IV - TP 2

Traitement d'images

1 Introduction

Nous allons nous intéresser dans ce TP à la création et à la modification d'images numériques. En python, on utilise la libraire `PIL` (pour Python Imaging Library) pour manipuler les images.
Pour charger un objet de type `Image`, on peut utilise le code suivant.

Code python

```
from PIL import Image
img = Image.open("baboon.png")
img.show()
```

Ceci affecte à la variable `img` un objet de type `Image` que l'on peut manipuler à l'aide des fonctions python du module `Image`.

Une image numérique peut être vue comme une tableau 2D (on dit aussi matrice) de pixels. Une image est dite de dimension $m \times n$ si le tableau de pixel qui la représente est constitué de m lignes de n pixels. Chaque pixel de l'image est repéré par ses coordonnées.
Attention, les conventions changent par rapport aux mathématiques : l'origine du repère est le coin supérieur gauche de l'image, et les ordonnées sont orientées **vers le bas**.

row \ col	0	1	2	3	4	5
0	(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)
1	(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)
2	(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)
3	(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)

x

y

Une image peut-être manipulée selon deux modes : le mode niveaux de gris (mode `"L"`) et le mode couleur (mode `"RGB"`).

- En mode noir et blanc, la couleur de chaque pixel est décrite par un nombre écrit sur 8 bits. `0` représente la couleur noire, `255` représente la couleur blanche.
- En mode couleur, la couleur de chaque pixel est décrite par un triplet de nombres `(r, g, b)` écrits sur 8 bits qui indique comment mélanger les trois couleurs rouge (**r**ed), vert (**g**reen) et bleu (**b**lue).

  Par exemple, `(255, 0, 0)` signifie un rouge pur (rouge au maximum, vert et bleu absents). `(0, 255, 0)` est un vert pur. `(0, 0, 255)` est un bleu pur. `(255, 255, 255)` représente le blanc (toutes les couleurs au maximum). `(0, 0, 0)` représente le noir (absence de toutes les couleurs). `(128, 128, 128)` serait un gris moyen.

Par exemple, le code ci-dessous créé une image de dimensions 4×2 au format noir et blanc et modifie le pixel de coordonnées `(3, 1)` afin que celui-ci soit blanc. Enfin, on sauvegarde l'image dans le fichier `exemple.png`.

Code python

```
img2 = Image.new("L", (4, 2))
img2.putpixel( (2, 1), 255)
img2.save("exemple.png")
```

On donne dans le tableau ci-dessous la liste de toutes les fonctionnalités utiles du module `Image`.

Instruction	**Description**
`nom_variable = Image.open("chemin_fichier")`	Ouverture d'un fichier image
`nom_variable.show()`	Affiche l'image `nom_variable`
`nom_variable.mode`	Mode de l'image : `"L"` en niveaux de gris, `"RGB"` en couleurs
`nom_variable.format`	Format de l'image: "PNG", "PPM", ...
`nom_variable.size`	Dimensions de l'image sous la forme d'un tuple `(largeur,hauteur)`
`nom_variable.save("nom_fichier")`	Sauvegarde l'image dans `nom_fichier`
`nom_variable = Image.new(mode,(l,h)`	Création d'une image dans le `mode` de dimensions `(l, h)`
`nom_variable.getpixel((x,y))`	Pixel de coordonnées `(x,y)`. Un nombre ou un triplet, suivant le mode de l'image
`nom_variable.putpixel((x,y), valeur)`	Écriture de la `valeur` dans le pixel de coordonnées `(x, y)`

2 Exercices

1 Récupérer le fichier `"baboon.png"` sur le github du cours.
Créer un nouveau fichier `tp.py` et y recopier les lignes suivantes.

Code python
```python
from PIL import Image
img = Image.open("baboon.png")
```

1. Déterminer dans la console, le format, le mode et les dimensions de l'image.
2. Lire et essayer de comprendre le code suivant, quel va être l'image produite ? Puis, tester ce code.

Code python
```python
from PIL import Image

im = Image.open("baboon.png")
largeur, hauteur = img.size
# L'instruction suivante crée une nouvelle Image en mode RGB
# et qui posséde les mêmes dimensions que img
img2 = Image.new("RGB", img.size)

for y in range(hauteur): #parcours des lignes
    for x in range(largeur): #parcours des colonnes d'une ligne
        pixel = im.getpixel((x, y))
        im2.putpixel((x, y),(pixel[0], 0, 0))

img2.show()
```

3. Modifier le programme qu'il créé une image qui ne garde que la composante en bleu de l'image et affiche cette image.

2 Ecrire un programme qui crée une image en mode `"RGB"` de dimension 512×512 pixels. Pour chacun des pixels, la couleur correspondra à trois nombres différents tirés au hasard, à l'aide de la fonction randint du module random.

