Université de Strasbourg
Première année de licence
Semestre 1

Algorithmique et Programmation 1

TP4 : Tuples et listes

1 Traitement de listes

Créer plusieurs (au moins 2) listes contenant au moins 10 entiers compris entre 0 et 255. Les boucles que vous écrirez dans cette section devront fonctionner quelque soit la longueur de la liste `lst` traitée. Donc je vous invite à les tester avec chacune des listes que vous avez créées.

1. Écrire une boucle `for` qui convertit chaque élément de `lst` en chaîne de caractères. Par exemple `lst=[5,0,50,200]` devra devenir `["5","0","50","200"]`.
2. Écrire une boucle `for` qui crée une nouvelle liste contenant des chaînes de caractères représentant les mêmes entiers que la liste `lst`. Dans l'exemple précédent, on souhaite créer la liste `["5","0","50","200"]` sans modifier la liste `lst=[5,0,50,200]`.
3. Soient `a` et `b` deux nombres. Écrire une boucle `for` qui remplace chaque élément `x` de la liste `lst` par `ax+b`. Par exemple pour `lst=[5,0,50,200]`, `a=1.5` et `b=1`, la liste `lst` devrait devenir `[8.5, 1.0, 76.0, 301.0]`.
4. Même chose qu'à la question précédente, mais on souhaite que, même quand `a` et `b` ne sont pas des entiers, la liste `lst` reste une liste d'entiers. Il faudra prendre la partie entière de l'expression `ax+b`. On souhaite aussi que ces entiers soient compris entre 0 et 255. Donc lorsque `ax+b` est inférieur à 0, l'élément de la liste sera changé en 0 et lorsque `ax+b` est supérieur à 255, l'élément de la liste sera changé en 255. Dans l'exemple de la question précédente, la liste `lst` serait devenue la liste `[8,0,76,255]`.

2 Image comportant des bandes déterminées par une liste

Dans le TP précédent, vous avez crée des images contenant des bandes ou des cases dont la couleur obéissaient à des règles d'alternance. Ici, nous allons dessiner des bandes dont la couleur est déterminée par une liste. On souhaite que ces bandes soient de largeur `largeur_bande` pixels. Par exemple pour `largeur_bande=20` et si `lst` est la liste suivante :

```
lst = [128, 98, 156, 84, 136, 38, 0, 214, 152, 11, 238, 16]
```

On devrait obtenir l'images suivante :

FIGURE 1 – La couleur de chaque bande est déterminée par la valeur de l'élément correspondant dans `lst`. Il y a autant de bandes qu'il y a d'élément dans `lst`.

La liste `lst` ci-dessus est seulement un exemple. Toutes les opérations qui suivent doivent pouvoir être fonctionnelles quelque soit la longueur de `lst`. Je vous invite à les tester sur plusieurs listes.

1. Créer une variable `largeur` représentant la largeur de l'image. Cette variable dépendra de la valeur de `largeur_bande` et de `lst`.
2. On suppose que la bande de gauche porte le numéro 0, la suivante le numéro 1 et ainsi de suite. Soit un pixel de coordonnées `(x,y)` avec $x \in [0, largeur[$ et $y \in [0, hauteur[$. Quelle est la relation qui, à partir des coordonnées `(x,y)` permet d'obtenir le numéro de la bande à laquelle appartient le pixel ?
3. On suppose que la hauteur de l'image est de `hauteur=50` pixels. Le code suivant crée une image de la taille de celle de la figure 1 ci-dessus. Il parcourt les pixels `(x,y)` et attribue à chaque pixel la couleur grise (128).

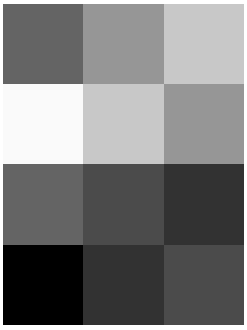
```
f = open("bandes.pgm","w")
f.write("P2\n" + str(largeur) + " " + str(hauteur) + "\n255\n")

for y in range(hauteur):
   for x in range(largeur):
      f.write(str(128) + " ")
f.close()
```

   Modifier le code ci-dessus pour qu'au lieu de dessiner du gris, ce programme attribue à chaque pixel `(x,y)` la couleur de la bande à laquelle il appartient. Vous devriez obtenir une image comparable à celle de la figure 1.

3 Même chose pour une liste de listes

Nous avons généré des bandes en fonction d'une liste. Nous pouvons aussi générer une image avec des carrés correspondant à une liste de listes d'entiers qu'on appellera `lstlst`. On souhaite que le côté de ces carrés soit de `taille` pixels. Par exemple pour `taille=40` et si `lstlst` est la liste suivante :

```
lstlst = [[100, 150, 200],
          [250, 200, 150],
          [100, 75,  50],
          [0,   50,  75]]
```



on devrait obtenir une image comme celle de la figure ci-contre.

