TP 0 : RÉVISIONS

1. Manipulations de listes

Cette année, la structure de données que nous allons préférentiellement utiliser en Python est la structure de liste, que vous avez déjà utilisé l'an dernier. Une liste est un ensemble d'éléments, pas nécessairement du même type (on peut avoir une liste d'entiers, une liste de flottants, une liste de listes, etc), rangés dans un certain ordre.

Les opérations basiques que l'on peut effectuer sur une liste sont l'ajout d'un élément, la lecture d'un élément, la suppression d'un élément. On peut aussi concaténer deux listes (mettre tous les éléments d'une liste à la suite de l'autre).

(1) Rappler les instructions de base qui permettent :
   (a) de lire le i-ème élément d'une liste t ;
   (b) d'extraire le dernier élément d'une liste t ;
   (c) d'ajouter x à la fin d'une liste t ;

(2) Comment s'écrit simplement la concaténation de la liste $t1$ et de la liste $t2$?

(3) Etant donné la liste $t = [1, 2, 3, 4, 6, 5]$, écrire une suite d'instructions basiques qui copie t dans une liste s et qui inverse les deux derniers éléments de t. En exécutant cette suite d'instructions, puis demandant à afficher les deux listes, que remarque t'on ?

(4) Ecrire un algorithme (avec une structure de boucle) qui prend pour argument une liste l et un élément x et qui renvoie un booléen (`True` ou `False`) disant si l'élément x est dans la liste l.

(5) Modifier l'algorithme précédent afin qu'il renvoie la place de l'élément recherché dans la liste si il est présent.

2. Manipulations d'images

Exercice 1. **Structure de l'image matricielle.** Une image au format bmp (bitmap) est en réalité composé d'un ensemble de pixels, et à chaque pixel est affecté une valeur pour chacune des couleurs rouge (R), vert (G) et bleu (B) codée sur 8 bits, donc un entier entre 0 et 255. Un pixel noir est représenté par la valeur (0,0,0), un pixel blanc par (255,255,255), un pixel rouge par (255,0,0), un pixel vert par (0,255,0) et un pixel bleu par (0,0,255). Toute autre valeur donnera un pixel de couleur différente, plus les valeurs étant proches, plus les teintes se ressemblent.

En Python, la bibliothèque pillow permet d'accéder aux valeurs du pixel situé à la ligne x et à la colonne y en appelant la fonction `image.getpixel((x,y))`.Pour modifier un pixel, on utilise la fonction `image.putpixel((x,y),value)` qui affecte la valeur *value* à ce pixel.

Pour essayer de comprendre le fonctionnement de cette bibliothèque on fournit le code ci-dessous (la fonction `Image.open` ouvre l'image choisie en un tuple manipulable par python, la fonction `image.size()` donne un tuple de deux éléments donnant la largeur et la hauteur de l'image en pixels et la fonction `image.show()` affiche l'image dans une fenêtre). Un tuple en python est un objet très semblable à une liste, la principale différence en termes de syntaxe étant qu'un tuple est une séquence d'objets entre parenthèses, alors que pour une liste il faut utiliser des crochets.

```
from PIL import Image

image = Image.open('oiseau.bmp')
```

```
(h,v) = image.size

for i in range(h) :
for j in range(v) :
(r,g,b) = image.getpixel((i,j))
image.putpixel((i,j), ((r+g+b)//3,(r+g+b)//3,(r+g+b)//3))
image.show()
```

(1) A votre avis, quelle opération réalise ce code ?

(2) Plutôt que de prendre la moyenne des 3 valeurs de rouge, bleu et vert, la luminosité d'un pixel est mieux approchée par la formule $l = 0.2989r + 0.5870g + 0.1140b$. Modifier le programme précédent en utilisant cette formule.

Exercice 2. **Effets "spéciaux".** On va dans cette partie écrire des programmes permettant de la manipulation d'images, dans le but d'obtenir différents effets.

(1) L'effet le plus simple est de transformer l'image en une image en noir et blanc. Pour ce faire, il faut définir un seuil s, et choisir en fonction de la luminosité du pixel et du seuil si le pixel doit être affiché en noir ou en blanc. Ecrire un programme réalisant cette opération.

(2) On souhaite maintenant réaliser le négatif d'une image, il suffit alors de changer chacun des niveaux de couleur par son complément à 255. Ecrire ce programme.

(3) Saturation d'une couleur : pour saturer une image en rouge, vert ou bleu, il suffit d'affecter la valeur 255 pour cette couleur à chaque pixel. Ecrire un programme réalisant la saturation en rouge.

(4) Miroir : écrire un programme effectuant une symétrie par rapport à un axe vertical ou un axe horizontal de l'image.

(5) Effet sépia : pour donner un effet ancien à une image, on peut utiliser un filtre sépia. Ceci consiste à passer l'image en niveau de gris, puis à affecter à chaque pixel une valeur dépendant de la luminosité et du seuil. Pour une luminosité l plus petite que le seuil, on affecte une valeur entra noir et sépia pur (codé par le triplet S=(94,38,18)) du type $l/s * S$. Si la luminosité est plus grande que le seuil, on affecte une valeur entre sépia pur et blanc selon $(l-s)/(255-s) * B + (255-l)/(255-s)S$. Ecrire ce programme.