G. SAUREL
16/11/2022

# Conception Orientée Objet

Solution sur la branche : Connect4-sol sur Git

## Revoir TP connect 4

Découpage ligne x colonne appelé Row Major
colonne x ligne — Colum Major

Q1 :

```
from .game import Grid, Player
from .game import Grid, Player
class DumbIA(Player):
    def play(self, grid: Grid) -> int:
        for line in grid.grid:
            for column, cell in enumerate(line):
                if cell == cell.EMPTY:
                    return column
```

Tuple qui mémorise aussi le n° de colonne dans le for() de rang supérieur

Rappel : la méthode play() du joueur retourne uniquement l'indice de colonne dans laquelle le jeton doit être placé. C'est la classe Game qui s'occupe de placer le jeton

Q2 : List comprehension

```
def tie(self) -> bool:
    return all(cell != Cell.EMPTY for line in self.grid
               for cell in line)
```

all() retourne faux dès qu'il reçoit faux et quitte
si retourne Vrai si tout a été parcouru sans faux

Q4 :

```
def play(self, grid: Grid) -> int
    print(grid)
```

```
return int(input("colum : "))
```

Q5 :

```
from .game import Grid, Player
from . game import Cell, Grid, Player
class CreatorB (Player):
def play (self, grid : Grid) -> int
  grid. grid [0] [0] = Cell. B
  grid. grid [0][1] = Cell. B
  grid. grid [0] [2] = Cell. B
  grid. grid [0] [3] = Cell. B
  return 3
```

Problématique : Comment empêcher le CreatorB de modifier la grille ? En Python, pas de classe protégée

Solution : Le joueur reçoit une représentation tactuelle de la grille et renvoie seulement la chaîne de caractère de la colonne où jouer (principe codage / décodage)

Sérialisation / désérialisation

```
def serialize_cell (cell : Cell) -> str
    return cell. value
```

rappel : la classe Cell est basée sur une énumération des symboles :

```
class Cell (ENUM)
    EMPTY = ". "
    A = "X"
    B = "O"
```

```
def deserialize_cell ( cell. str) -> Cell:
  if cell == ". ":
      return Cell. EMPTY
```

```
        if cell == "X":
            return Cell.A
    if cell == "O"
        return cell.B
```

De même pour la grille :

```
def seralize_grid (grid : Grid) -> str
    ret = ""
    for line in grid.grid:
        for cell in lin
        ret += " seralize_cell (cell)
    return ret
```

```
def deseralize_grid (grid : str) -> Grid:
    ret = Grid()
    for n, cell in enumerate (grid):
        ret.grid [n // ret.columns ][ n %. ret.
        columns ] = deseralize_cell (cell)
    return ret
```

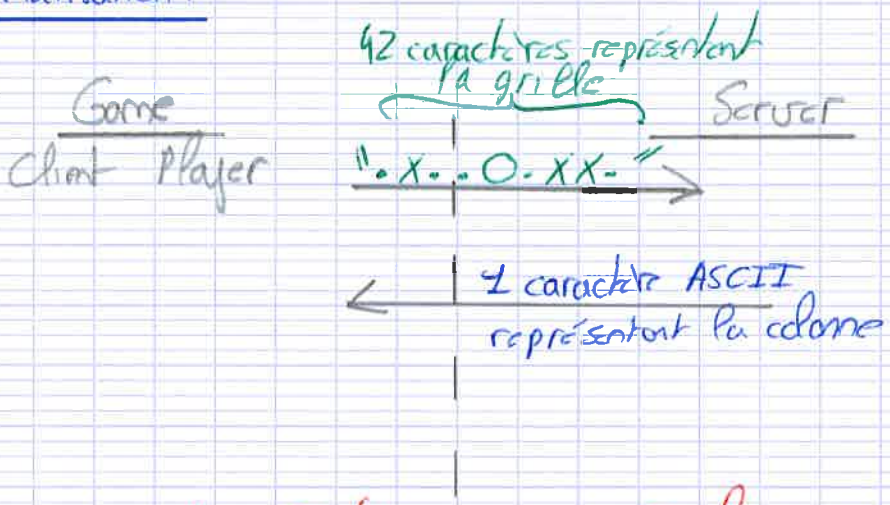
C'est le même principe que pour un jeu en réseau Client et serveur :

```
from . serd import deserialize
HOST = 'localhost"
PORT = 50821
class Server (Player):
    def __init__ (self):
        with socket.socket (socket.AF_INET, socket.
        stream) as s :
            try:
                s.bind (( HOST, PORT))
```

{ cf repo Git }

```
while True:
    s.listen()
    conn, adr = s.accept()
```

Schématisation :




## Les Systèmes Temps Réels
## Systèmes d'Exploitation Temps Réel

Diapo de H. DEMMOU

### Linux Temps Réel
### RTAI = Real Time Application Interface

Un système fonction en RT s'il est capable d'absorber toutes les infos d'entrées sans qu'elles soit trop vieilles pour être traitées.

Definitions IEEE : is a computing system whose correct behaviour depends not only on the value of the computation but also on the time at which outputs are produced.

- Reactive System : is a computing system in continuous interaction with the environment (as opposed to information processing)
- Safety Critical system : is a computing system where a failure may cause injury