❄ Chapitre 12 ❄

Les tableaux à 2 dimensions en Python

I. Comment représenter un tableau en Python

Un tableau à double entrée (ou tableau à deux dimensions) n'est pas un type de données reconnu nativement par Python. Nous allons utiliser des listes pour le représenter et pour le manipuler.

Voici un exemple de tableau :

Nom	Prénom	Âge
Dulac	Viviane	17
Enchanteur	Merlin	16
Lafée	Morgane	16
Pendragon	Arthur	18
Dulion	Yvain	17

Chaque ligne du tableau va correspondre à une liste en Python, le tableau consistant donc en une liste de listes.

```
tableau=[['Nom', 'Prenom','Age'],
         ['Dulac', 'Viviane','17'],
         ['Enchanteur', 'Merlin','16'],
         ['Lafee', 'Morgane','16'],
         ['Pendragon', 'Arthur','18'],
         ['Dulion', 'Yvain','17']]
```

On obtient donc une liste de la forme `tableau = [ligne1, ligne2, ligne3,...]`, chaque ligne étant elle-même une liste.

Pour accéder à la 3^e ligne du tableau, il faut donc écrire : `tableau[2]`.

Pour accéder au deuxième élément de la 3ème ligne (Merlin), on écrira donc : `tableau[2][1]`.

II. Les opérations sur les tableaux

Comme le tableau est une liste, on peut facilement lui ajouter une ligne avec la méthode .append déjà rencontrée.

Par exemple pour rajouter Debretagne Guenièvre, qui a 16 ans, à la fin de mon tableau, j'utilise la syntaxe suivante : `tableau.append(['Debretagne','Guenievre','16'])`.

La hauteur du tableau (ici 7) étant la taille de la liste principale, on peut connaître sa valeur en utilisant : `len(tableau)`.

La largeur du tableau (ici 3), constante cette année, est la taille de n'importe quelle ligne, donc on la trouve grâce à : `len(tableau[0])`.

Ces deux informations vont nous permettre de parcourir les tableaux à deux dimensions à l'aide de deux boucles for imbriquées l'une dans l'autre. La syntaxe est la suivante (tableau étant le tableau précédent) :

```
hauteur = ..............
largeur = ..............
for i in range(hauteur):
        for j in range(largeur):
                print(tableau[i][j])
```

Le parcours du tableau se fait alors ligne par ligne.

III. Exemple d'application : les images

La plupart des formats d'image sont dit matriciels, c'est à dire que l'image est simplement un tableau à deux dimensions dont chaque case contient la couleur d'un pixel de l'image. Pour récupérer ce tableau et manipuler les images de manière générale nous allons utiliser la bibliothèque PIL (Python Image Library), ou plus particulièrement un fork récent du nom de Pillow (pour Python 3).

La première ligne de votre éditeur devra être : `from PIL import Image`
Cela permet d'importer de nouvelles commandes. Les plus importantes sont :

- `image1 = Image.open(`chemin de l'image `)` : ouvre une image est la stocke dans la variable image1.
- `largeur, hauteur = image1.size` : permet de récupérer les dimensions de l'image.
- `image1.getpixel((i,j))` : permet de récupérer un triplet (rouge,vert,bleu) contenant les valeurs RGB du pixel colonne i et ligne j.
- `image1.putpixel((i,j),(r,g,b))` : permet d'écrire la couleur du pixel colonne i et ligne j.
- `image1.show()` : permet d'afficher l'image avec un logiciel externe.
- `image1.save(`chemin de l'image `)` : permet de sauvegarder l'image.

Il y a beaucoup d'autres commandes à découvrir pour les plus curieux mais celles-ci sont suffisantes pour la majorité des manipulations nécessaires.

Remarque :

Pour que cela fonctionne correctement avec pyzo, il est plus simple de faire en sorte que l'image soit enregistrée dans le même répertoire que votre script. Il faut alors utiliser la commande Exécuter le fichier comme script du menu exécuter (ctrl + shift + E).

Exemple 1:

Ouverture du fichier Fleur.jpg

```
from PIL import Image
image1 = Image.open('Fleur.jpg')
largeur,hauteur = image1.size
print(largeur,hauteur)
image1.show()
```

Modifions le programme précédent pour récupérer la couleur des pixels puis inverser la composante rouge avec la bleue :

```
from PIL import Image
def inversionRB(img):
        image1 = Image.open(img)
        largeur,hauteur = image1.size
        for i in range(largeur):
                for j in range(hauteur):
                        (r,g,b) = image1.getpixel((i,j))
                        #inversion des composantes
                        ..............................
        image1.show()
```

Exercice 1 A partir des exemples ci-dessus :

1. Créer maintenant une fonction permettant d'afficher la fleur en vert.
2. Créer une fonction permettant d'afficher une image en niveaux de gris (dans un premier temps on prendra pour chaque pixel la moyenne de ses composantes RGB. Par exemple un pixel (110, 120, 190) deviendra (140, 140, 140). On pourra améliorer la conversion dans un second temps).
3. Créer une fonction permettant d'afficher l'image en négatif. C'est à dire que les composantes de chaque pixel sont remplacées par leur complément à 255 : (110, 125, 207) ⟶ (145, 130, 48).
4. Autre chose ! (découper une image, faire un négatif de gris, appliquer un filtre à une composante, faire une symétrie, ...). Soyez créatifs !

IV. Exemple d'application : les fichiers CSV

Voir l'activité Fichiers CSV