Initiation au traitement d'images

Notions abordées : listes, boucles for, appels de fonctions, bibliothèques de fonctions, chaînes de caractères, tuples, fichiers ascii.

Les formats pixmap

Dans ce projet, nous utiliserons des images dans des formats très simples : les formats **pixmap**. Ces images sont reconnues nativement par Linux et OpenOffice, et sont lisibles sous Windows avec Gimp. Les extensions des formats pixmap en ascii sont :

- .pbm pour les images noir et blanc codée en caractères ASCII (P1),
- .pgm pour les images en niveau de gris codée en caractères ASCII (P2),
- .ppm pour les images en couleurs RGB codée en caractères ASCII (P3).

Remarque : certaines applications utilisent l'extension **.pnm**

Par exemple, voici trois images en noir et blanc (P1), en niveaux de gris (P2) et RVB (P3):

```
P1
10 10
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

```
P2
10 10
10
1 1 1 1 1 1 1 9 9 9
2 2 2 2 2 2 2 2 9 9
3 3 3 3 3 3 3 9 2 9
4 4 4 4 4 4 9 3 2 1
5 5 5 5 5 9 4 3 2 1
6 6 6 6 9 5 4 3 2 1
7 7 7 9 6 5 4 3 2 1
8 8 9 7 6 5 4 3 2 1
9 9 8 7 6 5 4 3 2 1
9 9 8 7 6 5 4 3 2 1
```

```
P3
# 3 colonnes et 2 lignes,
3 2
# ayant 255 pour valeur maximum de codage
255
255 0 0
0 255 0
0 0 255
255 255 0
255 255 255
0 0 0
```

Pour en savoir plus sur les images pixmap : https://fr.wikipedia.org/wiki/Portable_pixmap

Méthode pour créer une image pixmap avec GIMP :

- Faire **exporter sous…** dans le menu **fichier**.
- Choisir l'extension appropriée (.ppm, .pgm, .pbm)
- Choisir l'**option ASCII** après avoir cliqué sur le bouton exporter

Bibliothèque « boite_a_outils_nsi.py »

Une bibliothèque de fonctions « boite_a_outils_nsi.py » a été créée pour les besoins de ce mini projet. Les fonctions de cette boîte à outils permettent de créer, ouvrir, et sauvegarder des images simples aux format « PBM », « PGM » ou « PPM » (en ASCII).

Les fonctions utiles de cette bibliothèque sont :

lire_fichier_image(nomFichier) nomFichier est un fichier .pbm, .pgm ou .ppm

la syntaxe est : `maMatrice = lire_fichier_image(nomFichier)`
maMatrice est une liste de liste de nombres ou de tuples :
pour les fichiers P1, les nombres sont 0 ou 1.
Pour les fichiers P2, les nombres peuvent aller de 0 à valeur max
Pour les fichiers P3, les tuples peuvent aller de (0,0,0) à (valeur max, valeur max, valeur max)

Exemples :

image P1 de dimension 5 x 4
```
[[1,1,1,1,1],[1,0,0,0,1],[1,1,0,1,1],[1,0,0,0,1]]
```

image P2 de dimension 4 x 3 et de valeur max 255
```
[[200,120,5,6],[80,102,0,255],[41,125,241,114]]
```

image P3 de dimension 3 x 2 et de valeur max 10
```
[[(2,1,5),(7,10,9),(2,4,8)],[(5,4,5),(7,5,0),(5,7,6)]]
```

Si maMatrice contient une image pixmap :

- La hauteur de l'image est donnée par len(maMatrice)
- La largeur de l'image est donnée par len(maMatrice[0])

def lire_valMax(nomFichier) nomFichier est un fichier .pbm, .pgm ou .ppm.
Retourne la valeur max pour un fichier .pgm ou .ppm
Retourne None pour un fichier .pbm

creer_fichier_image_P1(t,nom) t est une liste de pixel, nom le nom du fichier à créer l'extension « .pbm » est mise automatiquement.

creer_fichier_image_P2(t,nom,max) t est une liste de pixel, nom le nom du fichier à créer max est la valeur maximale de codage (nbre de niveaux de gris). L'extension « .pgm » est mise automatiquement.

creer_fichier_image_P3(t,nom,max) t est une liste de pixel, nom le nom du fichier à créer max est la valeur maximale de codage (nbre de niveaux de gris). L'extension « .pgm » est mise automatiquement.

exemple d'application 1 : duplication d'image p3

```
#!/usr/bin/env python3
from boite_a_outils_nsi import *
from string import *

nom_image='p3-image' # nom du fichier image a ouvrir
extension='.ppm' # nom de l'extension
image = nom_image + extension # nom complet du fichier à ouvrir
matrice_source = lire_fichier_image(image) # importation de l'image
h=len(matrice_source) # hauteur de l'image en pixel
l=len(matrice_source[0]) # largeur de l'image
matrice_cible=creer_matrice(l,h) # création d'une matrice de dimensions
                                 identiques

# copie des tuples de la matrice source dans la matrice cible
for i in range (h): # liste qui contient les lignes
    for j in range (l): # liste qui contient les pixels
        matrice_cible[i][j] = matrice_source[i][j] #duplication

# creation du fichier cible
image_cible = nom_image + '_diplication' # création du nom
creer_fichier_image_P3(matrice_cible,image_cible,255) # enregistrement
```

exemple d'application 2 : rotation d'image p3 de 180°

```
#!/usr/bin/env python3
from boite_a_outils_nsi import *
from string import *

nom_image='p3-image' # nom du fichier image a ouvrir
extension='.ppm' # nom de l'extension
image = nom_image + extension # nom complet du fichier à ouvrir
matrice_source = lire_fichier_image(image) # importation de l'image
h=len(matrice_source) # hauteur de l'image en pixel
l=len(matrice_source[0]) # largeur de l'image
matrice_cible=creer_matrice(l,h) # création d'une matrice de dimensions
                                 identiques

# copie des tuples de la matrice source dans la matrice cible
for i in range (h): # liste qui contient les lignes
    for j in range (l): # liste qui contient les pixels
        matrice_cible[i][j] = matrice_source[h-i-1][l-j-1] #rotation

# creation du fichier cible
image_cible = nom_image + '_rotation' # création du nom
creer_fichier_image_P3(matrice_cible,image_cible,255) # enregistrement
```

mini-projet : programmes de transformations

A partir de l'image de base, réaliser les transformations suivantes :