TP n°3 : Tableaux

Instruction générale : hormis pour les exercices corrigés collectivement, vous ferez valider votre travail par l'enseignant.

1 Tableaux

1.1 Définition

En Python, un **tableau** correspond à une liste de listes de même longueur.

Exemple : le tableau suivant est une liste en contenant deux autres de longueur 3.

```
tableau=[[1,2,3],[4,5,6]]
```

Comme les tableaux sont des listes, toutes les opérations valables pour les listes le sont aussi pour les tableaux : on peut ajouter, supprimer, modifier, etc des éléments.

Se pose toutefois une question, les listes contenues dans la liste principale sont-elles des lignes ou des colonnes de notre tableau ? Autrement dit, lequel des deux tableaux ci-dessous l'exemple précédent représente-t-il ?

1	2	3
4	5	6

1	4
2	5
3	6

Il s'agit du premier tableau. On peut considérer cela comme plus naturel dans la mesure où on code le tableau en lignes de code. La liste principale contient donc les lignes et les listes secondaires contiennent ainsi les élément des colonnes.

On note nombreLigne $\times$ nombreColonnes la taille des tableaux. Par exemple, les deux tableaux ci-dessus ont tailles 2×3 et 3×2.

Exercice 1. Dans la console Python, écrire un tableau correspondant à celui ci-dessus à droite.

Il n'est évidemment pas pratique de déclarer ainsi de grands tableaux. C'est pourquoi il est plus intéressant d'utiliser de la compréhension.

Exemple : écrivons un tableau 5×10 rempli de 0 ; il a 5 lignes et 10 colonnes.

```
tableau=[[0 for colonne in range(10)] for ligne in range(5)]
```

Exercice 2. Déclarer en compréhension un tableau 10×5 ne contenant que des 1.

Exercice 3. Déclarer en compréhension le tableau ci dessous.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Exercice 4. [Fonction tableau] Écrire une fonction `fonctionTableau` prenant trois variables : `nbrLignes`, `nbrColonnes` et `valeur` puis créant un tableau de taille `nbrLignes` $\times$ `nbrColonnes` rempli de `valeur`.

Exercice 5. [Taille d'un tableau inconnu] Comment obtenir la taille d'un tableau déjà déclaré ?

1.2 Accéder aux éléments d'un tableau

La commande `tableau[i]` permet d'accéder à la ligne i du tableau. La commande `tableau[i][j]` permet d'accéder à la colonne j de la ligne i, il s'agit donc d'un élément.

Exemple : Dans la console Python, on obtient :

```
>>>tableau=[[1,2,3],[4,5,6],[7,8,9]]
>>>tableau[0]
[1,2,3]
>>>tableau[0][1]
2
```

Exercice 6. Toujours avec l'exemple ci-dessus, qu'obtient-on avec les commandes suivantes ? Le vérifier dans la console.

1. `tableau[2]` ;
2. `tableau[1][0]` ;
3. `tableau[2][1]`.

En accédant ainsi aux éléments d'un tableau, on peut facilement les déclarer ou les modifier.

Exemple : Dans la console Python, on obtient :

```
>>>tableau=[[1,2,3],[4,5,6],[7,8,9]]
>>>tableau[0][0]=0
>>>tableau[0][0]
0
```

On peut ainsi parcourir un tableau grâce à une double boucle et modifier les éléments nous intéressant.

Exemple : Supposons que `tableau` soit un tableau de taille 5×10 rempli de 0. Nous voudrions le remplir de 1 sans utiliser la compréhension mais en utilisant une double boucle.

```
for numLigne in range(5):
    for numColonne in range(10):
        tableau[numLigne][numColonne]=1
```

Exercice 7. Déclarer le tableau de l'exercice 3 à l'aide d'une double boucle.

Exercice 8. [Identité] En mathématiques, les tableaux sont appelés des matrices. Lorsque qu'une matrice possède autant de lignes que de colonnes, on parle de matrice carrée. Parmi les matrices carrées, l'une d'elle est nommée matrice identité, il s'agit de la matrice qui contient des 1 sur sa diagonale et des 0 partout ailleurs. Par exemple, pour un tableau de taille 3×3, on obtient

1	0	0
0	1	0
0	0	1

1. Quelles sont les matrices identités de tailles 2, 4 et 1 ?
2. Écrire une fonction créant une matrice / tableau identité de taille quelconque.

Exercice 9. [Fonction transposée] En mathématiques, lorsqu'on échange les lignes et les colonnes d'un tableau, on nomme le résultat transposée. Par exemple, la transposée du tableau de gauche est celui de droite.

1	2	3
4	5	6

1	4
2	5
3	6

Écrire un fonction prenant en entrée un tableau et donnant en sortie sa transposée.

Exercice 10. [Damier] Écrire un programme s'inspirant de celui-ci dessous et affichant un damier de taille quelconque. On utilisera la commande `damier[i][j]` de façon judicieuse.

```
import matplotlib.pyplot as plt

damier=[[0,1,0],[1,0,1],[0,1,0]]

plt.matshow(damier)
plt.show()
```

Exercice 11. [Cible] Écrire un programme affichant une cible de taille quelconque où une cible est un tableau de la forme

3	3	3	3	3
3	2	2	2	3
3	2	1	2	3
3	2	2	2	3
3	3	3	3	3

Pour une cible à deux zones, on aurait :

```
import matplotlib.pyplot as plt

damier=[[2,2,2],[2,1,2],[2,2,2]]

plt.matshow(damier)
plt.show()
```

2 Digression sur les images

Une image numérique est un tableau de pixels. Chaque pixel possède des coordonnées qui le positionne au sein de l'image et une valeur ou un ensemble de valeurs définissant sa couleur. Dans le système RVB, la couleur est définie à la l'aide des composantes Rouge, Vert Bleu. Notre pixel est donc défini ici par cinq valeurs. En modifiant les coordonnées ou les valeurs de couleurs du pixel, il est ainsi possible de modifier de l'image.

