TP à rendre	Traitement de l'image – programmer avec Python	TS - ISN

Pourquoi ce TP ?

Dans le TP « Numérisation des images », nous avons utilisé le logiciel GIMP pour transformer une image en couleur en niveaux de gris, pour réduire sa taille, pour changer sa couleur …

Dans ce TP, nous allons élaborer des algorithmes qui vont nous permettre de comprendre ce que fait le logiciel GIMP.

Des langages de programmation tels que Python, avec notamment la bibliothèque PIL (Python Imaging Library), permettent d'accéder à chaque pixel d'une image matricielle (ou bitmap), et d'en modifier ses caractéristiques (composantes rouge, verte, et bleue)

Le matériel nécessaire :

1)Les photos *« totem512.jpg »*, « *totem512.pgm »* et *« kangourou.pgm »* nécessaires à ce TP vous ont été envoyées par mèl. Copier-les sur votre clé USB dans un dossier nommé *« TP Image_votre nom ».*

Vous pouvez récupérer ce TP en version numérique sur le site ISN au lien suivant :

http://isnlgn.wix.com/isn-lgn-2013#!cahier-de-texte/c1lph

2) La bibliothèque PIL de Python : PIL (Python Imaging Library) est une bibliothèque de fonctions prédéfinies sur Python pour la manipulation des images.

Elle n'est pas pré-intégrée au logiciel. Vous allez devoir la télécharger au lien suivant :

http://www.lfd.uci.edu/~gohlke/pythonlibs/#pil

Descendez sur la page jusqu'à ce qu'apparaisse la ligne **pillow-2.0.0.amd-64-py3.2.exe**
Cliquer pour télécharger puis installer.

Si vous avez un souci d'installation, vous pouvez toujours nous écrire un mèl à isn.lgn@gmail.com mais n'attendez pas le dernier moment ! Votre devoir sera alors considéré comme non fait.

3) Un brouillon et un crayon pour écrire vos algorithmes de programmation en langage naturel sur papier. C'est toujours plus facile !

1. **Utilisation des commandes de PIL**

a. Ouvrir et enregistrer des fichiers d'images

- Tester le code suivant : la photo doit s'afficher à l'écran

```
from PIL import Image     # utilisation de la bibliothèque PIL

file="totem512.jpg"          #la photo tpnum est stockée dans la variable file

im=Image.open(file)       #  charge la photo dans la variable im
```

Attention ! Enregistrer ce script **dans le même dossier** « TP *Image_votre nom* » sous le nom *image1.py*

Remarque** : on pourrait écrire directement : **im=Image.open('totem512.jpg')

En complétant le script contenu dans le fichier image1.py par les lignes suivantes,

* Vous pouvez maintenant **obtenir des informations** sur cette photo :

```
print(im.format , im.size , im.mode)
```

Quelles sont les données qui s'affichent à l'écran ?

.....

***** Vous pouvez **enregistrer une photo sous un autre format** :

im.save ("totem512.png", 'png')	En quel format la photo est –elle changée ? .....
……	Compléter la ligne du script ci-contre pour faire afficher notre image.

***** Vous pouvez **créer une nouvelle image** à l'aide de la fonction **Image.new(mode,size)** où mode est une chaîne de caractère et size, sa taille en pixels.

Ajouter dans votre script image1.py :

`im2=Image.new("L",(256,256))`	Crée une image PGM de taille 256 x256
……	Ligne pour afficher im2

2. Utilisation des pixels

- **Accès aux pixels :**

On accède aux pixels via la fonction ***getpixel((x,y))*** qui renvoie un tableau indexé par des couples d'entiers.
x correspond à la valeur de la ligne et y à celui de la colonne : $(0,0)$ correspond au pixel en haut à gauche de l'image, $(512,512)$ celui en bas à droite .

Ecrire le script suivant et l'enregistrer sous le nom text_pixel.py :

`From PIL import Image`
 	
`print('Nom de l'image :')`
`file=input()`
`im=Image.open(file)`
`pixels=im.getpixel((0,0))`
`print (pixels)`
	Affiche la valeur du pixel en haut à gauche de l'image choisie par l'utilisateur

Utiliser ce script pour afficher des pixels de l'image en couleur puis de celle en noir et blanc. Que remarquez-vous ?

.....

Quelles sont les valeurs maximales possibles pour les entiers x,y de pixels(x,y) ? .....

- **Modifier une image :**

Pour modifier un pixel, on change sa valeur dans le tableau des pixels en utilisant la fonction ***putpixel((x,y),p) :*** on affecte la valeur p au pixel de coordonnées (x,y).

Tester les lignes de commande suivante ; <u>à ajouter</u> dans votre script test_pixel.py :

`im.putpixel((0,0),0)`
`im.save('totem512.jpg')`
`im.show()`
	Est-ce que cela fonctionne avec l'image en noir et blanc ? celle en couleur ? .....
	Quel est l'effet produit ?
	.....

Exercices à rendre (soit sur clé USB, soit par mèl) :

Exercice 1 : Inverser les niveaux de gris

Méthode :
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 .

Ecrire un script que vous nommerez « *contraste.py* » qui inverse les niveaux de gris de l'image « totem512.pgm »

Exercice 2 : Transformer une image en noir et blanc

Méthode :
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 *p* soit par 0 si $p \leq s$ soit par 255 sinon.

Ecrire un script que vous nommerez « *noiretblanc.py* » qui transforme l'image « totem.pgm » en noir et blanc.

Le tester avec différentes valeurs de seuil :

Exercice 3 : Fusionner deux images

Méthode :
Pour fusionner deux images, on calcule la nouvelle image pixel par pixel , la valeur de chaque pixel étant le maximum des valeurs des pixels correspondant dans chacune des images à fusionner.

Ecrire un script qui fusionne les deux images « totem512.pgm » et « kangourou.pgm »