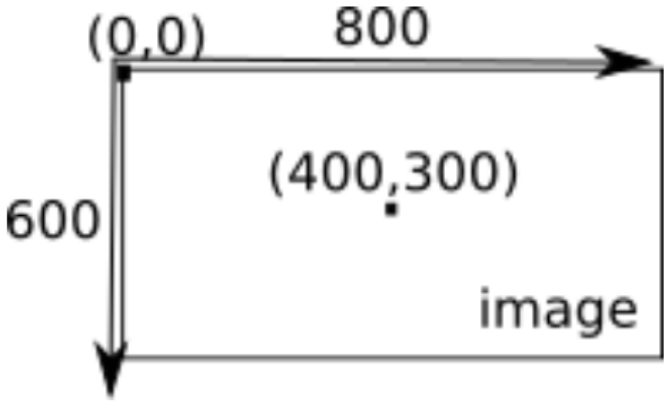
Projet : Module de manipulation d'image

Document basé sur les informations trouvé sur les sites :

- http://www.tangentex.com/TraitementImages.htm
- https://pixees.fr/informatiquelycee/n_site/snt_photo_transImg.html

I. Traitement d'image

Pour travailler sur les pixels, il faut travailler la valeur de la composante rouge R, la valeur de la composante vert V et la valeur de la composante bleu B pour un pixel donné.
Avant de commencer à écrire un programme qui nous permettra de travailler sur les pixels d'une image, il est nécessaire de préciser que chaque pixel a des coordonnées (x,y).



Comme vous pouvez le constater sur le schéma ci-dessus, le pixel de coordonnées (0,0) se trouve en haut à gauche de l'image.
Si l'image fait 800 pixels de large et 600 pixels de haut, le pixel ayant pour coordonnées (400,300) sera au milieu de l'image.
Dans un premier temps nous allons utiliser une simple photo de pomme pour faire nos premiers essais, ensuite, vous pourrez travailler avec l'image de votre choix. L'image de la pomme est téléchargeable ici :
http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/projet_manipulation_image/pomme.jpg
ou (si cela ne marche pas) :
http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/projet_manipulation_image/pomme.png

ATTENTION ! Cette image devra se trouver dans le même dossier que vos programmes Python.

A faire vous-même 1.

Saisissez et testez le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
r,v,b=img.getpixel((235,252))
print("canal rouge : ",r,"canal vert : ",v,"canal bleu : ",b)
```

Ce programme vous donne le canal rouge, le canal vert et le canal bleu du pixel de coordonnées (235, 252) de l'image `pomme.jpg`
Voici quelques explications :

- `img = Image.open("pomme.jpg")` Ici nous créons un objet image
- `r,v,b=img.getpixel((100,250))` Ici nous récupérer les niveaux de Rouge, de Vert et de Bleu.

A faire vous-même 2.

- Modifiez le programme du `À faire vous-même 1` pour qu'il affiche les valeurs du canal rouge, du canal vert et du canal bleu du pixel de coordonnées `(271,142)`, notez votre réponse.

  ……
  ……

A faire vous-même 3.

Il est possible de modifier les canaux RVB d'un pixel :

- Recopiez le script suivant afin de modifier la couleur du pixel du centre de vert en rouge.

```
from PIL import Image
img = Image.open("pomme.jpg")
img.putpixel((250,250), (255,0,0))
img.show()
```

A faire vous-même 4.

- Modifiez le programme du `À faire vous-même 3` afin de colorier le pixel de coordonnées (100,250) en bleu.

Modifiez un pixel c'est déjà bien, mais comment faire pour modifier plusieurs pixels ? La réponse est simple, nous allons utiliser des boucles `for`. Nous allons pouvoir balayer toute l'image et ne plus nous contenter de modifier les pixels un par un.

A faire vous-même 5.

Saisissez et testez le programme suivant

```
from PIL import Image
img = Image.open("pomme.jpg")
for y in range(200, 250):
    for x in range(260,295):
        img.putpixel((x,y), (255,0,0))
img.show()
```

Quelques commentaires sur ce programme :

- Les 2 boucles `for` nous permettent de parcourir un ensemble de pixels de l'image :
  - `for y in range(200, 250):`
    `for x in range(260,295):`

A faire vous-même 6.

Compliquons un peu la chose en modifiant tous les pixels de l'image. Nous voulons transformer toute l' image en niveaux de gris.
Il s' agit, pour chaque pixel, de calculer la moyenne des niveaux de rouge, vert et bleu et d' affecter cette valeur à ce pixel.
A noter : en Python pour avoir une division entière (le résultat est un entier), il faut utiliser l'opérateur // à la place de l'opérateur /

- Complétez et testez le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
...
...
for y in range(hauteur_image):
    for x in range(largeur_image):
        r,v,b=img.getpixel((x,y))
        gris = ...
        img.putpixel...
img.save("pomme_niveaux_gris.jpg")
img.show()
```

A faire vous-même 7.

