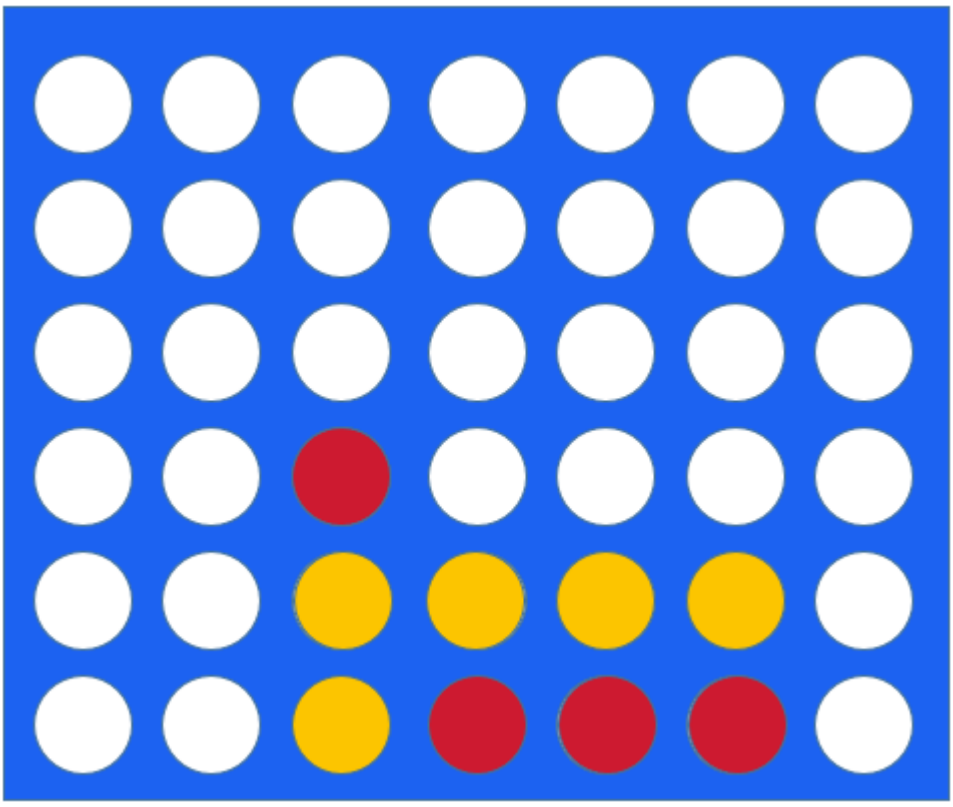
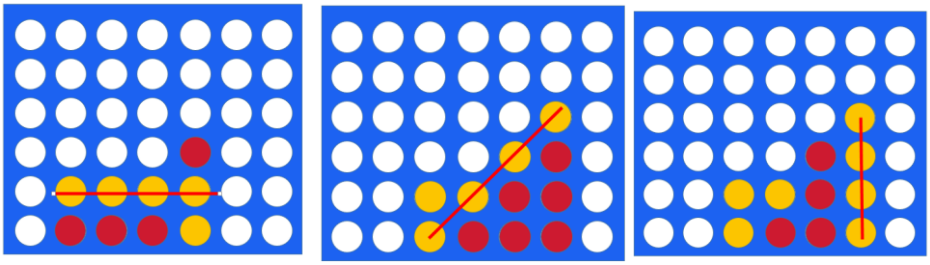
THE DISKS BATTLE



PRESENTATION OF THE GAME

Objective: Connect four of your disks in a row while preventing your opponent from doing the same. But, look out -- your opponent can sneak up on you and win the game!

DEMO: https://lululataupe.com/jeux-tablettes/tout-age/puissance-4/



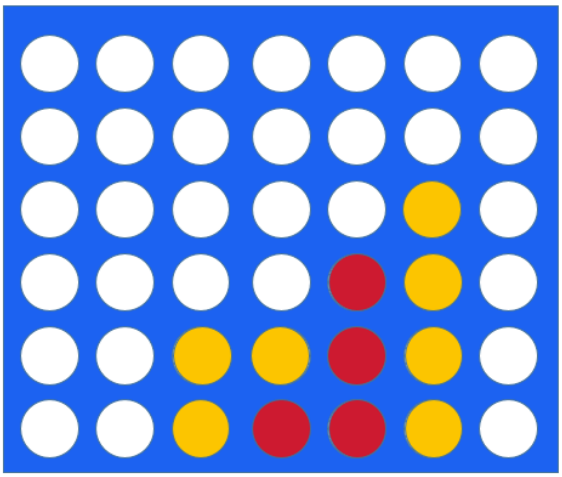
Here YELLOW player win all time: 4 yellow disks horizontally, vertically, or in diagonal!

