Première NSI

Introduction aux tableaux à 2 dimensions

qkzk

2021/12/04

Comment représenter plusieurs dimensions dans un tableau ?

Imaginons qu'on dispose de cases dans une grille à deux dimensions. Chaque case possède deux coordonnées : un numéro de ligne et un numéro de colonne.

```
    0 1 2 3 4
0       |
1       |
2  -----X
3
4
```

1. Quelles sont les coordonnées `(ligne, colonne)` de `X` ?
2. Quel objet pourrait-on utiliser pour représenter ce tableau ?
3. Lorsqu'on souhaite enregistrer des données fixes, qui ne changeront pas, quelle structure choisir ? Et lorsqu'on souhaite pouvoir faire évoluer cette structure ?
4. Proposer une instruction Python pour créér une grille comme ceci :

```
    0 1 2 3 4

0   0 0 0 0 0
1   0 0 0 0 0
2   0 0 0 0 0
3   0 0 0 0 0
4   0 0 0 0 0
```

5. Proposer une fonction Python qui prend deux paramètres `n_lignes` et `n_col` (nombre de lignes et de colonnes) et renvoie un tableau à 2 dimensions, mutable, rempli de zéros.
6. Adapter votre fonction pour initialiser le tableau avec un troisième paramètre `elem`
7. Proposer une fonction qui prend trois paramètres `(table, ligne, colonne)` et renvoie l'élément contenu dans la table à cette position.
8. Proposer une fonction qui prend trois paramètres `(table, ligne, colonne, elem)` et modifie le contenu du tableau en insérant `elem` à cette position.

Une autre représentation, en ligne

Il est possible aussi d'utiliser un seul tableau, de taille `ligne * colonne` pour représenter une grille.

Plutôt que d'avoir un tableau comme ça :

```
tableau_2d = [
        [a, b, c],
        [d, e, f],
        [g, h, i],
            ]
```

On a un tableau comme ça : `tableau_1d = [a, b, c, d, e, f, g, h, i]`.

Mais on souhaite qu'il modélise un univers comme ça :

```
tableau_1d = [
        a, b, c,
        d, e, f,
        g, h, i,
          ]
```

Notez bien la différence, on souhaite représenter un plan... sur une ligne.

Ainsi pour accéder à `f` on faisait : `tableau_2d[1][2]`

On fera maintenant : `tableau_1d[5]`

Admettons que les dimensions soient 10x7 : 10 lignes, 7 colonnes et 70 cases en ligne.

1. Proposer une fonction qui prend un numéro de ligne, un numéro de colonne et renvoie l'indice de la case dans le tableau.
2. Proposer une fonction qui prend l'indice de la case dans le tableau et renvoie son numéro de ligne.

   Même chose pour le numéro de colonne.
3. Proposer une fonction qui affiche le tableau tel qu'on l'imagine, avec deux dimensions :

   ```
   >>> print_table([a, b, c, d, e, f, g, h, i, j, k, l, m, n...])

       a b c d e f g h
       i j k l m n o p
       ...
       ...
   ```
4. Améliorer votre fonction pour qu'elle affiche aussi les numéros de ligne et de colonne :

   ```
   >>> print_table([a, b, c, d, e, f, g, h, i, j, k, l, m, n...])

           0 1 2 3 4 5 6 7

       0   a b c d e f g h
       1   i j k l m n o p
       2   ...
       3   ...
       ...
   ```
5. Accéder et modifier une valeur. Proposer une fonction qui renvoie la valeur d'un tableau 2D *en ligne* pour une ligne et une colonne donnée. Même chose pour modifier une valeur.

C'est ainsi qu'on range les couleurs des pixels d'une image non compressée dans un fichier.

Le fichier comporte quelques informations (dimensions de l'image, nombre de couleurs possibles et éventuellement leurs valeurs) puis un grand tableau avec les valeurs de chaque pixel. Lorsqu'on affiche/modifie l'image, on doit utiliser des fonctions similaires pour accéder à la valeur de chaque pixel.