Jouer avec des images

Lire / écrire dans un fichier

Cette partie est une reprise du notebook "compléments" ; si vous l'avez déjà fait vous pouvez néanmoins le reprendre.
Commencez par créer un simple fichier texte, avec le bloc-notes, ou Text Edit, contenant deux lignes séparées par un return. Enregistrez le dans le même dossier que ce TD, en format texte, sous le nom "a_modifier.txt". Créez également un fichier vide "copie.txt", dans le même dossier. Le programme suivant vous montre la lecture et l'écriture dans ce fichier.

In [ ]:
```
# Ouverture du fichier a lire et ecrire (mode "r+" read write)

nomFichier = "a_modifier.txt"
fichierInit = open(nomFichier, "r+")

# ouverture ou creation d'un nouveau fichier s'il n'existe pas l
ors du "open"
fichierCopie = open ("copie.txt", "r+")

# affichage du contenu du fichier par ligne
print("affichage ligne par ligne")
for line in fichierInit:
    print(line)

# repositionnement au debut du fichier et lecture d'une ligne
fichierInit.seek(0)   # la méthode "seek" repositionne au debut
du fichier
print("relecture de la première ligne")
print(fichierInit.readline())

# Recopie caractere par caractere dans le fichier "copie.txt"
# On pourrait aussi bien faire une copie par ligne, ou globale (
plus rapide)
fichierInit.seek(0)
for lettre in fichierInit:
    fichierCopie.write(lettre)

fichierInit.seek(0,2)   # positionnement a la fin du fichier
                        # Oieme bit avant la fin (2)
fichierInit.write(str(42))  # on ne peut ecrire que des chaines
de caracteres

print("lecture du fichier intial")
fichierInit.seek(0)
print(fichierInit.read())

print("lecture du fichier copie")
fichierCopie.seek(0)
print(fichierCopie.read())


fichierInit.close() # Fermeture du fichier source (essentiel)
fichierCopie.close() # Fermeture du fichier copie (essentiel)
```

Remarque : chemin relatif, chemin absolu

Dans l'instruction `nomFichier = "a_modifier.txt"`, le chemin d'accès est relatif. C'est à dire que le programme considère que le chemin d'accès part du répertoire courant. Si le fichier est dans un sous-repertoire "documents", on tape `nomFichier = "documents\a_modifier.txt"`. Un chemin absolu part de la **racine** du système : `nomFichier = "C:\utilisateur\documents\a_modifier.txt"`. Lors des tests, utilisez des chemins relatifs.
En effet, Il peut se produire divers problèmes avec les chemins absolus. Le caractère "\", dit backslash ou antislash, est un caractère spécial dans de nombreux langages, dont Python. par exemple, l'instruction `print("\"")` permet d'imprimer juste un guillemet. Si un programme comportant un chemin d'accès absolu renvoie une erreur sur le chemin, deux possibilités s'offrent à vous : taper `\\` au lieu de `\`, ou bien taper `/` au lieu de `\` (ce qui correspond aux chemins sous Linux). Pour l'anecdote, il arrive qu'aucune de ces deux possibilités ne fonctionne, et il est arrivé aussi que sur deux machines du lycée a priori clonées, donc avec le même Python, les comportements se sont avérés différents.

Lire et modifier un fichier image

Avec un éditeur d'images

Lancer Gimp. Prendre une image de votre choix, de préférence de petite taille. L'exporter en format pbm, ppm, et pgm (respectivement portable bitmap file format en noir et blanc, portabke pixmap file format, et petit gros mou file format, en niveaux de gris). Le faire dans les deux possibilités : format brut et format ASCII (ce qui vous fait donc 6 images au total). Sauvegarder vos images, dans un dossier où vous garderez ces originaux. En effet, on va les modifier de nombreuses fois : je vous conseille de ne travailler que sur des copies.

Avec un éditeur hexadécimal

Copier les deux images en brut et en ASCII en pgm dans un dossier de travail.

