Le but de ce TP est d'écrire un logiciel de traitement d'images.

0 Introduction

Copier l'ensemble du dossier partagé `TP5` sur votre bureau.

1 Fonctions

1. Ouvrir le fichier `fonctions.py` avec Thonny, et identifier les instructions `print`.
2. Exécuter le fichier, et observer la sortie : les instructions `print` sont-elles exécutées dans l'ordre dans lequel elles apparaissent dans le fichier ?
3. Exécuter le fichier pas à pas (fonction *debug* ; bouton avec le scarabé), pour comprendre l'ordre dans lequel sont exécutées les instructions.

2 Extraction de couleurs

1. *Extraction du rouge.*
   (a) Renommer le fichier `traitement-image.py` en `traitement-NOM1-NOM2.py` (où à `NOM1` et `NOM2` sont vos deux noms). L'ouvrir avec Thonny, et l'exécuter (choisir l'image `gere.jpg`, puis l'action `MACHIN`). Vérifier qu'une image `TRUC.png` a été crée, contenant uniquement les couleurs rouges de l'image d'origine.
   (b) Modifier le fichier pour que :
   - ☐ il affiche `Extraire le rouge` plutôt que `MACHIN` ;
   - ☐ la nouvelle image s'appelle `rouge.png` plutôt que `TRUC.png`.

   (c) Tester à nouveau cette fonction sur le fichier `rgbw.png`, et vérifier que le fichier `rouge.png` est conforme à vos attentes.
   (d) Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.
2. *Extraction du vert.* Le but de cette partie est d'ajouter une fonctionnalité à notre programme pour extraire le vert d'une image.
   (a) Copier la fonction `extrait_rouge`, et la renommer en `extrait_vert`. Ajouter la fonction `extrait_vert` à la liste des actions (`ACTIONS`).
   (b) Exécuter le programme. À ce stade, le programme doit proposer deux actions `Extraire le rouge` identiques.
   (c) Modifier la fonction `extrait_rouge` pour qu'elle extraie le vert plutôt que le rouge. Vérifier que les objectifs suivants sont réalisés :
   - ☐ le menu propose l'action `Extraire le vert.` ;
   - ☐ un fichier `vert.png` est produit ;
   - ☐ ce fichier contient uniquement la trame verte de l'image d'origine.

   (d) Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.
3. *Extraction du bleu.*
   (a) Recommencer la question précédente, pour que votre programme propose d'extraire la couleur bleu.

(b) Vérifier les objectifs suivants :

- ☐ le menu propose l'action `Extraire le bleu.` ;
- ☐ un fichier `bleu.png` est produit ;
- ☐ ce fichier contient uniquement la trame bleu de l'image d'origine ;
- ☐ cette action donne le résultat attendu sur les fichiers `gere.jpg` et `rgbw.png`.

(c) Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.

3 Assombrir une image

1. Copier la fonction `extrait_rouge` en la renommant en `assombrit`.
2. Modifier le traitement de chaque pixel (lignes ci-après) pour que la quantité de chaque couleur soit divisées par deux (par exemple, la couleur $(27, 182, 81)$ devient $(13, 91, 40)$).

```
original = source.getpixel((x, y))
nouvelle = (original[0], 0, 0)
dest.putpixel((x, y), nouvelle)
```

3. Vérifier les objectifs suivants.
   - ☐ le menu propose l'action `Assombrir l'image` ;
   - ☐ un fichier `sombre.png` est produit ;
   - ☐ ce fichier contient l'image d'origine, mais plus sombre ;
   - ☐ cette action donne le résultat attendu sur les fichiers `gere.jpg` et `rgbw.png`.
4. Vérifier que si vous appliquez à nouveau cette action sur le même fichier, vous obtenez la même image, encore plus sombre.
5. Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.

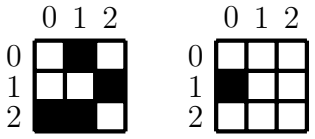
4 Noir et blanc

1. Copier la fonction `assombrit` en la renommant en `noiretblanc`.
2. Modifier le traitement de chaque pixel pour :
   - — calculer la moyenne des trois valeurs de la couleur d'origine ;
   - — affecter cette moyenne à chacune des valeurs des trois couleurs de l'image de destination.

