Les images au format PGM

Le format PGM (portable graymap file format) est utilisé pour des images en niveau de gris. C'est un format d'image matricielle, sans compression, assez peu utilisé mais qui présente l'intérêt d'être facilement manipulable.

Conversion d'une image JPEG dans le format PGM avec le logiciel GIMP Commencez par copier sur votre Bureau une image en couleur de format quelconque (JPEG, PNG, BMP, GIF, PPM...). Par exemple :

Avec le logiciel de traitement d'images GIMP, nous allons convertir cette image dans le format PGM : Ouvrir le fichier avec GIMP Fichier → Enregistrer sous Sélectionner le type de fichier (selon l'extension) → Image PGM Exporter Formatage des données → Brut Enregistrer

Renommez ce fichier : `image.pgm` Télécharger le fichier image.pgm

Un exemple de traitement : l'inversion de couleurs

Le script `inversion_image_pgm.py` permet d'inverser les couleurs d'une image. On obtient ainsi le "négatif" :

```
# script inversion_image_pgm.py
import imghdr

print("Inversion d'une image PGM en mode binaire à 256 niveaux de gris\n"))

NomFichierSource = 'image.pgm'
NomFichierDestination = 'imageInverse.pgm'

print('Fichier source :',NomFichierSource))
print('Fichier destination :',NomFichierDestination))

def Inversion(octet):
    # cette fonction fait une inversion du niveau de gris
    # 0 (noir)    -> 255 (blanc)
    # 255 (blanc) -> 0 (noir)
    return 255-octet

if imghdr.what(NomFichierSource)=='pgm':    # test du format de l'image
    FichierSource = open(NomFichierSource,'rb')
    TailleFichier = len(FichierSource.read())
    print('\nTaille du fichier (en octets) :',TailleFichier))

    Largeur = int(input('Largeur de l\'image (en pixels) ? '))
    Hauteur = int(input('Hauteur de l\'image (en pixels) ? '))
    NbPixel = Largeur*Hauteur

    TailleEntete = TailleFichier - Largeur*Hauteur

    FichierSource.seek(0)

    # extraction de l'en-tête
    # la variable Entete est une chaîne de caractères (type str)
    Entete = FichierSource.read(TailleEntete)

    # extraction des données
    # Data est une liste de nombre entiers (type list)
    # la fonction ord() retourne le contenu d'un octet sous forme d'un entier
    # ord('\xf3') -> 243
    Data = [ord(i) for i in FichierSource.read()]   # compréhension de listes

    FichierSource.close()

    if NbPixel == len(Data):
        print('Nombre de pixels :',Largeur*Hauteur)
        print("Nombre d'octets de données :",len(Data))
        print("Taille de l'en-tête :",TailleEntete)
```

```
        FichierDestination = open(NomFichierDestination,'wb')
        # écriture de l'en-tête du fichier destination
        FichierDestination.write(Entete)

        # écriture des données du fichier destination
        for i in Data:
            # la fonction chr() fait le contraire de la fonction ord()
            # chr(243) -> '\xf3'
            FichierDestination.write(chr(Inversion(i)))

        FichierDestination.close()
        print('Travail terminé !')

    else:
        print('Erreur dans la saisie des données !')

else:
    print("Ce n'est pas une image PGM !")
```

```
>>>
Inversion d'une image PGM en mode binaire à 256 niveaux de gris

Fichier source : image.pgm
Fichier destination : imageInverse.pgm

Taille du fichier (en octets) : 153654
Largeur de l'image (en pixels) ? 480
Hauteur de l'image (en pixels) ? 320
Nombre de pixels : 153600
Nombre d'octets de données : 153600
Taille de l'en-tête : 54
Travail terminé !
```

Le script crée le fichier `imageInverse.pgm` dans le répertoire courant. Affichons l'en-tête des fichiers image PGM :