Lancer ensuite l'éditeur hexadécimal Frhed.
Ouvrir l'image en format brut dans Frhed. Les premiers caractères sont lisibles, ils sont du type :
P5 # *commentaires* nombre1 nombre2 255 (cela peut être un autre nombre).
P6 est le nombre magique indiquant le format, nombre1 et nombre2 donnent respextivement la largeur et la hauteur de l'image (le nombre de colonnes puis le nombre de lignes). Enfin, 255 est la profondeur de codage du gris, qui admet 256 nuances (avec le 0). Ces lignes forment l'en-tête du fichier, elles contiennent les métadonnées de l'image. Les métadonnnées peuvent être bien plus conséquentes ; on peut par exemple trouver des commentaires, commençant par #, indiquant le propriétaire de l'image, des données GPS, etc... La suite du fichier est codée en bytes (ici des octets) et n'est que partiellement affichable.

Faire une copie de l'image en format ASCII, et l'ouvrir dans Frhed.
Ne pas modifier l'en-tête (vous riqueriez d'avoir des difficultés de lecture par la suite).
Vous remarquez que les données numériques sont séparées par un point. Ce "point" est un séparateur de données, il est fondamental de bien le repérer lors du traitement de l'image. En effet, il ne doit surtout pas être modifié, puisqu'il prévient le programme que l'on passe d'une donnée à la suivante. Promenez-vous sur les premiers caractères du fichiers. Comparez la partie de droite et celle de gauche : les points sont-ils toujours codés par le même nombre ?
Le script ci-dessous permet de tester les différents codages ASCII et le caractère correspondant. Modifiez-le pour trouver le code correspondant au séparateur de données, gardez-le en mémoire. Que fait ce caractère bien particulier ?

In [ ]:
```
asc = int("2f",16)      # si le code ASCII comporte des lettres
entre a et f
                        # c'est de l'hexadecimal, ici vous avez
la méthode pour
                        # trouver le caractère ASCII de code 2f
ou 2F
print("code ASCII ",asc," caractere ",chr(asc))
print('\n')             # Que fait ce caractère ?
asc = 97
print("code ASCII ",asc," caractere ",chr(asc))
```

Dans la partie où l'on peut lire l'en-tête, modifiez les premières données, **toujours dans la copie**. Mettez soit 0 soit 255 à la place des 20 premières données (après le 255 donc). Enregistrez et fermez le fichier. Rouvrez l'image sous Gimp. Agrandissez en haut à gauche, que constatez-vous ?

Avec Python

Les programmes que l'on va faire pour traiter une image sous Python doivent:

- ouvrir le fichier source
- ouvrir le fichier où l'on va sauver notre image modifiée. Si ce fichier n'existe pas, Python le crée lors de l'ouverture
- lire l'en-tête du fichier source et le garder en mémoire
- lire les données de l'image source
- traiter ces données
- écrire l'entête dans le fichier modifié
- écrire les données modifiées
- fermer les deux fichiers

On a déjà vu dans la première partie de ce TD l'ouverture et la fermeture des fichiers, ainsi que la lecture et l'écriture.

On va affiner un peu la lecture et l'écriture des données avant de passer au traitement global d'une image.

Le programme suivant affiche l'en-tête de l'image, puis les 20 premières données. Il travaille sur l'image originelle, en format pgm ASCII.

In [ ]:
```
print ("Affichage des premières données d'une image en format pg
m")

# La variable suivante contient le chemin d'accès à l'image init
iale.
# Bien sur, ces noms sont à modifier si vous utilisez une autre
image.

# chemins absolus :
#nomFichierSource = "C:Dudule\documents\imageb.pgm"
#nomFichierDestination = "C:Dudule\documents\essaitransforme.pgm
"

nomFichierSource = "lyceejj.pgm"

print ("Fichier source :",nomFichierSource)

# ouverture du fichier source en format ASCII
fichierSource = open(nomFichierSource,'r',encoding = 'ASCII')


TailleFichier = len(fichierSource.read())
print ("\nTaille du fichier (en octets) :",TailleFichier)

# lecture de l'entete du fichier (ici format pgm, 4 lignes d'ent
ete)
fichierSource.seek(0,0)
entete = fichierSource.readline() + fichierSource.readline() + f
ichierSource.readline() + fichierSource.readline()

print("en-tete")
print(entete,"\n")


# lecture des donnees du fichier source
donnees = [i for i in fichierSource.read()]

for i in range(20):
    print("donnee n°",i,"caractère",donnees[i])

# fermeture des fichiers
fichierSource.close()

print("c'est fait !")
```

Pensez-vous que les données soient réellement 1 puis 3 puis 8 ? Doit-on traiter aussi le saut de ligne ?
Réponses :
Vous pouvez comparer avec le fichier ouvert dans Frhed (ouvrir le fichier originel).