- Après avoir fait quelques recherches sur le "négatif d'une image", modifiez votre programme pour qu' il donne le négatif de l'image image en niveau de gris.

A faire vous-même 8.

- En jouant avec les coordonnées x,y de chaque pixel, modifier le programme ci-dessous afin qu' il créé une nouvelle image qui est le symétrique suivant un axe vertical de la première.

```
from PIL import Image
img = Image.open("pomme.jpg")
...
…
```

```
img2=Image.new('RGB', (largeur_image, hauteur_image))
for y in range(hauteur_image):
    for x in range(largeur_image):
        r,v,b=img.getpixel((x,y))
        img2.putpixel...
img2.save("pomme_symetrique_v.jpg")
img.show()
img2.show()
```

A faire vous-même 9.

- En jouant avec les coordonnées x,y de chaque pixel, modifiez le programme afin qu' il créé une nouvelle image qui est le symétrique suivant un axe horizontal de la première.

A faire vous-même 10. (pour les costauds)

- En jouant avec les coordonnées x,y de chaque pixel, modifiez le programme afin qu' il créé une nouvelle image qui est la rotation de 90° dans le sens des aiguille d' une montre de la première.

A faire vous-même 11. (pour les costauds)

- Complétez le programme afin qu' il demande à l' utilisateur s' il veut faire une rotation de +90°, de +180°, de +270°
- Modifiez le programme afin qu' il créé une nouvelle image qui est la rotation demandée.