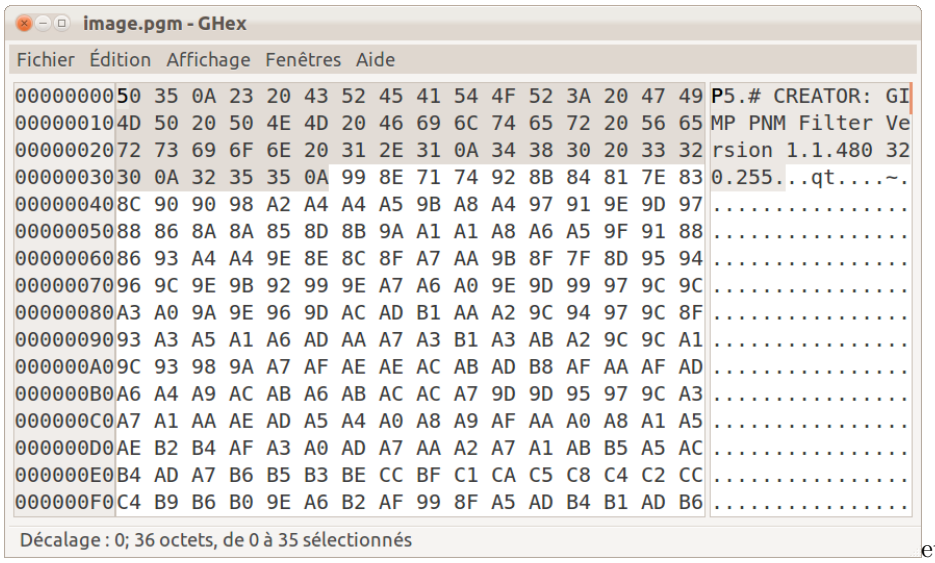
```
>>> print(Entete.encode('hex'))  # contenu de l'en-tête en hexadécimal
50350a232043524541544f523a2047494d5020504e4d2046696c746572205665727369f6e20312e310a3438302
>>> print(Entete)      # contenu de l'en-tête en chaîne de caractères
P5
# CREATOR: GIMP PNM Filter Version 1.1
480 320
255

>>>
```

L'en-tête contient en particulier la largeur et la hauteur de l'image (480 pixels

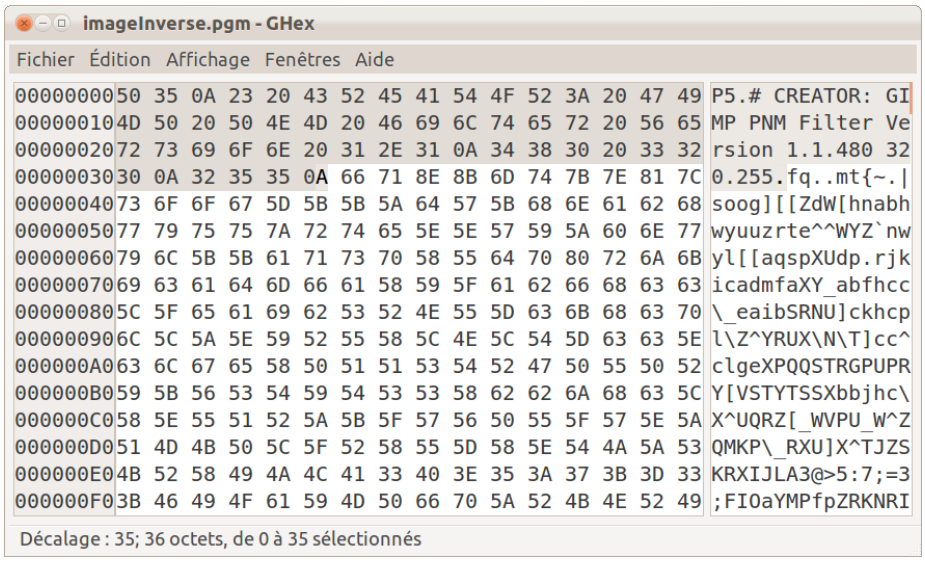
x 320 pixels) et le nombre 255 (soit 256 niveaux de gris). La couleur d'un pixel est codée sur un octet. On va du noir (0x00) au blanc (0xff) en passant par tous les niveaux de gris.

```
>>> print(Data[0],hex(Data[0]))  # premier octet : pixel en haut à gauche de l'image
153 0x99
>>> print(Data[1],hex(Data[1]))    # deuxième octet : pixel à droite du précédent
142 0x8e
>>> print(hex(Data[479]))  # 480ème octet : pixel en haut à droite
0xbf
>>> print(hex(Data[480]))  # 481ème octet : pixel de la deuxième ligne à gauche
0x92
>>> print(hex(Data[153599]))   # dernier octet : pixel en bas à droite de l'image (320ème li
0x25
```

Avec un éditeur hexadécimal, observons le contenu du fichier `image.pgm` :



et comparons avec le contenu du fichier `imageInverse.pgm` :



Inversion de couleurs avec le logiciel GIMP Il suffit de faire : Couleurs → Inverser

La librairie PIL (Python Imaging Library)

Cette librairie spécialisée de Python fournit des outils de traitement d'images.