1. Définir une variable `nb_lignes` qui représente le nombre de lignes de `lstlst`. Cette variable doit dépendre de `lstlst`. Dans notre exemple, `nb_lignes` vaut 4.
2. Définir une variable `nb_colonnes` qui représente le nombre de colonnes de `lstlst`. Cette variable doit aussi dépendre de `lstlst`. Dans notre exemple, `nb_colonnes` vaut 3.
3. Créer une variable `largeur` représentant la largeur de l'image. Cette variable dépendra de la valeur de `taille` et de `lstlst`.
4. Créer une variable `hauteur` représentant la hauteur de l'image. Cette variable dépendra de la valeur de `taille` et de `lstlst`.
5. On suppose que la ligne supérieure est la ligne 0 et que la colonne de gauche est la colonne 0. Soit un pixel de coordonnées `(x,y)` avec $x \in [0, largeur[$ et $y \in [0, hauteur[$. Quelle est la relation qui, à partir des coordonnées `(x,y)` permet d'obtenir le numéro de la ligne et de la colonne à laquelle appartient le pixel ?
6. En déduire un programme qui crée une image avec des carrés de taille `taille` (par exemple 40) et dont la couleur est déterminée par `lstlst`. Vous devriez obtenir une image comparable à celle représentée ci-dessus.

4 Projet labyrinthe : faire une image

1. Depuis le plateforme *Moodle*, télécharger le fichier `Labyrinthe.py`.
2. Dans le même répertoire que ce fichier, ouvrir un nouveau fichier et écrire :

```
import Labyrinthe
laby = Labyrinthe.creer(11,15)
for ligne in laby:
   print(ligne)
```

   La variable `laby` représente une liste de listes d'entiers. En vous inspirant des sections précédentes, et en choisissant une taille de case (par exemple 20) créer une image contenant :

   — pour chaque élément nul de `laby`, un carré noir de côté `taille` ;

   — pour chaque élément non-nul de `laby`, un carré blanc de côté `taille`.

3. Dans le TD4, vous avez écrit un programme qui, étant donné une liste d'entiers détermine l'indice de l'élément qui vaut 3. En vous inspirant de cet exercice, écrire un programme qui détermine la ligne et la colonne de l'élément qui vaut 3 (la sortie) puis de la ligne et de la colonne dont l'élément vaut 2 (l'entrée).

5 Pour aller plus loin : lecture de fichiers

1. Ouvrir un éditeur de texte et créer un fichier texte contenant plusieurs mots, ceux que vous voulez. Par exemple je crée le fichier `toto.txt` contenant le texte : `"Bonjour ! Tout va bien ?"`.
2. Dans le même répertoire, écrire le programme suivant :

```
fich = open("toto.txt","r")
toutlefichier = fich.read()
print(toutlefichier)
fich.close()

mots = toutlefichier.split()
print(mots)
```

   — La première ligne ouvre le fichier `toto.txt` en lecture ;

   — La deuxième ligne met tout le contenu du fichier dans une seule chaîne de caractères `toutlefichier`.

   — La quatrième ligne ferme le fichier.

   — La cinquième ligne crée une liste de chaînes de caractères contenant chacun un mot du texte.

   Dans le cas du fichier exemple, le résultat du programme serait :

```
Bonjour ! Tout va bien ?
['Bonjour', '!', 'Tout', 'va', 'bien', '?']
```

   Vérifier que vous obtenez le même résultat.

6 Pour aller plus loin : traitement d'images

À présent que nous savons lire et écrire dans un fichier texte, nous pouvons commencer à réaliser des opérations de traitement d'images simples. Ces opérations consistent essentiellement en trois étapes :

lecture : Lire un fichier image et en extraire une liste d'entiers représentant l'intensité des pixels ;

traitement : Transformer la liste d'entiers en une autre liste d'entiers ;

écriture : Enregistrer le résultat dans un autre fichier image.

6.1 Lecture

1. Téléchargez depuis la plateforme *Moodle* le fichier `minotaure.pgm` et lire ce fichier comme vous avez lu le fichier `toto.txt` à la section précédente. Si la liste affichée est trop longue, vous pouvez utiliser le slicing pour n'afficher que les 10 premiers éléments par exemple.
2. Dans la liste `mots` quel est l'indice de l'élément qui représente la largeur de l'image ? et la hauteur ? À partir de quel indice commencent les pixels ?
3. En utilisant le *slicing* extraire de `mots` une liste `pixels` de chaînes de caractères représentant la valeur des pixels ;
4. Dans la liste `pixels` convertir chaque chaîne de caractères en entier. Ainsi `pixels` est, à présent, une liste d'entiers représentant l'intensité des pixels de l'image.

6.2 Écriture

C'est à dessein que nous abordons l'écriture avant le traitement. Vous avez déjà écrit de nombreux fichiers au format pgm.

Ouvrir un nouveau fichier en écriture `minotaure2.pgm` et y écrire l'entête en spécifiant la même taille que celle de `minotaure.pgm`. Y inscrire le contenu de `pixels`. Vérifier que l'image de départ et l'image produite sont identiques.

6.3 Traitement

À présent, avant d'écrire la liste d'entiers dans le fichier image, nous allons transformer cette liste. Pour cela, vous pouvez utiliser l'opération que vous avez mise au point à la dernière question de la section 1. À titre de proposition, essayer :

— `a=0.5` et `b=0` ;

— `a=2` et `b=0` ;

— `a=20` et `b=-1872` ;

— `a=-1` et `b=255` ;

Nous verrons plus loin comment on peut mettre une image de fond sur notre labyrinthe.