Enter the Matrix

1 Classiques

Exercice 1.1 (Print)

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

```
>>> printmatrix(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
```

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

```
>>> prettyprintmatrix(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 |
-------------------------------
```

Exercice 1.2 (Init & load)

1. Écrire la fonction `initmatrix(`l`, `c`, `val`)` qui retourne une nouvelle matrice de $l \times c$ valeurs val.
2. Écrire la fonction `buildmatrix(`l`, `c`, `n`)` qui retourne une nouvelle matrice de $l \times c$ valeurs entières aléatoires dans $[0, n[$.

```
>>> from random import randint

>>> help(randint)
Help on method randint in module random:
randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

3. Écrire la fonction `loadmatrix(`$filename$`)` 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 `addmatrix(`A`, `B`)` qui additionne les deux matrices A et B (si elles sont de tailles identiques).

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 `multmatrix(`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 (Recherche – *C2# - 2017*)

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

Exemple d'application avec la matrice `Mat1` ci-contre :

```
>>>  searchMatrix(Mat1, -5)
(2, 5)
>>> searchMatrix(Mat1, 5)
(1, 4)
>>> searchMatrix(Mat1, 15)
(-1, -1)
```

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

`Mat1`

Exercice 2.2 (Maximum Gap – *C2 - 2017*)

Pour cet exercice, on définit le *gap* (écart) d'une liste comme étant l'écart maximum entre deux valeurs de la liste. Par exemple, dans la matrice ci-dessous le *gap* de la première ligne est 13.

Écrire la fonction qui retourne le gap maximum des lignes d'une matrice (que l'on supposera non vide).

Exemple d'application avec la matrice `Mat1` ci-contre :

```
>>> maxgap_matrix(Mat1)
19
```

En effet le gap maximum est celui de la ligne du milieu (19 = 14 - (-5)).

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

`Mat1`

Exercice 2.3 (Symétrique - *C2# - 2016*)

La matrice transposée d'une matrice est la matrice A^T, obtenue en échangeant les lignes et les colonnes de A.

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

Une matrice *symétrique* est une matrice carrée qui est égale à sa propre transposée.

Écrire la fonction `symmetric(A)` qui teste si une matrice non vide est symétrique.

3 Un peu de magie

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

1. **Tests :**
   (a) Écrire une fonction qui vérifie si une matrice est un carré magique.
   (b) Écrire une fonction qui vérifie si une matrice est un carré magique normal.
2. Écrire une fonction qui construit un carré magique de coté n (n impair) par la *méthode siamoise* (voir Wikipedia).

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

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 un script pour calculer 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, 1, 3, 1, 1],
 [1, 2, 2, 1, 8],
 [2, 2, 1, 5, 3],
 [2, 1, 4, 4, 4],
 [5, 2, 7, 5, 1]]

>>> harrypotter(T)
32
```

2. Bonus :
   Modifier le script pour qu'il donne le chemin à emprunter pour récupérer le maximum de pierres.