# Enter the Matrix

## 1 Classics

### Exercise 1.1 (Print)

1. Write the function `printmat(`$M$`)` that displays the matrix $M$.
2. Bonus : Write the function `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
```

```
>>> 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|
```

### Exercise 1.2 (Init & load)

1. Write the function `init(`$l$`, `$c$`, `$val$`)` that builds a new matrix with $l \times c$ values $val$.
2. Write the function `load(`$filename$`)` 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 `add_matrices(`$A$`, `$B$`)` that sums the two matrices $A$ and $B$. The function raises an exception if the matrices do not have the 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 `mult_matrices(`$A$`, `$B$`)` that multiplies the two matrices $A$ and $B$ (when possible, i.e. their dimensions are correct).

# 2 Searches and tests

**Exercise 2.1 (Maximum gap - *Midterm S2# 2020*)**

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

Write the function that returns the maximum gap of the lines of a matrix of size at least $2 \times 2$.

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

```
>>> maxGap(Mat1)
19
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 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`

Indeed the maximum gap is the one of the middle line (19 = 14 - (-5)).

**Exercise 2.2 (Symmetric - *Midterm S2# 2020*)**

The transpose of a matrix $A$ is the matrix $A^\mathsf{T}$ 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.
Exemples :

Symmetric matrix:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | **2** | 3 | 0 |
| 1 | 3 | **10** | 4 |
| 2 | 0 | 4 | **1** |

Non symmetric matrix:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | **2** | 3 | 0 |
| 1 | 4 | **10** | 4 |
| 2 | 0 | 3 | **1** |

Write the function `symmetric` that tests whether a non empty square matrix is symmetric.

**Exercise 2.3 (Sorted matrix – Midterm S2 2021)**

**Definition:**
A matrix is *sorted* in increasing order if:

- each line of the matrix is sorted in increasing order
- all elements of each line of the matrix (except the first one) are strictly greater than all the elements of the previous line.

1. Write the function `list_sorted(L, n)` that checks if the non empty list `L` of length `n` is sorted in increasing order.
2. Write the function `matrix_sorted(M)` that checks if the matrix `M` is *sorted* in increasing order. The matrix `M` is supposed non empty.

*Application examples:*

```
>>> matrix_sorted([[1,2,3]])
True
>>> matrix_sorted([[1,2,3], [40,5,6]])
False
>>> matrix_sorted([[1,2,3], [7,8,9], [4,5,6]])
False
>>> matrix_sorted([[1,2,3], [4,5,6], [7,8,9]])
True
>>> matrix_sorted([[1,-1,3], [4,5,6], [7,8,9]])
False
```

# 3 A kind of magic

**Exercise 3.1 (Magic Square – *Midterm S2 2020*)**

**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.

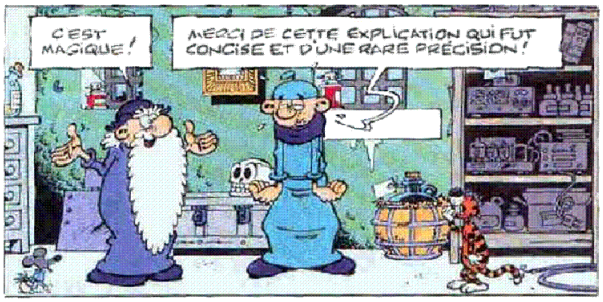
**Normal Magic Square**

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

Write a function that builds a normal magic square of order $n$ ($n$ odd greater than 2) using the Siamese method (see 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 |




**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 from 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 function that computes the maximum 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, 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: Modify the function so that it gives the path to follow to grab the maximum number of stones.