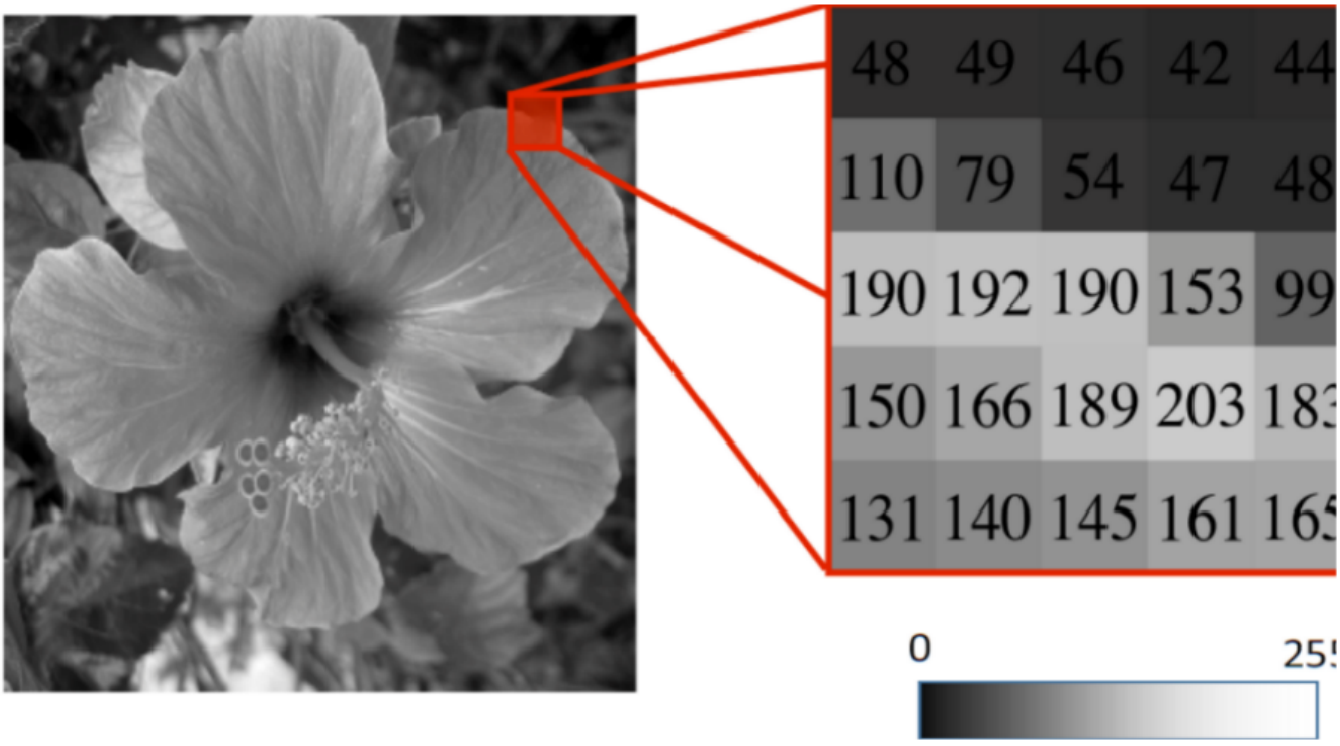
II. Détection des contours

A. Principes

Tu vas découvrir la détection de contours. Elle permet, par exemple, de repérer des discontinuités dans la profondeur de l'image, de repérer des zones dans une image, de reconnaître des objets

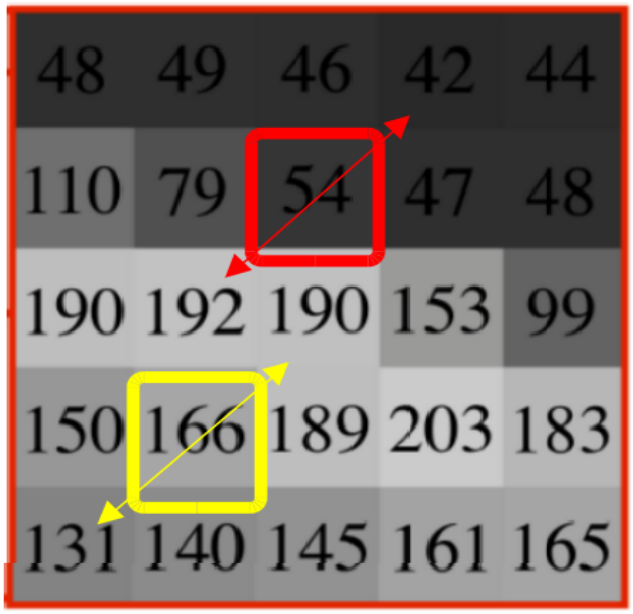
A faire vous-même 12.

Tracez le contour que tu visualises sur le carré de valeurs en niveaux de gris. Sélectionne les pixels.



Un contour se matérialise par une rupture (discontinuité) d'intensité dans l'image suivant une direction donnée.
Prenons un pixel P de coordonnées x,y : P(x,y). Si le niveau de gris des pixels voisins est différent du pixel P(x,y), alors on considérera que ce pixel fait partie d'un contour.
Pour cela il faut donc calculer une différence qui soit la plus significative. Il faudra donc prendre les 2 pixels qui sont diamétralement opposés au pixel central P(x,y). On va donc faire une soustraction entre les deux niveaux de gris des pixels voisins diamétralement opposés.

Quelle est cette différence pour le pixel P(x,y) encadré en rouge (et de valeur 54). Écris l'opération effectuée ?
Même question pour le pixel encadré en jaune (de valeur 166).
Parmi ces deux pixels, lequel ou lesquels semble(nt) appartenir à un éventuel contour et pourquoi ?
Pour éviter d'avoir un signe négatif dans la soustraction, il suffira d'élever au carré le résultat qui donnera toujours un résultat positif.



Nous n'avons pris en compte que 2 pixels voisins. Cela privilégie qu'une seule direction. Pour éviter cela on va donc prendre 4 pixels. Refais tous tes calculs de la même façon que précédemment pour les 2 pixels rouge et jaune, mais en prenant compte maintenant 4 pixels. Tu obtiendras 2 résultats (soustraction élevée au carré) que tu additionneras.

B. Programme python :

A faire vous-même 13.

- Complétez le programme suivant pour faire de la détection de contour comme expliqué ci-dessus :

```
from PIL import Image
img = Image.open("image_niveaux_gris.jpg")
...
valeur_seuil = 40
valeur_seuil = valeur_seuil*valeur_seuil+valeur_seuil*valeur_seuil
img2 = Image.new("RGB",(largeur_image,hauteur_image))
for y in range...
    for x in range...
        ...
        ...
        ...
        ...
        norme = ...
        if norme > valeur_seuil:
            img2.putpixel...
        else:
            img2.putpixel...
img2.show()
```

