TP 4 – MATRICES

Info1.Algo1 - 2022-2023 Semestre Impair

Matrices

Une matrice est une structure en deux dimensions, dont le nombre de lignes et de colonnes est fixé et les éléments de même type.

$$\underbrace{\begin{pmatrix} 8 & 3 & \cdots & 4 \\ 9 & 7 & \cdots & 8 \\ 5 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 9 & 6 & \cdots & 2 \end{pmatrix}}_{\longleftrightarrow \text{ nb de colonnes}} \quad \updownarrow \text{nb de lignes}$$

On manipulera dans ce TP les matrices sous la forme d'un tableau de tableaux, en imposant que le nombre de lignes et de colonnes et le type des éléments soient fixés dès la première affectation. Une fois initialisée, les seules opérations autorisées sur une matrice sont la lecture et l'écriture à un indice donné.

Exemple :

```
ma_matrice = [None] * 2
for indice_ligne in range(2):
  ma_matrice[indice_ligne] = [0] * 3
ma_matrice[1][2] = 9
for indice_ligne in range(len(ma_matrice)):
  for indice_colonne in range(len(ma_matrice[0])):
    print(ma_matrice[indice_ligne][indice_colonne], end=" ")
  print("\n")
#  [ [0,0,0],
#    [0,0,9] ]

autre_matrice = [[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]]
print(autre_matrice[1][2]) # 6
```

Applications directes

Exercice 1 - Parcourir un tableau de tableaux ★

On peut utiliser une représentation matricielle pour stocker une image ou un plateau de jeu rectangulaire.

Compléter la fonction `affichage` dans le fichier **ex01_affichage_matrice.py** pour qu'elle réalise l'affichage ligne par ligne de la matrice passée en paramètre.

On rappelle que la fonction `print` propose de préciser le délimiteur de fin de ligne pour que celui-ci soit autre chose qu'un retour à la ligne.

Exercice 2 - Vérifier la structure d'un tableau de tableaux ★

Pour être une matrice, un tableau de tableaux d'entiers doit être composé de lignes qui contiennent toutes le même nombre d'élements (qui correspond au nombre de colonnes).

Ainsi `m1 = [[6,6,7],[7,8,9,0]]` est un tableau de tableaux, mais ce n'est **PAS** une matrice

Dans le fichier **ex02_verification.py**, compléter la fonction `est_matrice` qui accepte en paramètre un tableau de tableaux (variable de type `list` dont chacun des éléments est de type `list`) et retourne `True` si les longueurs de chacun des tableaux (*éléments du tableau principal*) sont bien les mêmes (`False` sinon).

Dans la suite des exercices, on suppose que toutes les matrices données en paramètre respectent bien la condition donnée ci-dessus.

Exercice 3 : Sommes ★

1) Dans le fichier **ex03_sommes.py**, compléter la fonction `sommes_lignes` qui accepte en paramètre une matrice et retourne un tableau composé des sommes de chacune de ses lignes.

Exemple :

Si la matrice en paramètre est :

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{pmatrix}$$

... alors la fonction `sommes_lignes` retourne le tableau :

$$\begin{pmatrix} 6 & 18 & 30 \end{pmatrix}$$

2) Compléter la fonction `sommes_colonnes` qui accepte en paramètre une matrice et retourne un tableau composé des sommes de chacune de ses colonnes.

Exemple :

Si la matrice en paramètre est :

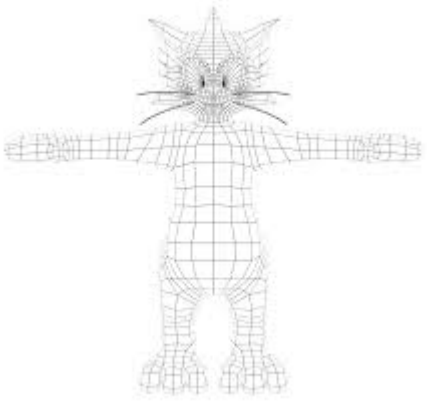
$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{pmatrix}$$

... alors la fonction `sommes_colonnes` retourne le tableau :

$$\begin{pmatrix} 15 & 18 & 21 \end{pmatrix}$$

Exercices matrices

Exercice 4 - Le chat, la souris et la matrice ★



Vous avez en entrée une carte, en deux dimensions, composée de caractères dans laquelle un chat est représenté par un `'C'` et une souris est représentée par un `'s'`.

Le reste de la carte est composée de `'.'`

Le chat essaie d'attraper la souris, il se prépare à sauter...