Installation sur tablette ou smartphone sous Android Dans Google Play, rechercher puis installer l'application QPython - Python for Android.

Installation sur PC sous Windows ou GNU/Linux PIL n'est pas présente par défaut. Il faut donc la télécharger et l'installer. Télécharger la librairie PIL

Traitement d'images avec la librairie PIL Le script suivant permet de convertir une photo en couleur au format JPEG en une image en niveau de gris au format PGM, puis de la transposer (retournement, miroir). Les résultats sont affichés dans des fenêtres graphiques indépendantes et enregistrés dans des

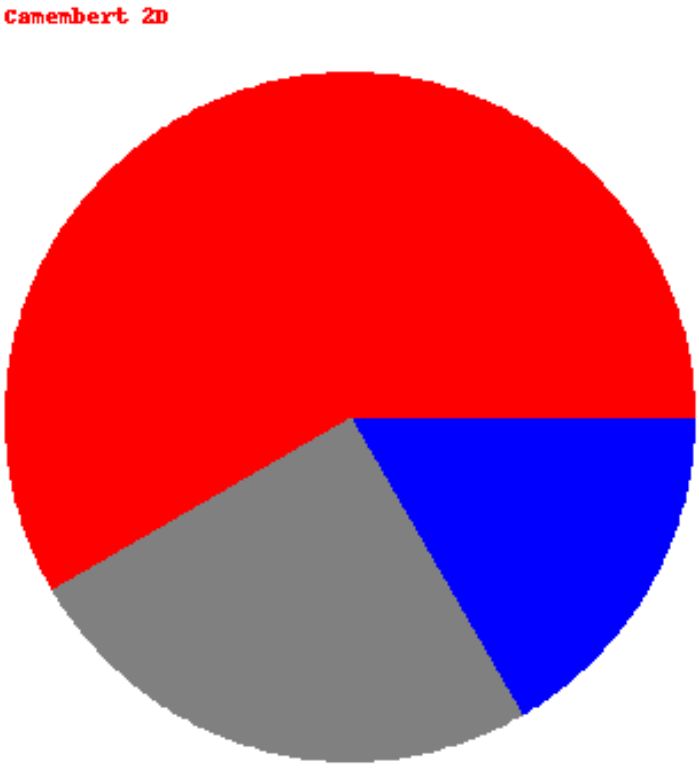
fichiers images :

```
# Python 2.7
# PIL 1.1.7
# importation du module Image de la librairie PIL
from PIL import Image

# ouverture de l'image
img = Image.open('photo.jpg')
# affichage de l'image
img.show()
# affichage de la taille de l'image (en pixels)
print(img.size)
# conversion au format PPM (en couleur) et enregistrement de l'image
img.save('photo.ppm','PPM')
img.show()
# conversion en niveau de gris (pour obtenir le format PGM)
img0 = img.convert('L')
# enregistrement dans le fichier image.pgm
img0.save('image.pgm')
img0.show()

# retournement de l'image
```

```
img1 = img0.rotate(180)
# affichage et enregistrement de l'image retournée
img1.show()
img1.save('image_retourne.pgm')

# miroir horizontal
img2 = img0.transpose(Image.FLIP_LEFT_RIGHT)
img2.show()
img2.save('image_miroir_horizontal.pgm')

# miroir vertical
img3 = img0.transpose(Image.FLIP_TOP_BOTTOM)
img3.show()
img3.save('image_miroir_vertical.pgm')
```

```
>>>
(480, 320)
```