RULES

1. Game start with an empty game board
2. The 2 players alternatively drop one of their disk into an unfilled column
3. The first player who can connect 4 disk horizontally, vertically or in diagonal, win
4. Note: If the board fills up before either player achieves four in a row, then the game is a draw.

BOARD

The board is a grid of 7 columns and 6 rows:



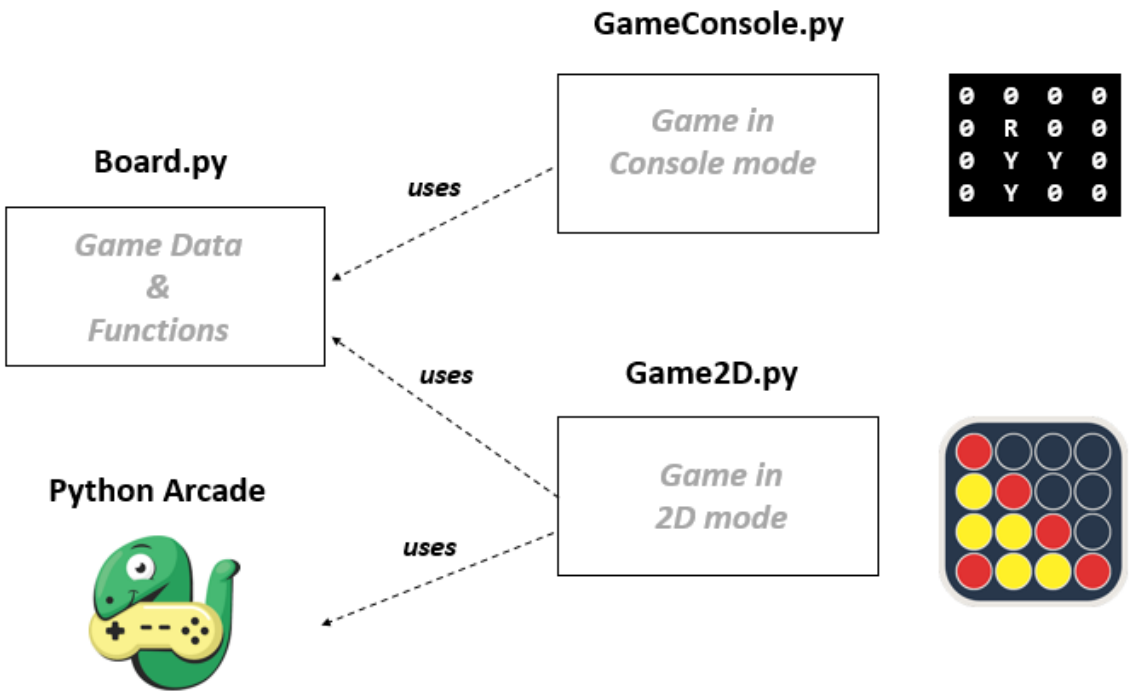
ARCHITECTURE

We want to separate the game data and functions, from the interaction in

- In console in a first step
- Using a graphical library (Arcade Python) in a second step

Therefore:

- Usage of console (get player actions and display the board and messages) should be only located in GameCOnsole.py
- Use of graphical interface (get mouse events, display the board, display messages) should be only located in Game2D.py
- The data (current status of the board) and functions to play (clear the board, add a disk to a column, see how is winning..) should be only located in Board.py
-



CONSOLE MODE

When game starts, the board is empty (no disks), and the console shall display the board with a welcome message:

```
WELCOME TO DISK-BATTLE !

A B C D E F G
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

Note: we display a letter to identify each column (it can also be a number, as you which)

The first player (YELLLOW) shall enter a position to start the game:

```
Player YELLOW, enter column to play:  C
```

Here player has chosen the column C, so we drop the YELLOW disk on column C

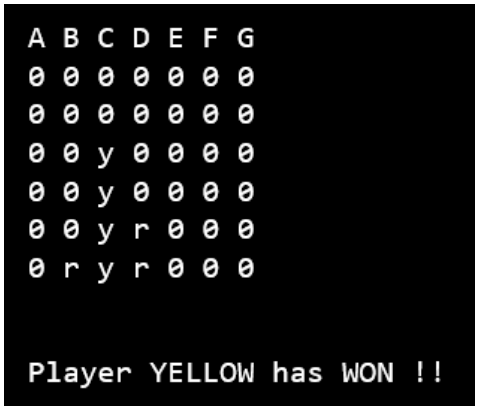
- The column C is empty, so the disk will move down to the first row

```
WELCOME TO DISK-BATTLE !

