Enter the Matrix

1 Classiques

Exercice 1.1 (Print)

1. Écrire la fonction `printmat(`M`)` qui affiche la matrice M.
2. Bonus : Écrire la fonction `prettyprint(`M`, `d`)`.

```
>>> M = [[17, 24, 1, 8, 15],
         [23, 5, 7, 14, 16],
         [4, 6, 13, 20, 22],
         [10, 12, 19, 21, 3],
         [11, 18, 25, 2, 9]]
```

```
>>> printmat(M)
17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9
```

```
>>> prettyprint(M, 3)
-------------------------------
|  17 |  24 |   1 |   8 |  15 |
-------------------------------
|  23 |   5 |   7 |  14 |  16 |
-------------------------------
|   4 |   6 |  13 |  20 |  22 |
-------------------------------
|  10 |  12 |  19 |  21 |   3 |
-------------------------------
|  11 |  18 |  25 |   2 |   9 |
-------------------------------

# a little help: you can use format

>>> s = "|{:5d}|"
>>> print(s.format(12))
|   12|
>>> print(s.format(1254))
| 1254|
```

Exercice 1.2 (Init & load)

1. Écrire la fonction `init(`l`, `c`, `val`)` qui retourne une nouvelle matrice de $l \times c$ valeurs val.
2. Écrire la fonction `load(`$filename$`)` qui charge une matrice d'entiers depuis un fichier : les éléments d'une même ligne sont séparés par des espaces et les lignes par des '\n'.

Exercice 1.3 (Addition de matrices)

Écrire la fonction `add_matrices(`A`, `B`)` qui additionne les deux matrices A et B. La fonction déclenche une exception si les matrices ne sont pas de même dimensions.

Exercice 1.4 (Produit de matrices)

Si $A = (a_{i,j})$ est une matrice $m \times n$ et $B = (b_{i,j})$ est une matrice $n \times p$, alors leur produit $M = AB = (m_{i,j})$ est une matrice $m \times p$ donnée par :

$$\forall (i,j) \in [1,m] \times [1,p],\ m_{i,j} = \sum_{k=1}^{n} (a_{i,k}.b_{k,j})$$

Écrire la fonction `mult_matrices(`A`, `B`)` qui multiplie les deux matrices A et B (si possible, i.e. leurs dimensions sont correctes).

2 Recherches et tests

Exercice 2.1 (Minimax)

Écrire une fonction qui retourne **la position** de la valeur minimale parmi les maximums de chaque ligne d'une matrice d'entiers non vide.

1	24	12	18	4
10	15	15	0	18
8	14	0	16	2
22	4	8	14	22
19	7	23	5	5

Matrice `M`

Exemple d'application sur la matrice `M` :

```
>>> posMinimax(M)
(2, 3)
```

Exercice 2.2 (Recherche)

Écrire la fonction `searchMatrix(M, x)` qui retourne la position (i, j) de la première valeur `x` trouvée dans la matrice non vide `M`. Si `x` n'est pas présent, la fonction retourne $(-1, -1)$.

	0	1	2	3	4	5	6
0	1	10	3	0	-3	2	8
1	-1	0	1	8	5	0	-4
2	10	9	14	1	4	-5	1
3	10	-3	7	11	6	3	0
4	7	8	-5	1	5	4	10

`M1`

Exemples d'applications avec la matrice `M1` ci-dessus :

```
>>>  searchMatrix(M1, -5)
(2, 5)
>>> searchMatrix(M1, 5)
(1, 4)
>>> searchMatrix(M1, 15)
(-1, -1)
```

Exercice 2.3 (Symétrique)

La matrice transposée d'une matrice est la matrice $A^\intercal$, obtenue en échangeant les lignes et les colonnes de A.

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \text{ alors } A^\intercal = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

Une matrice *symétrique* est une matrice carrée (de taille $n \times n$) qui est égale à sa propre transposée.
Exemples :

Matrice symétrique :

	0	1	2
0	**2**	3	0
1	3	**10**	4
2	0	4	**1**

Matrice non symétrique :

	0	1	2
0	**2**	3	0
1	4	**10**	4
2	0	3	**1**

Écrire la fonction `symmetric` qui teste si une matrice carrée non vide est symétrique.

3 A kind of magic

Exercice 3.1 (Carré magique)

Carré magique

Un carré magique d'ordre n est composé de n^2 entiers, écrits sous la forme d'un tableau carré. Ces nombres sont disposés de sorte que leurs sommes sur chaque rangée, sur chaque colonne et sur chaque diagonale principale soient égales.

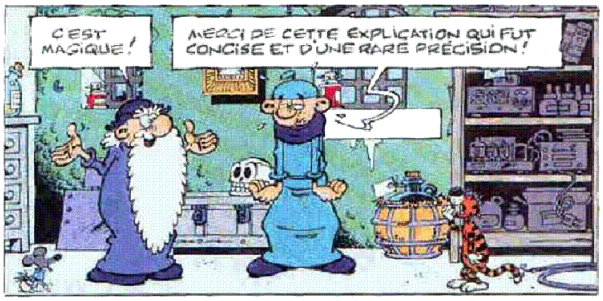
Carré magique normal

Un carré magique constitué de tous les nombres entiers de 1 à n^2, où n est l'ordre du carré est un carré magique normal.

Écrire une fonction qui construit un carré magique normal d'ordre n (n impair supérieur à 2) par la *méthode siamoise* (voir Wikipedia).

	0	1	2	3	4
0	17	24	1	8	15
1	23	5	7	14	16
2	4	6	13	20	22
3	10	12	19	21	3
4	11	18	25	2	9

	0	1	2	3	4
0	11	18	25	2	9
1	10	12	19	21	3
2	4	6	13	20	22
3	23	5	7	14	16
4	17	24	1	8	15



Exercice 3.2 (Harry Potter)

Dans une des chambres secrètes de Poudlard se trouvent de nombreuses pierres philosophales. Le sol de la chambre est couvert par $h \times w$ dalles carrées : h rangées de dalles de l'entrée de la chambre (la première rangée) au fond de la chambre (la dernière rangée) et w colonnes de dalles de gauche à droite.

Chaque dalle peut contenir de 1 à 100 pierres. Harry doit récupérer autant de pierres que possible, sous réserve des restrictions suivantes :

— Il commence sur la dalle de son choix sur la première rangée et récupère les pierres qui s'y trouvent. Puis, il se déplace sur une dalle de la rangée suivante, recueille les pierres de cette dalle, et ainsi de suite jusqu'à ce qu'il atteigne la dernière rangée.

— Quand Harry se déplace de dalle en dalle, il ne peut se déplacer que sur la dalle juste en avant ou en diagonale (gauche ou droite).

1. Écrire une fonction qui calcule le nombre maximum de pierres qu'Harry peut récupérer en un seul voyage de la première à la dernière rangée.

```
>>> T
[[3, 1, 7, 4, 2],
 [2, 3, 1, 1, 1],
 [1, 2, 1, 1, 8],
 [2, 1, 1, 5, 3],
 [2, 1, 4, 4, 4],
 [5, 2, 7, 5, 1]]

>>> harrypotter(T)
32
```

2. Bonus : Modifier la fonction : elle donne le chemin à emprunter pour récupérer le maximum de pierres.