

Informatique tronc commun TP 08

11 janvier 2017

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Commencez la séance en créant un dossier au nom du TP dans le répertoire dédié à l'informatique de votre compte.
3. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
4. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux.
5. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez les !
6. Vous devez être autonome. Ainsi, avant de poser une question à l'enseignant, merci de commencer par :
 - relire l'énoncé du TP (beaucoup de réponses se trouvent dedans) ;
 - relire les passages du cours¹ relatifs à votre problème ;
 - effectuer une recherche dans l'aide disponible sur votre ordinateur (ou sur internet) concernant votre question.

Il est alors raisonnable d'appeler votre enseignant pour lui demander des explications ou une confirmation !

On s'intéresse dans ce TP à la manipulation d'images en Python. **On commencera par se rendre sur le site de la classe et par enregistrer dans le répertoire du TP les fichiers suivants :**

- `degrade.pgm` ;
- `joconde.pgm`.

Instructions de rendu

Attention : suivez précisément ces instructions. Vous enverrez à votre enseignant un fichier d'extension `.py`. Votre fichier portera un nom du type `tpXX_berne_zannad.py`, où `XX` est à remplacer par le numéro du TP et les noms de vos enseignants par ceux des membres du binôme. Le nom de ce fichier ne devra comporter ni espace, ni accent, ni apostrophe. Dans ce fichier, vous respecterez les consignes suivantes.

1. Dans le cas fort improbable où vous ne vous en souviendriez pas.

- Écrivez d'abord en commentaires (ligne débutant par #), le titre du TP, les noms et prénoms des étudiants du groupe.
- Commencez chaque question par son numéro écrit en commentaires.
- Les questions demandant une réponse écrite seront rédigées en commentaires.
- Les questions demandant une réponse sous forme de fonction ou de script respecteront pointilleusement les noms de variables et de fonctions demandés.

1 Images en niveaux de gris

On cherche à représenter des images. Pour simplifier, on ne cherchera pas à représenter les couleurs mais seulement la luminosité des différents points de l'image. Pour cela, on découpe l'image, supposée rectangulaire, en carrés de même taille (appelés *pixels*). On remplace alors chaque carré par un entier naturel indiquant la luminosité (moyenne) de l'image sur le carré considéré (la luminosité est donnée dans une unité arbitraire, les valeurs allant de 0 pour un carré noir à N pour un carré blanc, où N est une valeur arbitrairement fixée). On obtient ainsi une matrice de $n \times p$ valeurs entières appartenant à l'intervalle $[0, N]$.

Pour lire et écrire une telle image dans un fichier, on peut utiliser le format de données PGM (Portable GrayMap), dans sa version texte (*plain format*).

Ce format de données consiste à représenter une image de la façon suivante par un fichier.

- Le fichier est un fichier texte (ne comportant que des caractères ASCII).
- On appellera *blanc* tout caractère qui est soit un retour à la ligne, soit un caractère espace, soit un caractère de tabulation (en fait, un autre caractère)
- Toutes les valeurs contenues dans le fichier sont séparées par un ou plusieurs blancs.²
- La première ligne du fichier doit contenir la «valeur magique» constituée des deux caractères «P2» (cette contrainte sert à distinguer un fichier pgm en niveaux de gris d'un autre type de fichier) et doit être suivie d'un blanc.
- Les autres valeurs écrites dans le fichier sont toutes des entiers naturels, écrits en base 10 (autrement dit, «douze» est représenté par la succession des caractères 1 et 2).
- Après la valeur magique, on trouve un nombre représentant la largeur p de la matrice puis un nombre représentant la hauteur n de l'image, puis un nombre donnant la valeur de N choisie pour cette image (intensité de gris représentant le blanc), puis toutes les valeurs de la matrice représentant l'image, dans l'ordre où on les lirait normalement si la matrice était un texte écrit en français (c'est-à-dire de gauche à droite puis de haut en bas).
- Le fichier peut contenir des commentaires ; ceux-ci commencent par le caractère # et finissent avec le retour à la ligne suivant. Ils doivent être simplement ignorés. Pour faciliter le travail demandé par la suite, *on supposera qu'il n'y a aucun commentaire dans les fichiers que l'on traitera.*
- On doit avoir $N \in \llbracket 0, 2^{16} \rrbracket$.
- Les lignes du fichier doivent faire au plus 70 caractères.

2. À toutes fins utiles, pour toute chaîne de caractères s , l'expression $s.split()$ désigne la liste obtenue par découpage de s en utilisant les blancs comme séparateurs.

On rappelle qu'on peut convertir une chaîne représentant un nombre décimal avec `int` et réciproquement, convertir un nombre en chaîne le représentant avec `str`.