- Testez-le pour plusieurs valeurs de seuil

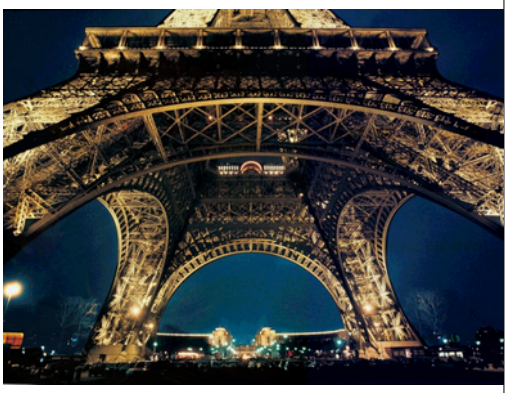
A faire vous-même 14. (pour les costauds)

- Écrivez un programme complet qui reprends tout ce que vous avez fait depuis le début
- Le corps de programme est un menu qui demande ce qu'il faut à l'utilisateur et qui crée les objets images
- Le reste est déporté dans des fonctions

III. VOL D'UN TABLEAU AU MUSEE D'ORSAY.

Mini-projets sur les images numériques : La stéganographie
Par groupe de 2 élèves
Fichiers à déposer sur pronote

A. Préambule



Suite au vol d'un tableau de Van Gogh au Musée d'ORSAY, les experts de la gendarmerie (IRCGN) ont intercepté le même message envoyé à différents receleurs connus. Ce message, signé par un mystérieux As Fantom, montre une photographie de ses vacances à Paris (ci-contre). Ils comptent sur votre perspicacité et vos compétences pour découvrir s'il n'y a pas anguille sous roche!!!

La stéganographie consiste à dissimuler un message dans un autre. Il s'agit plus précisément ici de voir comment on peut cacher une image dans une autre, en dissolvant en quelque sorte ses pixels dans ceux de l'image d'origine.

B. Pré-requis

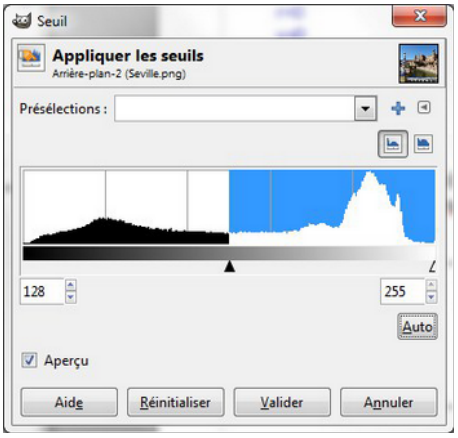
Dans l'écriture d'un nombre, les chiffres ont un poids croissant de la droite vers la gauche : Par exemple en binaire, le premier 1 (à gauche) de 11000101 représente $128=1x2^7$ alors que celui de droite représente $1=1x2^0$.

C. Votre premier programme sur la modification d'image.

A faire vous-même 15.

- Ecrivez une fonction noirEtBlanc qui est une version modifiée du programme niveauxDeGris fait précédemment.
- Pour passer d'une image couleur, à une image N&B. On pourra par exemple choisir un seuil au-dessous duquel on sera noir et au-dessus blanc (un peu comme la détection de contour).

On peut confronter l'image obtenue avec ce que donne GIMP (menu Couleurs > Seuil...), qui permet par ailleurs de faire varier le seuil de 0 à 255 à l'aide d'un curseur et d'avoir un aperçu en direct du résultat :



D. Première étape : Modifier l'image qui va cacher le message ou une autre image.

Si l'on met à zéro les 2, 3, voire 4 bits de poids faible (ceux de droite) des composantes RVB d'un pixel, l'image est-elle toujours de qualité acceptable ? L'idée étant de pouvoir les utiliser

pour insérer une autre image.
Mettre à zéro les deux bits de droite consiste à faire un ET logique avec `11111100`, c'est à dire 252.
Dans le cas des 4 bits de droite, on fait un ET avec `11110000`, c'est à dire 240.
En utilisant les deux boucles imbriquées vues précédemment, on obtient assez facilement le résultat voulu.

A faire vous-même 16.

- Créez une fonction que l'on nommera `mise_a_zero_bit_faible` qui va réaliser cette opération sur les composantes R, V, B de chaque pixel de l'image `pont-soupirs.bmp` :

http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/projet_manipulation_image/pont-soupirs.bmp

Originale

Modifiée

E. Seconde étape : la stéganographie proprement dite.

On peut envisager d'insérer une image 2 dans une image 1 de la façon suivante :

1. Considérons le pixel (i,j) de chaque image (images de même taille, 512 x 512 pixels ou autre).
2. La composante bleue de ce pixel vaut par exemple `00101111` pour l'image 1.
3. On met à 0 les 4 bits de droite (cf. étape 1) : on obtient `00100000`.

