Travaux dirigés #1

Donner le code `Python` et `Pseudo-code` des 2 premières questions et le `Pseudo-code` pour le reste.

Gestion d'images en niveau de gris

Une manière simple de représenter une image est de considérer une matrice (*liste de listes*) de valeurs entières. Comprises entre 0 et une valeur maximum p, ces valeurs représentent une intensité (un niveau de gris) et permettent donc d'obtenir une image. La valeur 0 correspond au noir, et la valeur p au blanc. Par exemple pour $p = 255$:

0	255	140	90
220	0	45	200
110	70	0	80
40	230	150	0

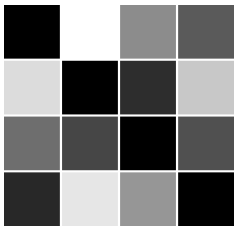


FIGURE 1 – Une matrice 4 par 4 et son image en niveaux de gris correspondante.

Dans la suite de ce TD, on fixe $p = 65536$. Une manière simple de parcourir l'image est donc la suivante :

```python
for ligne in image:
  for pixel in ligne:
    # pixel : une valeur entière
```

```
Pour i de 1 à taille(image) faire
  Pour j de 1 à taille(image[i]) faire
    # image[i][j] : valeur entière
```

On rappelle que la fonction `enumerate` permet de parcourir une liste en récupérant un couple `(indice,élément)`.

Exercice 1 *Images of you.*

Chacune des fonctions suivantes nécessite, dans un premier temps, de créer une image noire.

1. Donner un code permettant de remplir une matrice de 0, les dimensions étant saisies au clavier par l'utilisateur. Transformer ce code en **fonction**.
2. Définir une fonction permettant de *seuiller* une image : tout pixel inférieur au paramètre S sera mis à 0, et toute valeur supérieure à S à 65536. Quels sont les paramètres attendus pour cette fonction ?
3. Définir une fonction permettant d'appliquer un filtre *négatif* à une image : toutes les valeurs de pixel sont inversées. *Attention* : ne pas oublier que toutes les valeurs doivent être positives.
4. Définir une fonction permettant d'appliquer un filtre *gradient* à une image : la valeur de chaque pixel est remplacée par la différence entre le pixel courant et le pixel situé au-dessus. *Attention* : ne pas oublier que toutes les valeurs doivent être positives.

FIGURE 2 – Exemple de filtres négatif et gradient sur l'image précédente.

Exercice 2 *This is one for the good days — And I have it all here — In red, blue, green.*

La première étape pour appliquer les filtres précédents sur n'importe quelle image est de convertir une image quelconque en niveau de gris. Une représentation classique des images est d'attribuer trois valeurs entières entre 0 et 255, représentant respectivement les valeurs `Rouge`, `Verte` et `Bleue` (`RVB` ou `RGB`). Ainsi :

$(94, 173, 191)$	$(13, 169, 16)$	$(41, 157, 239)$
$(85, 13, 97)$	$(158, 170, 71)$	$(213, 66, 153)$
$(121, 216, 166)$	$(255, 197, 62)$	$(169, 19, 140)$

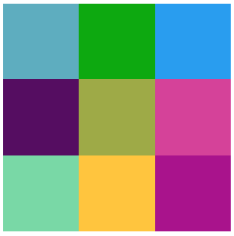


FIGURE 3 – Une image 3×3 et ses pixels `R,V,B`.

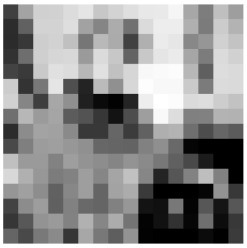
Une image `RGB` peut donc être considérée comme un tableau à 2 dimensions où chaque valeur contient un triplet (tuple en `python`) contenant les intensités `(R,G,B)`.

Tuples. Les tuples en python sont des ensembles d'éléments particuliers, déclarés comme suit : $t = (1, 2, 3, 4, 5)$. Les tuples peuvent être manipulés comme des listes, à la différence près que les éléments ne peuvent pas être modifiés.

