Rappels

- Les valeurs étant comprises entre 0 et 255, cela signifie qu'on peut les coder sur 1 octet (8 bits). Évidemment, les bits de poids forts (valeur 128, 64 ou 32) ont une valeur plus importante que les bits de poids faible : changer un bit de poids faible n'a que peu d'incidence sur la couleur du pixel.

- Pour mettre un bit à 0, on effectue un masque (et logique) avec un nombre dont la valeur est la somme des poids de tous les bits à garder (ceux qui restent à 1).

Pour mettre à 0 le bit de poids faible d'une variable, on écrira :

```
var = 195                #par exemple
resultat = var & 254     #resultat vaut 194
```

Pour mettre à 0 les 2 bits de poids faible d'une variable, on écrira :

```
var = 195                #par exemple
resultat = var & 252     #resultat vaut 192
```

- Pour décaler des bits vers la droite ou la gauche, on utilise les opérateurs >> et <<.

Présentation

On peut « mélanger » 2 images (moyenne des valeurs R,G et B des pixels), mais le résultat risque d'être décevant. On peut toutefois « cacher » une image dans une autre en gardant les bits de poids fort de l'image principale (on met à 0 les bits de poids faible) et en ajoutant les bits de poids fort de l'image cachée « à la place » des bits de poids faible de l'image principale.

Exemple :

Imaginons qu'on veut mélanger un pixel dans l'image 1 dont l'octet rouge vaut **190** c'est-à-dire 10111110 avec l'octet de l'image 2 qui vaut **121**, c'est-à-dire 01111001

L'idée est de recombiner les 5 chiffres les plus significatifs du premier pixel avec les 3 chiffres les plus significatifs de l'image 2. Les chiffres significatifs sont les plus importants, ceux qui se trouvent à gauche. Au contraire, les unités ne sont pas très importantes pour nous. De nos 2 octets d'origine 10111 110 et 011 11001, on ne garde que les chiffres les plus significatifs, en mettant ceux de la première image en tête. Ce mélange des deux octets d'origine produit l'octet suivant : 10111 011

Exercice n°5

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

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

On passe à cette fonction l'image `starWarsImageCachee.png`. Elle doit créer (et renvoyer) une image de mêmes dimensions. On doit récupérer les 3 bits de poids faible des composantes RGB de chaque pixel de l'image de départ. Ces 3 bits doivent être décalés pour devenir des « poids forts » et seront ainsi les composantes RGB du pixel à sauvegarder dans l'image d'arrivée. Le programme principal sauvegardera l'image dans le répertoire «`images`».

Exercice n°6

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

```
def melanger(image:Image,image2:Image)->Image:
```

On passe à cette fonction les images `starWars.png` et `ceciEstUnEssai.png`. Elle doit créer (et renvoyer) une image de mêmes dimensions mélangeant les 2 images suivant le processus décrit précédemment : on garde les 5 bits de poids fort de l'image principal et on lui ajoute les 3 bits de poids fort (mais décalés de 5 rangs) de l'image secondaire. Le programme principal sauvegardera l'image dans le répertoire «`images`».

Exercice n°7

Nous allons ici tenter de remplacer un fond vert dans une image. Le fichier avec le fond vert est extrait d'un clip de Vald : vald.png. Et l'image à incruster est une salle de classe : classe.png. Les deux images font la même taille pour faciliter votre travail. Vous pouvez éventuellement changer l'image à incruster. Il pourra être plus facile de détecter le vert avec les couleurs en HLS plutôt que RGB (https://iamvdo.me/blog/les-avantages-de-hsl-par-rapport-a-rgb).

Pour convertir une couleur r,g,b en hls :

```
import colorsys
h,l,s = colorsys.rgb_to_hls(r/255,g/255,b/255)
```

A vous de faire l'incrustation. On pourra, dans un 1er temps, faire un test uniquement sur la valeur de h, qui devra être comprise dans une fourchette équivalente à la couleur verte.

Exercice n°8

Faire de même avec l'image « spiderman.png » à incruster dans un champ de blé (« ble.png ») ou dans le désert (« desert.png »).

Exercice n°9

Pouvez-vous mettre dans les bras de Daenarys (« GOT.png ») une tête de dragon (« dragon.png » ou une autre image que vous chercherez sur Internet) ?

Pour « cacher » un texte dans une image, on peut aussi utiliser les codes ASCII des caractères du texte. Ainsi, il est impossible de « voir » le texte apparaître dans l'image principale. Par exemple, on pourra prendre le code ASCII de chaque caractère et mettre un des bits du code ASCII à la place du bit de poids le plus faible d'une des composantes d'un pixel. L'image n'en sera que très peu modifiée.

Exercice n°10

Coder la fonction « retrouveMessageCache() » à qui on passe une image dans laquelle un message (se terminant par le caractère '\n') est caché dans la première ligne de pixels (uniquement la composante rouge) et qui renvoie le message caché.

Pour cela, on propose l'algorithme suivant :

* Créer un message vide (chaîne de caractères)
* Pour chaque pixel de la première ligne - 8, par pas de 8
    * Initialiser la variable car à 0
    * Répéter 8 fois
        * Récupérer les composantes r,v,b d'un pixel
        * Ajouter à car le dernier bit de r en le décalant d'un rang vers la gauche
    * Ajouter au message le caractère représenté par car (utiliser la fonction chr())
    * Si car vaut '\n', quitter la boucle
* Retourner le message

Tester cette fonction avec l'image "starWarsMessageCachePourLesCiel.png.png" et afficher le résultat.

Exercice n°11

Coder la fonction « cacherMessage() » à qui on passe une image et un message. On propose l'algorithme suivant :

* Créer une liste vide (tous les bits seront placés dans cette liste)
* Pour chaque lettre du message.
    * Mettre dans la variable carac le code ASCII de la lettre (utiliser la fonction ord)
    * Répéter 8 fois
        * Mettre le bit de poids faible dans la liste
        * Déplacer les bits de carac d'un rang vers la droite
* Pour chaque élément de la liste
    * Récupérer les composantes r,v,b d'un pixel de la 1ère ligne de l'image (même rang que la liste)
    * Modifier la composante r pour que son bit de poids faible prenne la valeur de l'élément de la liste
    * Modifier le pixel pour qu'il tienne compte de la nouvelle valeur de r
* retourner l'image modifiée

Tester cette fonction avec l'image « starwars.png » et un message que vous inventerez. Utiliser la fonction précédente pour retrouver le message de départ.