Dessiner avec la librairie PIL Le module ImageDraw de la librairie PIL permet de créer des formes géométriques simples dans une image : ligne, cercle, rectangle, texte... Voici un script qui fabrique l'image d'un camenbert 2D (au format PNG) :




```
# script camenbert.py
# Python 2.7
# PIL 1.1.7
from PIL import Image,ImageDraw

# création d'une image 400x400 (fond blanc)
img = Image.new("RGB",(400,400),"#FFFFFF")

# création d'un objet Draw
dessin = ImageDraw.Draw(img)

# dessine un arc partiel et le remplit
dessin.pieslice((50,50,350,350),0,60,fill="blue")
dessin.pieslice((50,50,350,350),60,150,fill="gray")
dessin.pieslice((50,50,350,350),150,360,fill="red")
```

```
# dessine du texte
dessin.text((50,20),"Camenbert 2D",fill="red")

img.save("camenbert.png","PNG")
img.show()
```

Exercices

Exercice 10.1 *

1. Reprendre le script `inversion_image_pgm.py` de manière à réduire l'image initiale à deux couleurs (noir et blanc) en utilisant un seuil :

2. Faire la même chose avec le logiciel GIMP.

Exercice 10.2 **

1. Reprendre le script `inversion_image_pgm.py` de façon à retourner l'image (rotation à 180 degrés) :

2)
Faire la même chose avec le logiciel GIMP.

Exercice 10.3 ** Reprendre le script `inversion_image_pgm.py` afin de faire une transformation miroir horizontal :

2)
Faire la même chose avec le logiciel GIMP.

Exercice 10.4 ** Reprendre le script `inversion_image_pgm.py` afin de faire une transformation miroir vertical :

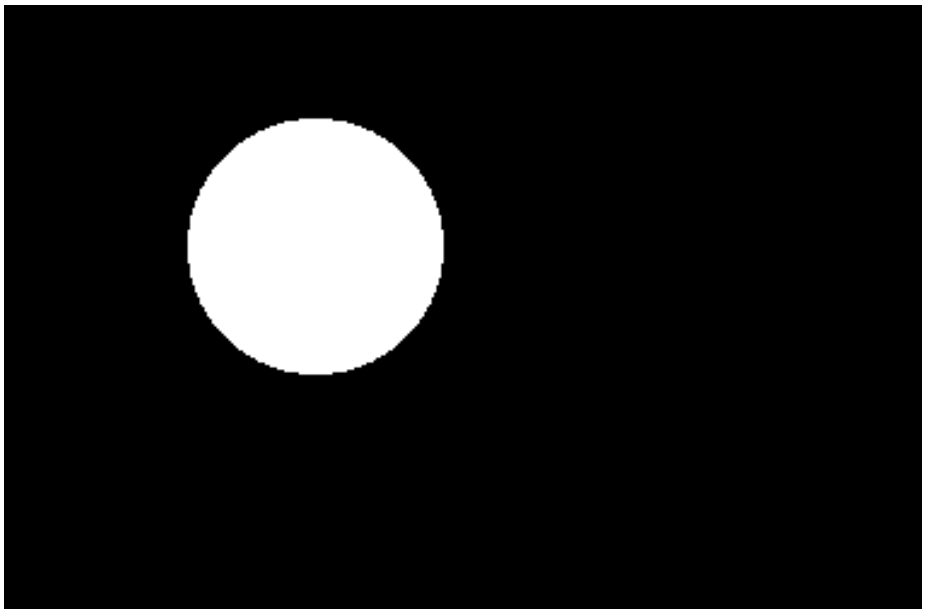
2)

Faire la même chose avec le logiciel GIMP.

Exercice 10.5 * Redimensionnement d'une image JPEG A l'aide de la méthode resize() du module Image de la librairie PIL, redimensionner une image JPEG. Par exemple :

