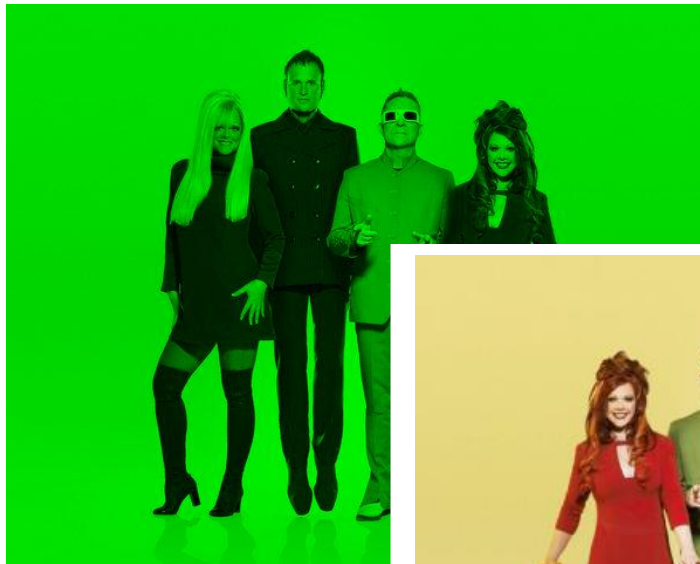


# N.S.I.

## Comment modifier le contenu d'une image ?



L. Riss

Lycée R.Doisneau - Corbeil

N.S.I.

## Notre problématique : **comment modifier le contenu d'une image ( régler la luminosité)et extraire les informations de taille, d'offset, de résolution, etc...**

On souhaite développer un programme capable de régler la luminosité des images scannées.

*Nous connaissons déjà le contenu d'une image Bitmap ( nous savons déjà retrouver les informations de taille, résolution, etc grâce à un éditeur hexadécimal)*

L'algorithme du 1er programme sera le suivant :

- Ranger les octets du fichier image dans une liste
- Vérifier qu'il s'agit bien d'un fichier BITMAP
- Aller chercher la valeur de la taille et de l'offset du fichier
- Exprimer la taille en koctet et l'offset en octet et les afficher à l'écran
- Augmenter la luminosité de l'image

### **1/ Ranger les octets du fichier image dans une liste de valeur que nous pourrions facilement manipuler**

Le programme permettant ceci est donné en annexe 1. Nous n'allons pas tout retaper ! Nous allons copier le fichier source ainsi que quelques images BMP à modifier dans notre espace de travail .

➔ Télécharger l'archive *dossier-eleves-Image.zip* qui se trouve sur <http://ressource.elec.free.fr/softs.html> , copier le contenu de l'archive dans un répertoire que vous nommerez «python images » dans votre espace de travail.

➔ ouvrez **Spyder ou Pycharm**

➔ ouvrez le fichier source « Images.py »

Exécutez et testez le programme.

Pour l'instant, le programme affiche à l'écran le 1<sup>er</sup> octet du fichier

Proposer un complément de programme permettant l'affichage aussi du second octet et testez votre solution.

**2/ vérifier que le fichier est un fichier BITMAP**

Rappelez à quoi peut on savoir que le fichier est bien un fichier BITMAP (aidez vous du document BMP où est expliqué le rôle de chaque octet)

Proposez alors un algorithme en « si...alors...sinon... » permettant d'effectuer ce test.

Codez en Python votre solution et testez.

*rappel : un « si...alors... sinon... » se code ainsi*

if (conditions) :

instructions}

else :

instructions

*Quelques tests possible : égalité : "=", inégalité : "<, >, <=, >= », différence : « != »*

*Pour mettre plusieurs conditions : et : « and » , ou : « or »*

*exemple de codage : le programme ci-dessous affiche à l' écran « ok » si « a »=5 et si b= « 6 » :*

```
if (a == 5) and (b==6) :
```

```
    print ("ok")
```

```
else :
```

```
    print ("pas ok")
```

**3/ trouver la valeur de la taille du fichier.**

Retrouver dans le document « BMP » que vous possédez déjà quels sont les 4 octets qui codent la taille du fichier. \_\_\_\_\_

On rappelle que le poids des octets est inversé (l'octet de poids fort est l'octet 5, l'octet de poids faible est l'octet 2)

Les valeurs de ces 4 octets se retrouvent dans notre liste ! :o) . Nous n'avons plus qu'à aller les chercher. Par exemple, pour l'octet 5 :

**octet5 = ListePixels [5]**

La valeur totale de ces quatre octets va se calculer de la manière suivante :

**Taillefichier = octet2 + (256\*octet3) + (256\*256\*octet4) + (256\*256\*256\*octet5)**

*Votre travail :* développez la partie du programme qui permette cet algorithme :

-> déclarer les 4 variables de type entiers (octet2, octet3, octet4, octet5) en leur assignant leur valeur correspondante

-> déterminer la taille du fichier (en utilisant la formule donnée)

-> afficher le résultat à l'écran (en soignant la présentation)

**4/ trouver la valeur de l'offset du fichier.**

On rappelle que l'offset décrit à partir de quel octet les informations relatives aux couleurs des pixels commence (aider vous encore ici du document « BMP »).

-> développez le bout de programme permettant de calculer puis d'afficher l'offset de l'image

**5/ réglage de la luminosité**

Si on souhaite augmenter la luminosité d'une image, il suffit d'augmenter la valeur de chaque couleur du pixel.

On rappelle qu'un pixel est composé de 3 octets codant respectivement pour le bleu, le vert et le rouge. Chacune de ces couleurs étant codée sur un octet, sa valeur se situe entre 0 et 255.

Il va falloir traiter l'ensemble des pixels. Nous allons avoir besoin de créer une boucle répondant à cet algorithme (une boucle « for ») (pour cet exemple, on augmentera de 100 chaque valeurs) :

***pour chacune des valeurs du tableau, en commençant à l'offset et en s'arrêtant à la fin du tableau :***

***- valeur = valeur +100***

***- si valeur > 255 alors valeur = 255 (on limite à 255 la valeur)***

➔ Complétez le programme pour qu'il augmente de 100 la luminosité de l'image.

Aide :

Rappel : programme avec une boucle  
« for » :

```
for i in range (0,4) :
    a = tableau[i]
    a = a*2
    tableau[i] = a
```

*Ce programme multiplie par 2 chaque valeur du tableau.*

*Détails:*

*pour chaque valeur de i<4 :*

*a = la valeur de tableau indexé par i*

*on multiplie « a » par 2*

*on range la nouvelle valeur de « a » dans tableau*

➔ Complétez votre programme pour que ca soit à l'utilisateur du programme de choisir de combien il souhaite changer la luminosité (de la même manière qu'on propose le nom du fichier à modifier

**6/ modification des couleurs**

➔ Proposez un bout de programme qui élimine le bleu des images

➔ Proposez un bout de programme qui augmente le vert des images

## Annexe 1

[illegible]