Pour sauter, le chat enchaine des déplacements dans quatre directions : le haut, le bas, la gauche, la droite, vers une case immédiatement adjacente.
Un saut vers une case immédiatement en diagonale est ainsi de longueur 2.
Un saut de "cheval du jeu d'échec" est un saut de longueur 3.

Cette fois-ci la distance maximale (`distance_max`) que le chat peut parcourir en sautant est un paramètre de la fonction que vous devez écrire (tous les chats ne naissent pas égaux en saut en longueur...).

Vous devez déterminer si le chat peut attraper la souris à partir de sa position :

- S'il est capable de l'attraper en un saut vous renvoyez la valeur `True`.
- Sinon, vous retournez la valeur `False`.
- Si jamais... les deux animaux ne sont pas présents, on retourne `False` *(un chat tout seul n'attrape pas de souris, une souris seule s'échappe forcément...)*.

Exemples :

```
saut = 5
carte = [
  "..C......",
  ".........",
  "....s...."
]
```

Pour une longueur de `saut` égale à `5`, la fonction retourne `True` : le chat est à moins de `5` déplacements haut/bas/gauche/droite de la souris.

```
saut = 5
carte = [
  ".C.......",
  ".........",
  "......s.."
]
```

Pour une longueur de `saut` égale à `5`, la fonction retourne `False` : **la souris est trop loin**

Dans le fichier **ex04_matrice_chats_souris** compléter la fonction `chat_attrape_souris` qui prend en paramètre un matrice de caractères et un entier `saut` et retourne `True` si le chat peut attraper la souris en un saut, `False` sinon.

Remarque : La distance manipulée dans cet exercice n'est pas la distance classique euclidienne mais la https://fr.wikipedia.org/wiki/Distance_de_Manhattan.

Exercice 5 - Construire une matrice ★

Dans cet exercice, on souhaite constituer une matrice dont les éléments sont les **produits des éléments** de deux tableaux (un tableau `colonne` et un tableau `ligne`).

Exemple : si les deux tableaux donnés sont `colonne=[1,3]` et `ligne=[4,6,5]`, le calcul est effectué de la façon suivante :

$$\begin{pmatrix} 1 \times 4 & 1 \times 6 & 1 \times 5 \\ 3 \times 4 & 3 \times 6 & 3 \times 5 \end{pmatrix}$$

On obtient alors la matrice suivante :

$$\begin{pmatrix} 4 & 6 & 5 \\ 12 & 18 & 15 \end{pmatrix}$$

Dans le fichier **ex05_matrice_produit.py**, compléter la fonction `matrice_produit` qui prend en paramètre les tableaux `colonne` et `ligne` et retourne la matrice ainsi obtenue.

Exercice 6 - matrice symétrique ★

Une matrice carrée est une matrice de nombres ayant le même nombre `n` de lignes et de colonnes. On dit qu'une matrice carrée est symétrique lorsque les valeurs sont identiques de part et d'autre de sa diagonale principale.

Exemples :

Matrice symétrique :

$$\begin{pmatrix} 1 & 5 & 3 \\ 5 & 6 & 10 \\ 3 & 10 & 12 \end{pmatrix}$$

Matrice non-symétrique :

$$\begin{pmatrix} 1 & 5 & 3 \\ 5 & 6 & 8 \\ 3 & 10 & 12 \end{pmatrix}$$

1) Dans le fichier **ex06_matrices_symetrique.py**, compléter la fonction `est_carree` qui retourne `True` si la matrice donnée en paramètre est carrée et `False` sinon.

2) Compléter la fonction `est_symetrique` qui retourne `True` si la matrice carrée donnée en paramètre est symétrique et `False` sinon.

Exercice 7 - matrice transposée ★

On appelle matrice transposée une matrice dont on permute l'organisation ligne/colonne.

Exemple :

Si la matrice est :

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

... alors sa matrice transposée est :

$$\begin{pmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{pmatrix}$$

Dans le fichier **ex07_matrice_transposee.py**, compléter la fonction `transposer` qui accepte en paramètre une matrice et retourne sa matrice transposée.

Exercice 8 - Parcours diagonal et maximum ★★

(Source : variante CT S1 déc. 2016)

La carte d'un jeu de plateau comporte des cases qui rapportent des points (un nombre entier). Ces cases sont organisées en lignes, et une case est identifiée par ses coordonnées `(indice_ligne, indice_colonne)`.

On rappelle que des coordonnées peuvent se manipuler comme un tuple, et qu'il est possible d'accéder à chacune des coordonnées (mais pas de les modifier) en utilisant les indices `0` et `1` :

```
position = 2, 3
ligne = position[0]
colonne = position[1]
# ou de manière plus compacte
ligne, colonne = position
position[0] = 4 # erreur
```

Une ligne de cases est représentée par une liste de nombres entiers et la carte d'un jeu est représentée par une liste de listes.