Exemple : le programme Python suivant récupère une image et la noircit (accessible ici avec l'image associée).

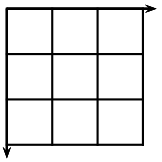
```
import PIL
from PIL import Image

def Noirssissement(image) :
    # récuperation des dimensions
    (xmax,ymax) = image.size
    # parcours de l'image
    for x in range(xmax):
        for y in range(ymax):
            # récuperation du pixel
            px = image.getpixel((x,y))
            (r,g,b) = px
            # noircissement du pixel
            px2 = (r//2, g//2, b//2)
            image.putpixel((x,y), px2)

image = Image.open("tux.png")
image = image.convert('RGB')

image.show()
Noirssissement(image)
image.show()
```

Remarque : dans une image l'axe des ordonnées n'est pas orienté vers le haut mais vers le bas, ainsi le « premier » pixel n'est pas en bas à gauche mais en haut à gauche.



Exercice 12. [Éclaircissement] Récupérer le programme ci-dessus et le modifier afin qu'il éclaircisse l'image au lieu de la noircir.

Exercice 13. [Négatif] Écrire un programme donnant le négatif d'une image.

Exercice 14. [Symétries] Écrire un programme donnant les symétries horizontale et verticale d'une image.

Exercice 15. [Rotation] Écrire un programme donnant la rotation à 90 degré d'une image.

Exercice 16. [Superposition] Écrire un programme superposant deux images de même taille (on pourra superposer un Tux et l'une de ses symétries).

3 Projets

3.1 Instructions générales

Le nom des variables, des fonctions, etc devra être en français et code de votre niveau, utilisant les notions vues en cours. Les programmes en anglais ou utilisant des notions qui n'ont pas été vues en cours ne seront pas acceptés. Vous rendrez les deux fichiers buse.py et jeuDeLaVie.py dans un dossier au format Nom Prénom.

3.2 BUSE

C'est la fin de la cinquième année pour Harry, Hermione, Ron, Neville et Malefoy qui doivent donc passer leur Brevet Universel de Sorcellerie Élémentaire (BUSE). Ils doivent tous passer cinq épreuves notées entre 0 et 20 pour les besoins de cet exercice.

1. Créer un tableau dont chaque ligne sera au format

   `['nom du personnage',note 1,...,note 5]`

   Tous les personnages susmentionnés devront apparaître dans le tableau mais vous pouvez en ajouter d'autres si vous voulez. Les notes des personnages devront être prises aléatoirement entre 0 et 20 sauf pour Hermione dont elles seront entre 18 et 20.
2. Ajouter trois colonnes au tableau. Une pour la moyenne, une pour le minimum et une dernière pour le maximum de chaque élève.
3. Ajouter une ligne au tableau. Elle contiendra les moyennes des notes à chaque épreuve, la moyenne générale, la note minimale et la note maximale de toutes les épreuves.
4. Un élève obtient une BUSE dans une matière si sa note à l'examen de celle-ci est supérieure ou égale à 10. Faire en sorte que votre programme affiche pour chaque élève une phrase de la forme

   `nomÉlève a eu nombreBUSE ; moyenne : valeurMoyenne ; note minimale : noteMin ; note maximale : noteMax`

3.3 Le jeu de la vie

Le jeu de la vie est un automate cellulaire crée par John Conway en 1970. On pourra regarder la vidéo de Sciences Étonnantes pour une présentation plus complète. Il s'agit d'une grille (ou tableau) sur laquelle évoluent des cellules ; une case du tableau contient un 1 si une cellule est présente et un 0 s'il n'y a pas de cellule.

Le jeu se décompose en une succession de tours et la vie des cellules obéit à deux règles à chacun d'entre eux :

survie : si une cellule à 2 ou 3 cellules voisines, elle survit au tour prochain, sinon elle meurt ;

naissance : si une case vide a exactement 3 cellules voisines, une cellule naît sur celle-ci au prochain tour.

Exemples : on se concentre dans chacun des exemples suivants sur la case centrale.

Survie

Tour actuel :

1	1	0
0	1	0
1	0	0

Tour suivant :

1	1	0
0	1	0
1	0	0

Mort par surpopulation

Tour actuel :

1	1	0
0	1	1
1	0	0

Tour suivant :

1	1	0
0	0	1
1	0	0

Mort par sous-population

Tour actuel :

0	0	0
0	1	0
1	0	0

Tour suivant :

0	0	0
0	0	0
1	0	0

Naissance

Tour actuel :

0	0	1
0	0	1
0	1	0

Tour suivant :

0	0	1
0	1	1
0	1	0

Votre mission, si vous l'acceptez (en fait vous n'avez pas le choix), est de programmer le jeu de la vie. Vous êtes libre de choisir la taille de la grille et la configuration de départ des cellules (vous pouvez choisir une configuration aléatoire). On représentera le tableau à l'aide de cases blanches et noires dans matplotlib.