   Par exemple, la moyenne des trois valeurs de la couleur $(27, 182, 81)$ est $\frac{27+182+81}{3} = 96$, donc la nouvelle couleur sera $(96, 96, 96)$.
3. Vérifier les objectifs suivants.
   - ☐ le menu propose l'action `Convertir en noir et blanc.` ;
   - ☐ un fichier `noiretblanc.png` est produit ;
   - ☐ ce fichier contient l'image d'origine, mais convertie en noir et blanc.
   - ☐ cette action donne le résultat attendu sur les fichiers `gere.jpg` et `rgbw.png`.
4. Vérifier que si vous appliquez à nouveau cette action sur le même fichier, il ne sera pas modifié à nouveau (il restera en noir et blanc).
5. Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.

5 Symétrie axiale (axe vertical)

1. *Étude théorique.*
   (a) Copier chaque pixel de coordonnées (x, y) de l'image de gauche (en noir et blanc) aux coordonnées $(2 - x, y)$ de l'image de droite.
   Par exemple, le pixel de coordonnées $(2, 1)$, qui est noir, sera copié aux coordonnées $(2 - 2, 1)$, c'est-à-dire $(0, 1)$.



   (b) Quelle symétrie a été réalisée ?
   (c) Dans la transformation $(2 - x, y)$, que représente le « 2 » par rapport à l'image initale ?

2. *Mise en œuvre pratique.*
   (a) Copier la fonction suivante dans votre programme (faire un simple copier-coller depuis le ficher `squelette-symetrie.py`), et ajouter cette fonction à la liste des actions `ACTIONS`.

```
1  def action_symetrie_gauchedroite(nom):
2  """Faire une symérie gauche-droite"""
3  source = Image.open(nom).convert('RGB')
4  largeur = source.width
5  hauteur = source.height
6  dest  = Image.new('RGB', (largeur, hauteur))
7
8  for x in range(largeur):
9      for y in range(hauteur):
10         couleur = source.getpixel((x, y))
11         dest.putpixel(
12             (···-x, y),
13             couleur
14           )
15
16 dest.save("gauchedroite.png")
```

   (b) En utilisant la question précédente, compléter les points de suspension de la ligne 12 pour que la fonction effectue une symétrie axiale par rapport à un axe vertical.

3. Vérifier les objectifs suivants.
   - ☐ le menu propose l'action `Faire une symétrie gauche-droite.` ;
   - ☐ un fichier `gauchedroite.png` est produit ;
   - ☐ ce fichier contient le symétrique de l'image d'origine ;
   - ☐ cette action donne le résultat attendu sur les fichiers `gere.jpg` et `rgbw.png`.

4. Vérifier que si vous appliquez à nouveau cette action sur le même fichier, vous obtenez l'image d'origine.

5. Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.

6 Symétrie axiale (axe horizontal)

1. Copier la fonction définie à la partie précédente, et la modifier pour ajouter une action *Faire une symétrie haut-bas.*.
2. Vérifier les objectifs suivants.
   - ☐ le menu propose l'action `Faire une symétrie haut-bas.` ;
   - ☐ un fichier `hautbas.png` est produit ;
   - ☐ ce fichier contient le symétrique de l'image d'origine ;
   - ☐ cette action donne le résultat attendu sur les fichiers `gere.jpg` et `rgbw.png`.
3. Vérifier que si vous appliquez à nouveau cette action sur le même fichier, vous obtenez l'image d'origine.
4. Copier votre fichier `traitement-NOM1-NOM2.py` dans le dossier partagé.

7 Pour aller plus loin...

Ajouter d'autres fonctions à votre programme. Voici quelques exemples ; vous n'êtes pas obligés de tous les traiter ; vous n'êtes pas obligés de les traiter dans cet ordre ; vous pouvez ajouter d'autres actions qui ne sont pas dans cette liste.

- ☐ Ajouter un cadre à l'image (et proposer à l'utilisateur de choisir la couleur du cadre).
- ☐ Au lieu d'écrire les nouvelles images dans un fichier au nom prédéfini, demander à l'utilisateur de choisir le nouveau nom pour son image.
- ☐ Doubler les dimensions de l'image.
- ☐ Effectuer une rotation de l'image de 90°.
- ☐ Inverser les couleurs de l'image.
- ☐ Permuter les valeurs des trois couleurs de chaque pixel (par exemple, la couleur $(27, 182, 81)$ devient $(81, 27, 182)$).
- ☐ Réduire de moitié les dimensions de l'image.
- ☐ Transformer les couleurs de l'image pour que la nouvelle image soit reconnaissable, mais que ses couleurs soient bizarres/belles/psychédéliques…
- ☐ Éclaircir l'image (le contraire d'assombrir, traité à la partie 3) ; vérifier qu'une image éclaircie puis assombrie est l'image d'origine.
- ☐ …