On va donc faire un petit programme qui permet de travailler sur les données réelles. Ceci nécessite de transformer toute une ligne en un nombre, puis ensuite de retransformer ce nombre en suite de caractères.

In [ ]:
```
nomFichierSource = "lyceejj.pgm"

print ("Fichier source :",nomFichierSource)

# ouverture du fichier source en format ASCII
fichierSource = open(nomFichierSource,'r',encoding = 'ASCII')

entete = fichierSource.readline() + fichierSource.readline() + f
ichierSource.readline() + fichierSource.readline()

print("en-tete")
print(entete,"\n")

for i in range(5):       # un traitement idiot sur les 5 premier
es lignes
    line = fichierSource.readline()
    nombre = int(line)
    print(line, end=' ')
    nombre = nombre//2      # une opération au hasard !
    print("valeur",str(nombre))

fichierSource.close()
```

Il ne reste plus qu'à recoller les différents morceaux pour faire l'...

Exercice : inverser les niveaux de gris d'une image

Le programme ci-dessous est presque complet. Il réalise toutes opérations d'ouverture/fermeture, lecture/conversion des données/écriture. Il ne vous reste qu'à mettre l'unique ligne de code, dans la fonction traitement, qui transformera un pixel noir en un pixel blanc, un gris foncé en un gris clair, etc...

In [ ]:
```
# coding: utf8
# from codecs import open

print ("Inversion des niveaux de gris d'une image \n")

def traitement(valeur):
    """
    cette fonction ...
    @param : valeur entier
    @return :valeurbis entier
    """
    valeurbis = 255 - valeur   # a changer parce que là ça ne fa
it rien !
                           # si vous voulez vous pouvez reprendre
l'exemple
                           # ci-dessus pour en voir l'effet
    return valeurbis

# les deux variables suivantes contiennent le chemin d'accès à l
'image initiale d'une part,
# et à l'image transformée d'autre part (le fichier sera crée au
tomatiquement)
# Bien sur, ces noms sont à modifier.

# chemins absolus :
#nomFichierSource = "C:Dudule\documents\imageb.pgm"
#nomFichierDestination = "C:Dudule\documents\essaitransforme.pgm
"

nomFichierSource = "lyceejj.pgm"
nomFichierDestination = "transformation.pgm"


print ("Fichier source :",nomFichierSource)
print ("Fichier destination :",nomFichierDestination)


# ouverture du fichier source en mode lecture d'octets
fichierSource = open(nomFichierSource,'r',encoding='ASCII')


TailleFichier = len(fichierSource.read())
print ("\nTaille du fichier (en octets) :",TailleFichier)

# lecture de l'entete du fichier (ici format pgm, 4 lignes d'ent
ete)
fichierSource.seek(0,0)
entete = fichierSource.readline() + fichierSource.readline() + f
ichierSource.readline() + fichierSource.readline()

print("en-tete")
print(entete,"\n")


# lecture des donnees du fichier source
#donnees = [i for i in fichierSource.read()]
#nb donnees = len(donnees)
```

In [ ]:

```
```

Remarque

Le code ci-dessous effectue de manière plus basique, mais moins compacte, la conversion des données récupérées une par une entre deux séparateurs, en un nombre.

In [ ]:

```
i = 0
while i < 20:
    chaine = ""         # on cree une chaine vide
    while donnees[i] != '\n':          # on ajoute les caractères (chiffres)
        chaine = chaine + donnees[i]   # à la chaine de caractères
        i = i + 1
    nombre = int(chaine)               # on transforme la chaine en nombre
    print("nombre à traiter",nombre)
    i = i + 1                          #on saute le séparateur de données
```

Frederic Mandon (mailto:frederic.mandon@ac-montpellier.fr), Lycée Jean Jaurès - Saint Clément de Rivière - France (2015)