Ainsi, dans la carte ci-dessous, la case `(0,3)` a pour valeur `5`, et la case `(3,2)` vaut `4`.

```
          0  1  2  3  4
plan =   [
       0  [1,2,0,5,3],
       1  [0,1,3,1,2],
       2  [3,1,2,0,0],
       3  [0,3,4,1,0],
       4  [0,0,1,2,3]
                     ]
```

Un point de départ est identifié par ses coordonnées et un chemin par une liste de directions parmi `"NE"`, `"SE"`, `"SO"`, `"NO"` (resp. haut-droite, bas-droite, bas-gauche et haut-gauche).

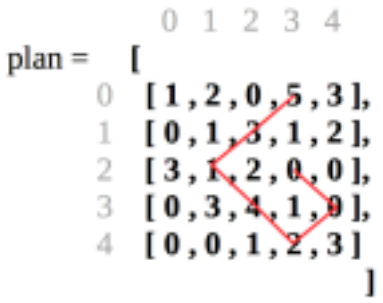
Exemple : `["SE","SO","NO","NO","NE","NE"]` est un chemin.

Dans le fichier **ex08_parcours_diagonal.py**, compléter le code de la fonction `maximum_parcours` qui prend en entrée une carte, un couple `(indice_ligne, indice_colonne)` de coordonnées du point de départ, et un chemin (sous forme de liste de chaînes de caractères) et qui retourne la plus grande des valeurs rencontrées (les points de départ et d'arrivée sont inclus).

Exemple : Avec

- la carte suivante (6 lignes, 6 colonnes) :
- le point de départ `(2,3)`
- le chemin `["SE","SO","NO","NO","NE","NE"]`

. . . le chemin réalisé est représenté ci-dessous, et la fonction retourne `5`.



Indications :

- On supposera que le point de départ est bien sur la carte et que les chemins ne font jamais sortir de la carte. Si le chemin fait repasser par une case déjà visitée cela n'a aucune importance.
- Écrivez une fonction qui prend en entrée un couple de coordonnées et une direction, et renvoie les coordonnées du point atteint après un déplacement dans la direction donnée. Nous vous rappelons qu'on suppose que les

chemins ne “sortent” pas de la carte, vous n’avez donc pas à vous préoccuper de ce souci !

Exercice 9 - Rotation d’une matrice ★★

Dans cet exercice, on s’intéresse à l’effet sur une matrice d’une rotation d’un quart de tour dans le sens horaire :

Exemple :

Si la matrice est :

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

. . . alors la matrice pivotée dans le sens horaire est :

$$\begin{pmatrix} 9 & 5 & 1 \\ 10 & 6 & 2 \\ 11 & 7 & 3 \\ 12 & 8 & 4 \end{pmatrix}$$

Dans le fichier **ex09__matrice__pivotee.py**, compléter la fonction `pivoter` qui accepte en paramètre une matrice et retourne la matrice pivotée dans le sens horaire.

Exercice 10 - Minimum local ★★

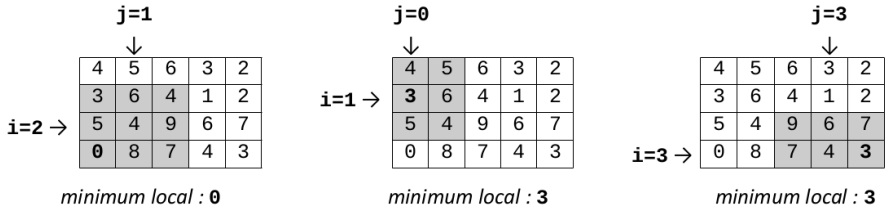
Dans le fichier **ex10__minimum__local.py**, compléter la fonction `minimum_local` qui accepte en paramètres :

- une matrice d’entiers non vide (paramètre `matrice`)
- deux indices `i` *(ligne)* et `j` *(colonne)* caractérisant une position dans la matrice.

La fonction `minimum_local` parcourt l’ensemble des positions voisines de la position `(i,j)` donnée et retourne la plus petite des valeurs rencontrées.

Exemples :

Les exemples ci-dessous donnent la valeur du minimum local pour différentes positions dans une même matrice :



- L'ensemble des positions voisines parcourues inclut la position `(i,j)` et est représentée par une zone grisée.
- L'usage des fonctions élaborées de Python (`min`, `max`, etc.) est interdit.