Enfin, on représentera une matrice (de nombres) à n lignes et p colonnes comme un tableau (ou liste Python) de longueur n , chacun de ses éléments étant un tableau de longueur p . On décrit donc la matrice ligne par ligne. Ainsi, l'indice « ligne i colonne j » de la matrice représentée par M sera $M[i][j]$.

Exemple 1.0.1. La matrice $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}$ sera représentée par `[[0,1,2] , [3,4,5]]`.

2 Travail demandé

Commencez par recopier les fonctions `image_noire`, `dim` et `lit_valeurs` données ci dessous : ce sont des exemples utiles, utilisez les !

```
def image_noire(n, p):
    """Construit la matrice n*p d'une image noire."""
    img = [0]*n
    for i in range(n):
        img[i] = [0]*p
    return img

def dim(img):
    """Donne le couple (n, p) des dimensions de la matrice img. n :
    nombre de lignes, p : nombre de colonnes. La matrice est supposée
    avoir au moins une ligne."""
    n = len(img)
    p = len(img[0])
    return (n,p)

def lit_valeurs(nom_de_fichier):
    """Lit le contenu du fichier image f et retourne la liste des
    valeurs lues (séparées par des blancs) sous forme d'une liste
    de chaînes de caractères. La première valeur est normalement
    'p2'."""
    with open(nom_de_fichier, 'r') as f:
        c = f.read()
    return c.split()
```

N'hésitez pas à ouvrir l'image `degrade.pgm` et `joconde.pgm` dans un éditeur de texte puis dans un lecteur d'images, afin de comprendre le codage des images.

2.1 Sauvegarde d'images

Q1 Écrire une fonction `sauve_image(img, N, nom_de_fichier)` prenant en argument une matrice `img` représentant une image, l'entier N comme niveau de gris maximal (dans $\llbracket 0, 2^{16} \rrbracket$) ainsi qu'une chaîne `nom_de_fichier` et sauveant l'image dans le fichier nommé `nom_de_fichier`, au format PGM.

Q2 Écrire une fonction `sauve_rectangle_noir(n, p, N, nom_de_fichier)` sauvant dans le fichier `nom_de_fichier` un rectangle noir, de côté n pixels de hauteur et p de largeur, où le blanc est d'intensité N . Vérifier que l'image produite par

```
sauve_rectangle_noir(100, 200, 255, 'rectangle_noir.pgm')
```

est bien ce que vous attendiez grâce à GIMP ou la visionneuse d'images.

Q3 Écrire une fonction `sauve_rectangle_blanc(n, p, N, nom_de_fichier)` sauvant dans le fichier `nom_de_fichier` un rectangle blanc, de côté n pixels de hauteur et p de largeur, où le blanc est d'intensité N . De même, vérifiez l'image produite par

```
sauve_rectangle_blanc(100, 200, 255, 'rectangle_blanc.pgm').
```

Q4 (Question optionnelle) Écrire une fonction `sauve_echiquier(p, N, nom_de_fichier)` produisant dans le fichier `nom_de_fichier` l'image d'un échiquier, où chaque case de l'échiquier a pour côté p pixels et N est le niveau d'intensité du blanc. Pour mémoire un échiquier a 64 cases, et dans sa représentation traditionnelle, la case en bas à droite est blanche.

2.2 Lecture et modification d'images

On pourra, pour tester les fonctions écrites ici, utiliser d'une part les images précédemment produites, d'autre part utiliser l'image disponible sur le site de la classe.

Q5 Écrire une fonction `lit_image(nom_de_fichier)`, où la chaîne `nom_de_fichier` représente le nom d'un fichier PGM et retournant un couple `(img, N)` où N est le niveau d'intensité du blanc de l'image et `img` la matrice des pixels.

On suppose que le fichier respecte les contraintes données dans l'énoncé et on ne fera aucun effort particulier pour gérer les situations où il ne les respecterait pas.

Par exemple, votre fonction a le droit d'accepter un fichier dont les lignes font plus de 70 caractères. On suppose de plus que le fichier ne contient aucun commentaire.

Remarque 2.2.1. On pourra utiliser la fonction `lit_valeurs`, que vous trouverez sur le site de classe.

Q6 Écrire une fonction `negatif(fichier_entree, fichier_sortie)` prenant en argument deux noms de fichiers `fichier_entree` et `fichier_sortie`. La fonction lit d'abord le fichier `fichier_entree` et crée le fichier `fichier_sortie` obtenu en remplaçant chaque pixel de niveau de gris k par un pixel de niveau $N - k$, où N est l'intensité du blanc du fichier d'entrée.

Q7 (Question optionnelle) Écrire une fonction `rotation90(fichier_entree, fichier_sortie)` lisant le fichier `fichier_entree` et créant le fichier `fichier_sortie` obtenu en effectuant une rotation de 90 degrés (dans le sens trigonométrique) de l'image originale.