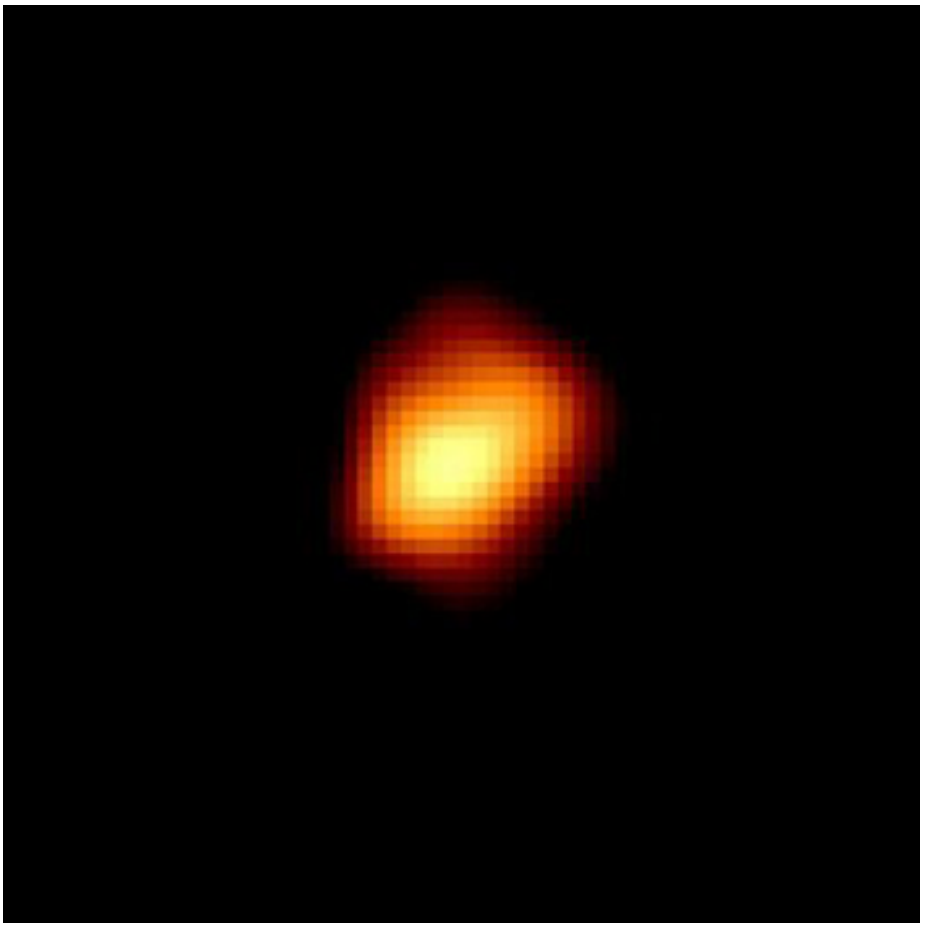
```
>>>
Redimensionnement d'une image JPEG

Nom de l'image originale ? photo.jpg
Taille de l'image originale : (largeur, hauteur) = (1944, 2592)

Nouvelle taille :
Hauteur en pixels ? 800
Largeur en pixels : 600

Enregistrement de l'image redimensionnée
Nom de l'image redimensionnée ? photo_1.jpg
>>>
```

Exercice 10.6 ** Calcul du diamètre d'un disque 1) A l'aide de la méthode getpixel() du module Image de la librairie PIL, estimer le diamètre d'un disque blanc d'une image à fond noir. getpixel((x,y)) retourne la couleur du pixel de coordonnées (x,y). Par exemple :

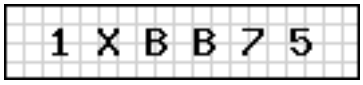


```
>>>
Nom du fichier ? disque.png

Largeur en pixels :  380
Hauteur en pixels :  250

Nombre de pixels (blanc) :  8820
Position du centre : x = 129.0 y = 100.0
Diamètre en pixels : 105.971565925
>>>
```

2) Application en astronomie Estimer les dimensions de cette étoile rouge géante

(image au format JPEG) : 

Exercice 10.7 ** Captcha Avec le module ImageDraw de la librairie PIL et en s'inspirant du script `camenbert.py`, créer un générateur de captcha :



Exercice 10.8 ** Ecrire un script qui permet d'extraire de l'en-tête d'un fichier PGM :

- la largeur de l'image (en pixels)
- la hauteur de l'image (en pixels)
- le nombre de niveaux de gris
- les commentaires

Exercice 10.9 *** L'image suivante contient un code secret. Saurez-vous le décrypter ?

Indice n°1 : Python n'est pas nécessaire.

Indice n°2 :

Télécharger le fichier enigmaCodeSecret.pgmRemarque : Au jour d'aujourd'hui (13/04/2017), 3 personnes ont trouvé (Jacques B., Dick O. et Le B.) !

Webographie

- Le format PGM
- Site officiel du logiciel libre et gratuit GIMP
- Editeur hexadécimal
- Eduscol : notion d'image numérique
- Documentation sur la librairie PIL (Python Imaging Library)
- Documentation sur le module Image de la librairie PIL
- Documentation sur le module ImageDraw de la librairie PIL
- Des exemples de traitement d'image avec le module Image de la librairie PIL