**Consignes.** Avant toute chose, prenez conscience que les corrections sont effectuées à l'aide d'un automate, par conséquent toute violation des règles conduira inévitablement à la délivrance d'une note nulle.

- Comment livrer mon code ?
  Si mon nom de famille est **yunès** et que je souhaite rendre le travail pour l'exercice 12 du tp 3 alors je dois construire une archive au format ZIP de nom `yunes-tp3-ex12.zip`, le nom ayant été converti en minuscules, sans accent et sans espace (*ie* : **de Gaulle** doit être traduit en `degaulle`). Si vous avez un homonyme rajoutez des initiales, par exemple `yunesjb-tp1-ex1`. Respectez votre nommage pour tout le semestre.
- Que mettre dans l'archive ?
  Le code source (et uniquement le code source) correspondant à l'exercice.

De plus, suivez bien les instructions complémentaires des livrables.

Pour ce TP, il faut récupérer les fichiers de trace du TP n°2 (`access.log` et `error.log`).

## Exercice 1

Créer une class Python de nom `Grid` permettant de représenter une grille d'objets (le tout dans un fichier de nom `Grid.py`, c'est donc un module python). Les dimensions de la grille étant fixée à la construction de l'instance et doit contenir que des `None`. De plus il faut que cette grille s'affiche «normalement» (voir l'exemple). Ainsi elle doit pouvoir fonctionner avec le code suivant (`main.py`) :

```
import Grid
g = Grid.Grid(2,3)
print(g[0][0])
print(g)
```

qui produira:

```
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Livrable : l'archive contenant les codes `Grid.py`, `main.py`.

## Exercice 2

On réutilisera la classe `Grid` de l'exercice précédent. Créer une classe `Pawn` permettant de représenter un pion (un X un O) de tel sorte que s'il est placé dans la grille à une certaine position, l'affichage de la grille corresponde:

```
import Grid
import Pawn
g = Grid.Grid(2,3)
g[1][0] = Pawn.Pawn('X')
print(g)
```

qui produira :

```
+---+---+---+
|   |   |   |
+---+---+---+
| X |   |   |
+---+---+---+
```

Livrable : l'archive contenant `Grid.py`, `Pawn.py`, `main.py`.

## Exercice 3

On réutilisera les codes précédents. Modifier la classe `Grid` de sorte que l'on puisse poser un pion sur une colonne et que celui-ci glisse en tombant d'une case à la fois :

```
import Grid
import Pawn
g = Grid.Grid(2,3)
p1 = Pawn.Pawn('X')
g.put(0,p1)
```

qui produira :

```
+---+---+---+
| X |   |   |
+---+---+---+
|   |   |   |
+---+---+---+

+---+---+---+
|   |   |   |
+---+---+---+
| X |   |   |
+---+---+---+
```

Attention à bien vérifier si on peut déposer le pion dans la colonne, etc. Tester avec plusieurs pions, remplissez une colonne, etc
Livrable : l'archive contenant le programme `Grid.py`, `Pawn.py` et `main.py`.

## Exercice 4

Compléter les classes de sorte que l'on puisse jouer au jeu puissance 4 (vous trouverez facilement les règles de ce jeu).
Livrable : l'archive contenant tous les codes utiles dont `main.py` qui, lancé, permettra de jouer.