Code python

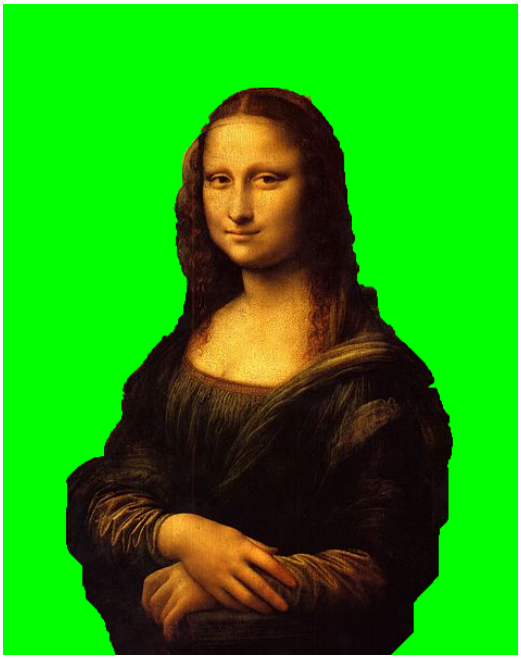
```
from random import randint

print(randint(0, 10)) # tire un nombre aléatoire entre 0 (inclus) et 10
↪ (inclus)
```

3 Pour convertir une image couleur en niveau de gris, un pixel représenté par ses composantes `(r, g, b)` doit être remplacé par un pixel à une seule composante codé par un nombre entier compris entre `0` et `255` et appelé luminance.

Une première approche est d'utiliser la moyenne $m = \dfrac{r+g+b}{3}$ pour calculer la luminance.

Ecrire un programme qui crée une image en mode `"L"` correspondant à la conversion en niveau de gris de l'image `"baboon.png"`.

4 Récupérer les fichiers `"mona_fond.png"` et `cri.jpg` depuis le github du cours.



1. Quel est le code RGB de la couleur des pixels verts dans l'image `mona_fond.png` ?
2. Ecrire un programme créant une image `mona_cri.png`. Cette image devra contenir une incrustation de Mona Lisa dans le tableau Le Cri : chacun des pixels verts de Mona Lisa doit être remplacé par un pixel du Cri de mêmes coordonnées.

5 Pour inverser les contrastes d'une image en niveaux de gris, il suffit d'appliquer à chaque pixel `x` la valeur `255 - x`. Le blanc devient noir et vice-versa.
Compléter dans le programme ci-dessous le code de la fonction `inverser_gris` pour qu'il prenne en argument une `Image` en niveau de gris, et renvoie une `Image` dont les contrastes sont inversés.

Code python

```python
from PIL import Image

im = Image.open("boats.png")

def inverser_gris(image):
    """ Image -> None """
    # À compléter

im2 = inverser_gris(im)
im.show()
```

On pourra utiliser le fichier `"boats.png"` du dépôt github afin de tester la fonction.

6 Pour transformer une image en niveaux de gris en une image en noir et blanc, on convient d'un seuil `s` (avec $0 \leq s \leq 255$) et on remplace chaque pixel de valeur `p` soit par `0` si $p \leq s$ soit par `255` sinon.

1. Ecrire une fonction `convertir_nb`, qui prend en argument une image et un seuil. Elle renvoie l'image convertie avec la méthode de l'énoncé, sans modifier l'image originale.
2. Tester cette fonction sur l'image `"boats.png"` pour différentes valeurs de seuil.
3. Ecrire une fonction `convertir_gris`, qui prend en argument une image en mode RGB et renvoie une image en niveau de gris. Pour convertir en niveau de gris un pixel de code `(r,g,b)` on calcule la luminance `l` avec la formule $l = 0,299r + 0,587g + 0,114b$.
4. Appliquer la fonction `convertir_gris` puis `inverser_gris` puis `convertir_nb` avec un seuil de `130` à l'image `"baboon.png"`.

7 Le drapeau du Cameroun est un drapeau tricolore dont les couleurs sont vert `(0, 122, 94)`, rouge `(208, 20, 37)` et jaune `(252, 208, 23)`.

Créer et afficher une image de dimensions 640×429 correspondant au drapeau camerounais (sans l'étoile du milieu).