographie ce rectangle les photosites activés sont ceux sensibles : o Au rouge o Au bleu o Au vert

3) Photosite et pixel

Un photosite ne correspond pas à un pixel car il ne contient l'information que pour une couleur (Rouge ou Verte ou Bleue) alors que nous l'avons vu précédemment le pixel a besoin des trois. Même une image RAW est donc d'abord recomposée par le processeur d'image qui s'appuie sur les pixels adjacents pour recréer les deux canaux colorés manquant. En utilisant les informations des photosites rouges et bleus à proximité d'un photosite « vert », le processeur compose un « pixels » codant pour les trois couleurs. Lorsque l'on est en lumière faible, il faut plus de photosites pour recomposer un pixel (perte de définition).

4) Signal brut

Pour avoir une idée du signal brut à la sortie de capteur nous allons faire un programme à partir du fichier « CarreRouge.png » du programme 6.
Voilà ceux que l'on veut obtenir :



Ouvrir le programme « carrerougebrutELV.py et le compléter

```
# --on importe le module Image de la bibliothèque PIL
from PIL import Image

# --on ouvre le fichier image
img = Image.open('carrerouge.png')

# --on récupère et on affiche la largeur et hauteur de l'image
colonne, ligne = img.size
print(colonne)
print(ligne)

# --on balaye toutes les colonnes
for x in range (colonne):
    # --on balaye toutes les lignes
    for y in range(ligne):
        # --on vérifie la parité des colonnes pour les laisser rouges
        # --ou les colorier en noir
        ... x % 2 == 0:
            img.putpixel((x, y), (...,...,...))
        # --on vérifie la parité des lignes pour les laisser rouges
        # --ou les colorier en noir
        .. y % 2== 0:
            img.putpixel((x, y), (...,...,...))

# --on affiche l'image
img.show()
# --on enregistre l'image en la nommant 'nom.png'
img.save('RougeBrut.png')
# --on ferme l'image
img.close()
```

Mettre le bon code R,V,B

Mettre la bonne condition

Observer l'image « RougeBrut .png » pour voir le rendu du signal brut du capteur.