Cours: Matrices

1ère NSI

Introduction

Une matrice ou un tableau à 2 dimensions permet de représenter des données sous forme d'un tableau, par exemple un tableur excel ou encore une grille comme aux échecs par exemple.

Avez vous d'autres exemples de choses que l'on pourrait représenter de cette manière ?

Compétences

- Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i][j].
- Itérer sur les éléments d'un tableau.

Rappels sur les tableaux

On peut déclarer un tableau simple de la manière suivante:
`tab= [100,101,102,103]`
Si on veut parcourir ce tableau et afficher son contenu on peut le faire par exemple d'une des manières suivantes:

```
1 for val in tab:
2     print(val)
```

```
1 i = 0
2 while i < len(tab):
3     print(tab[i])
4     i+=1
```

Matrices

Pour définir un tableau à plusieurs dimensions, on défini un tableau qui contient d'autres tableaux. Voici comment on peut faire:

```
1 tab=[[21,12,53,77],
2       [74,94,29,11],
3       [42,13,66,23]]
```

Utiliser une matrice

Pour accéder à une case de notre matrices on peut écrire le code suivant: `matrice[i][j]`. On peut visualiser les indices des cases de la matrice de la manière suivante:

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]

Table: Indices des cases de la matrice

On a donc dans la notation `matrice[i][j]` le 1er élément qui représente le numéro de la ligne et le deuxième qui représente le numéro de la colonne. De plus tout comme les tableaux à 1 dimension, les indices commencent à 0.

Exercice

En considérant que la variable qui contient la matrice suivante s'appelle "matrice".

54	59	57	53	51
22	27	25	24	29
41	44	46	48	40
55	57	59	52	51

Table: Une matrice

Quelles sont les valeurs affichées par les lignes suivantes ?

- print(matrice[0][0]) .....
- print(matrice[3][2]) .....
- bonus: print(matrice[2]) .....

écrire le code pour:

- remplacer la valeur 59 par 54: .....
- remplacer la valeur 46 par 42: .....

Accéder a tous les éléments un par un (Boucle for)

On peut accéder a tous les éléments dans l'ordre, soit avec des boucles for, soit avec des boucles while de la manière suivante:

```
for ligne in matrice: #On récupère chaque element de
                      #la variable "matrice"
    for valeur in ligne: #On récupère chaque
                         #element de la ligne
        print(valeur)
```

Accéder a tous les éléments un par un (Boucle while)

On peut accéder a tous les éléments dans l'ordre, soit avec des boucles for, soit avec des boucles while de la manière suivante:

```
i = 0
while i < len(matrice): #Nombre de lignes
    j = 0
    while j < len(matrice[i]): #Taille de la ligne
        print(matrice[i][j])
        j+=1
    i+=1
```

Créer des matrices de manière dynamique

Code détaillé:

```
1 nb_lignes = 4
2 taille_ligne = 5
3
4 matrice = []
5 for i in range(nb_lignes):
6   ligne = []
7   for j in range(taille_ligne):
8     ligne.append(0)
9   matrix.append(ligne)
```

Ce code permet de créer une matrice avec 4 lignes et chaque ligne contient 5 elements (tous initialisés à 0).
Une solution plus compacte serait d'utiliser la compréhension de liste:

```
1 lignes = 4
2 taille_ligne = 5
3 matrice = [[0]*taille_ligne for i in range(lignes)]
```

Exercice 1: création

Créer la matrice suivante:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Table: Une matrice

Exercice 2: Affichage

1) Compléter/Modifier le programme suivant pour afficher la ligne demandée de la matrice:

```
ligne_a_afficher = int(input("Quelle ligne afficher ?"))
...
for .........:
    print(........., sep='')
    # sep='' permet de ne pas sauter de lignes entre chaque
    print
```

2) Utiliser le programme précèdent pour afficher toutes les valeurs de la matrice ligne par ligne.

Exercice 3: Vérification

- Écrire un programme qui vérifie si une valeur est dans la matrice.
- Afficher toutes les cases contenant une valeur supérieure à 5.

Exercice 4: Modification

3 niveaux de difficulté:

- 1: Ajouter une valeur constante à toutes les cases.
- 2: Ajouter à chaque case la valeur de la case suivante.
- 2: Remplacer les nombres pairs par -1.
- 3: Remplacer les valeurs de la diagonale qui va d'en haut à gauche vers en bas à droite par la somme des lignes et colonnes qui se croisent dessus.

Exercice 5: Pour aller plus loin ... 1/2

Soit une matrice de longueur n et de hauteur m. On veut savoir combien de manière il y à de la traverser. en considérant les limitations suivantes:

- On commence en haut à gauche.
- On ne peut se deplacer que vers le bas ou la droite.
- On termine en bas à droite.

On a par exemple:

- 2 manières de traverser une matrice de taille 2x2.
- 3 manières de traverser une matrice de taille 2x3.
- 20 manières de traverser une matrice de taille 4x4

Exercice 5: Pour aller plus loin ... 2/2

Aide:
Trouvez un moyen de compter le nombre de chemins uniques qui permettent d'arriver dans chaque case.

1			

Table: Matrice 4x3 vide

Vraiment...

Un dernier pour la route (très difficile). Pour une matrice n*m contenant des valeurs aléatoires, trouvez un chemin avec les mêmes règles que l'exercice 4 avec la somme de valeurs maximale.

5	7	9	5
10	2	4	2
3	7	6	1

Table: Matrice 4x3 aléatoire