La composante bleue de ce pixel vaut par exemple `01110001` pour l'image 2. On fait un décalage de 4 bits vers la droite : `00000111`. En Python, les 4 bits de gauche sont alors mis à zéro, un ET logique avec `00001111` c'est à dire 15 n'est donc pas nécessaire ensuite.

Aide :

Python nous donne 6 opérateurs de base pour agir directement sur les bits:

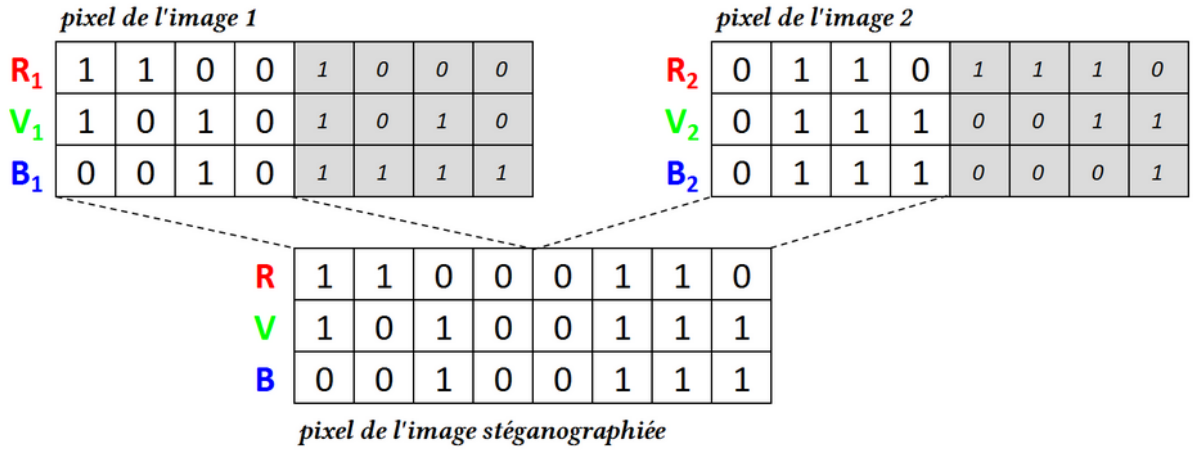
- `&` : et
- `|`: ou
- `^`: ou exclusif
- `~`: inversion des bits du nombre situé à droite
- `>>`: décalage d'un bit à droite (correspond à une division par 2)
- `<<`: décalage d'un bit à gauche (correspond à une multiplication par 2)

A faire vous-même 17.

- Créez une fonction que l'on nommera `decalage4BitsVersLaDroite` qui va réaliser cette opération sur l'image tableau.bmp :

http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/projet_manipulation_image/tableau.bmp

La composante bleue du pixel (i,j) de l'image stéganographiée est obtenue par un OU entre les deux précédentes : `00100111`.
On fait de même pour les composantes rouge et verte.



L'image 2 étant codée dans les bits de poids faibles, elle devrait (presque) ne plus être visible.

A faire vous-même 18.

- Créez une fonction `stegano` qui s' appuie sur les 2 fonctions précédentes pour arriver au résultat suivant :

image1.bmp + image2.bmp = image3.bmp

F. Troisième étape : La mission confiée par les experts.

A faire vous-même 19. (pour les costauds)

- Proposez une fonction `defaireImage` qui permette de sortir les images 1 et 2 de l'image 3. La tester, puis l'appliquer sur l'image `souvenir-vacances.bmp` fournie par les experts :

http://ninoo.fr/LC/1ere_NSI/seq1_le_langage_informatique_python/projet_manipulation_image/souvenir-vacances.bmp

1ère NSI **GRILLE D'EVALUATION :** **Noms :** **Note : /18**	Rien d'écrit	Ne fonctionne pas	Fonctionne imparfaitement	Fonctionne avec beaucoup d'aide	Fonctionne avec aide	Fonctionne sans aide	Fonctionne de manière optimale et avec les commentaires
Implication dans le travail préparatoire (niveauDeGris, negatif, symetrique)							
Implication dans le travail préparatoire II (ContourImage)							
noirEtBlanc							
miseAZeroBitFaible							
decalage4BitsVersLaDroite							
Stegano							
DefaireImage (bonus)							