1. Définir une fonction `niveau_de_gris` permettant de convertir une image RGB donnée en paramètre en image en niveau de gris. Pour ce faire, les intensités ne sont pas simplement ajoutées et divisées par 3, mais la formule suivante est utilisée :

$$0.2125 * R + 0.7154 * V + 0.0721 * B$$

2. **Approfondissement.** Donner le code d'une fonction permettant de flouter une image selon le principe suivant : la fonction attend en arguments, en plus de l'image, un entier représentant la dimension du carré définissant les zones à flouter. **On supposera que l'image est carrée et que cet argument divise la dimension de l'image**. Cette dernière sera donc découpée en un certain nombre de carrés de taille c, et à l'intérieur de chaque carré les valeurs des pixels seront remplacés par la moyenne du carré. Voir ci-contre pour une image de taille 512×512 avec un carré de côté 32.



Comment tester les fitres ?

L'extrait de code suivant vous permet de tester facilement, dans `jupyter`, vos filtres :

```
# permet d'afficher les images
import matplotlib.pyplot as plt

original = img_as_int(data.astronaut())
hauteur = original.shape[0]
largeur = original.shape[1]
# une matrice représentant une image en niveau de gris
image = niveau_de_gris(original, hauteur, largeur)
plt.imshow(image, cmap=plt.cm.gray)
# pour supprimer les axes abscisse/ordonnée
plt.axis("off")
plt.show()
```

Exercice 3 — *Un joyeux non-anniversaire.*

Désireux de souhaiter un joyeux non-anniversaire à son ami le Lièvre de Mars, le Chapelier fou a décidé de disséminer 300 colis dans le Pays des Merveilles. Chaque colis possède un numéro entre 1 et 300 : un seul contient le cadeau de son ami, mais il s'auto-détruira dans 24 heures. Les colis sont de plus habilement protégés : il est impossible d'en ouvrir un avant de les avoir tous réunis au même endroit. Après une recherche acharnée, accompagné de plusieurs comparses, le Lièvre de Mars arrive à réunir tous les colis en 23 heures. Le temps demandé pour ouvrir un colis est de 20 secondes. Il lui reste donc 60 minutes pour s'assurer d'ouvrir le colis contenant le cadeau. En ouvrant les premiers colis, le Lièvre de Mars réalise que chaque colis possède une image avec un unique symbole : soit $\rightarrow$ soit $\leftarrow$.

1. Le premier comparse du Lièvre de Mars suggère d'ouvrir les colis sans tenir compte des identifiants ni des symboles. Cette stratégie assure-t-elle que le cadeau sera trouvé à temps ?
2. Au bout de 20 minutes et 60 colis ouverts, son second comparse propose lui de prendre en compte les identifiants des colis restants en les triant dans l'ordre croissant. On admet que cette opération lui prend un temps de 32 minutes. Il propose ensuite de les ouvrir dans l'ordre, en partant du dernier non ouvert si un $\rightarrow$ est ouvert, et en revenant au premier non ouvert sinon. A nouveau, cette stratégie a-t-elle une chance de fonctionner ?
3. Alors qu'il ne reste que 5 minutes et réalisant que leurs stratégies ne marchent pas, ils décident en désespoir de cause de faire appel à leur cellule informatique [1]. Voyant le nombre de colis restants et profitant de leur tri, cette dernière propose immédiatement une solution fonctionnelle. Laquelle ?
4. Donner une implémentation de cette solution, en considérant les colis comme une liste déjà triée.

Exercice 4 — *This is my last sort.*

Approfondissement. Peu de temps après cette folle journée, le Lapin Blanc, qui a difficilement vécu cette épreuve, terrifié à l'idée d'ouvrir le cadeau en retard, se demande si le fait que les colis soient identifiés de 1 à X n'aurait pas permis de les trier plus rapidement que ce que le second comparse a réalisé. Pour cela, il considère l'ensemble des colis (dans l'ordre dans lesquels ils ont été initialement ouverts) comme un tableau T.

1. En supposant l'existence d'un autre tableau contenant X éléments, comment pourrait-on obtenir une version triée du tableau initial ?
2. Mécontent de devoir passer par de nombreuses variables intermédiaires (les cases de l'autre tableau), le Lapin Blanc finit par trouver une solution permettant de n'utiliser (presque) que le tableau original pour trier le tableau : laquelle ?

1. On admettra que cette dernière a sa place au Pays des Merveilles.