TRAITEMENT D'IMAGES

On utilisera la librairie PIL et nos programmes commenceront par :

```
from PIL import Image
```

L'image à traiter sera `"starWars.jpg"` à placer dans un répertoire `"images"`.

Introduction

Chaque pixel possède une composante rouge, verte et bleue et c'est la somme des 3 qui permet d'obtenir une couleur. On va ainsi pouvoir récupérer pour une image donnée les 3 composantes RGB d'un pixel, ou fixer ces 3 composantes pour un pixel. Le triplet (0,0,0) correspond au noir et le triplet (255,255,255) au blanc.

- `img = Image.open("images/XXX.jpg")`

Ce code permet l'ouverture du fichier "XXX.jpg" en tant qu'image.

- `r,v,b=img.getpixel((100,250))`

Ce code récupère les valeurs de la composante rouge (r), de la composante verte (v) et de la composante bleue (b) du pixel de coordonnées (100,250).

- `img.putpixel((250,250),(255,0,0))`

Ce code permet de colorier le pixel de coordonnées (250,250) en rouge.

On rappelle que pour sauver une image sur le disque, il faut écrire le code suivant :

```
img.save("images/YYY.jpg")
```

Pour visualiser une image :

```
img.show()
```

Pour créer une image :

```
img = Image.new("RGB",(largeur_en_pixel,hauteur_en_pixel))
```

Exercice n°1

Coder la fonction dont la signature est donnée ci-dessous :

```
def echauffement()->Image:
```

Cette fonction doit créer (et renvoyer) une image de 100*100 pixels ayant 4 lignes horizontales (sur fond noir). Une sera rouge, la suivante verte, la suivante bleue et la dernière jaune (uniquement les composante R et G à 255). Le programme principal sauvegardera l'image dans le répertoire «`images` ».

Exercice n°2

Coder la fonction dont la signature est donnée ci-dessous :

```
def opposer(image:Image)->Image:
```

On passe à cette fonction l'image `starWars.jpg`. Elle doit créer (et renvoyer) une image de mêmes dimensions. Chaque pixel de la nouvelle image sera traduit en son opposé, c'est à dire que ses 3 composantes RGB auront pour valeur l'opposé (255- la valeur) des composantes RGB du pixel originel. Le programme principal sauvegardera l'image dans le répertoire «`images` ».

On peut accéder aux attributs width et height d'une image (`img.width, img.height`).

Exercice n°3

Coder la fonction dont la signature est donnée ci-dessous :

```
def niveauDeGris(image:Image)->Image:
```

On passe à cette fonction l'image `starWars.jpg`. Elle doit créer (et renvoyer) une image de mêmes dimensions. Chaque pixel de la nouvelle image sera traduit en un niveau de gris, c'est à dire que ses 3 composantes RGB auront pour valeur la moyenne des composantes RGB du pixel originel. Le programme principal sauvegardera l'image dans le répertoire «`images` ».

Exercice n°4

Coder la fonction dont la signature est donnée ci-dessous :

```
def faireUnCadre(image:Image,epaisseur:int)->Image:
```

On passe à cette fonction l'image `starWars.jpg`. Elle doit créer (et renvoyer) une image de mêmes dimensions. On demande de créer un cadre blanc autour de l'image dont l'épaisseur est passée en paramètre. Le programme principal sauvegardera l'image dans le répertoire «`images`».