Enter the Matrix

1 Classics

Exercise 1.1 (Print)

1. Write the function `printmatrix(M)` that displays the matrix M.
2. Bonus : Write the function `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 |
-------------------------------
```

Exercise 1.2 (Init & load)

1. Write the function `initmatrix(l, c, val)` that builds a new matrix with $l \times c$ values val.
2. Write the function `buildmatrix(l, c, n)` that builds a new matrix with $l \times c$ random integers in $[0, n[$.

```
>>> 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. Write the function `loadmatrix(filename)` that loads an integer matrix from a file: elements of a line are separated by spaces, each line is ended by '\n'.

Exercise 1.3 (Matrix sum)

Write the function `addmatrix(A, B)` that sums the two matrices (if they have same dimensions).

Exercise 1.4 (Matrix product)

If $A = (a_{i,j})$ is an m-by-n matrix and $B = (m_{i,j})$ is an n-by-p matrix, then their matrix product $M = AB = (m_{i,j})$ is the m-by-p matrix such that:

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

Write the function `multmatrix(A, B)` that multiplies the two matrices A and B (when possible, i.e. their dimensions are corrects).

2 Searches and tests

Exercise 2.1 (Research – *C2# - 2017*)

Write the function `searchMatrix(M, x)`v that returns the position (i, j) of the first value `x` found in the matrix `M`. If `x` is not present, the function returns $(-1, -1)$.

Example of result with the matrice `Mat1` *on the right:*

```
>>>  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`

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

In this exercise, the *gap* of a list is defined as the maximum difference between two values of the list. For instance, in the matrice below, the *gap* of the first line is 13.

Write the function that returns the maximum gap of the lines of a matrice (assumed non empty).

Example of result with the matrice `Mat1` *on the right:*

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

Indeed the maximum gap is the one of the middle line (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`

Exercise 2.3 (Symmetric - *C2# - 2016*)

The transpose of a matrix A is the matrix A^T obtained by switching the rows and columns of the matrix A.

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

A square matrix whose transpose is equal to itself is called a *symmetric* matrix.

Write the function `symmetric(A)` that tests whether a non empty matrix is symmetric.

3 A little magic

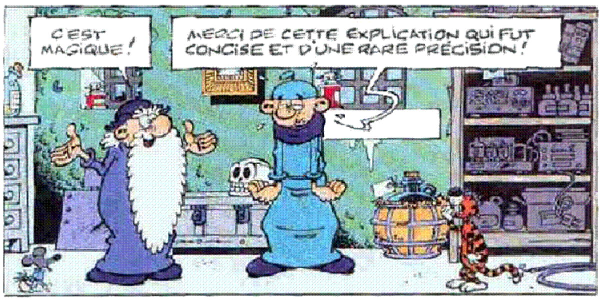
Exercise 3.1 (Magic Square)

Magic Square
A magic square of order n is an arrangement of n^2 numbers, usually integers, in a square grid, where the numbers in each row, and in each column, and the numbers in the main and secondary diagonals, all add up to the same number.

Normal Magic Square
A magic square that contains the integers from 1 to n^2 is called a normal magic square.

1. **Tests:**
   (a) Write a function that tests whether a matrix is a magic square.
   (b) Write a function that tests whether a matrix is a normal magic square.
2. Write a function that builds a size of n-odd magic square using the Siamese method (see 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



Exercise 3.2 (Harry Potter)

One of the secret chambers in Hogwarts is full of philosopher's stones. The floor of the chamber is covered by $h \times w$ square tiles, where there are h rows of tiles from front (first row) to back (last row) and w columns of tiles from left to right. Each tile has 1 to 100 stones on it. Harry has to grab as many philosopher's stones as possible, subject to the following restrictions:

- He starts by choosing any tile in the first row, and collects the philosopher's stones on that tile. Then, he moves to a tile in the next row, collects the philosopher's stones on the tile, and so on until he reaches the last row.
- When he moves from one tile to a tile in the next row, he can only move to the tile just below it or diagonally to the left or right.

1. Write a script to compute the maximum possible number of philosopher's stones Harry can grab in one single trip from the first row to the last row.

```
>>> 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:
   Modify the script so that it gives the path to follow to grab the maximum possible stones.