A B C D E F G
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 y 0 0 0 0


Player RED, enter column to play:  D
```

The player RED play then his action, and so on, until a player has won:



When a player win: we display a message to inform the end of the game

BOARD FILE

The following constants must be defined:

- BOARD_ROWS = 6
- BOARD_COLUMN = 7

Note that it might be possible to change those value to play with a bigger or smaller grid

The following data must be defined:

It's up to you to define how you will store the status of the board, who is the current player etc….

Tips: really thing about the different data structures you want to use before starting to code

The following 3 function (at least) must be defined:

1 clearBoard()

Clear the board to start a new game

@return nothing!

2 canPlay(columnIndex)

@param columnIndex (integer): column index to play – Column index starts from 0 to 6

@return true if you can play on this column

3 play(isRed, columnIndex)

Play on given column, as a RED player or as a YELLOW player

@param isRed (Boolean): If true, play a RED disk, otherwise, play a YELLOW disk

@param columnIndex (integer): column index to play – Column index starts from 0 to 6

@return you are free to return whatever you want (or nothing)

Note: We recommend you to return the positon (column, row) of the DISK in the board, since you might use it afterward

4 getBoardStatus()

@return the current board status *(see below)*

This function returns the status of the BOARD as a string, among the following keywords:

STATUS	MEANING
ON_PROGRESS	No winner for now
RED_WON	RED player has won
YELLOW_WON	YELLOW player has won
BOARD_FULL	The board is full, but no winner

GRAPHICS MODE

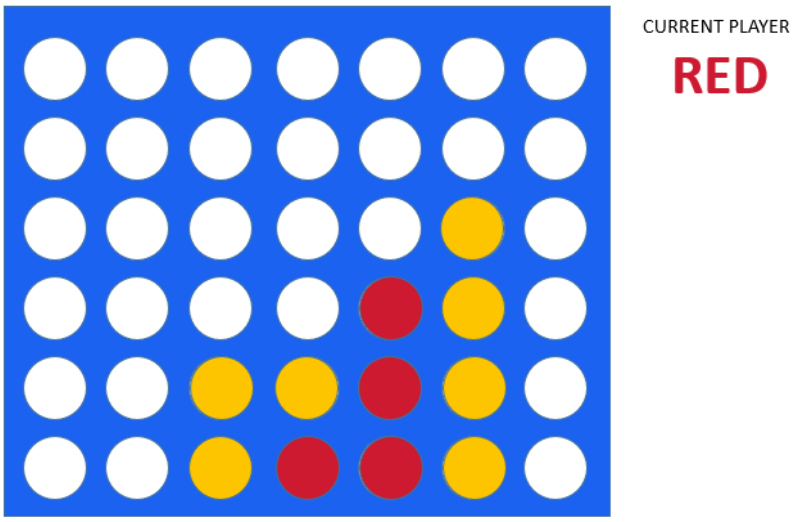
We use the **library ARCADE**

Please follow the instructions to install it and test it:

https://arcade.academy/

The interface just consist of:

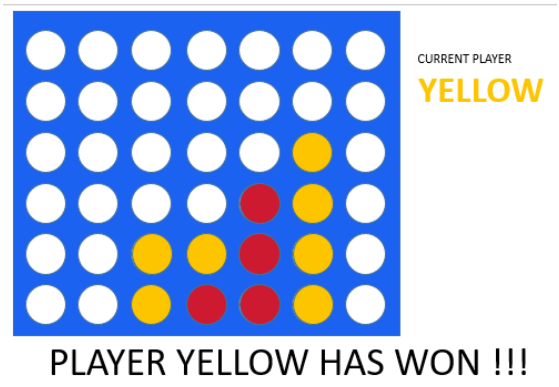
- The **board**
- The **current player** RED or YELLOW (as a text)



- The player click on a column
- The disk will drop on this column

Important: **if the column is full, it should not possible to play on this column**

- As soon as a player win or the board is full, as message shall be displayed on screen to inform about the end of the game :



DEVELOPPEMENT PROCESS & RULES

HOW TO START?

- You should first be clear about the rules and the scenario of the game
- You should then **think about the variables you need** to manage the game
- Then you can start to implement the function one by one
    - Note: you may have to add additional functions to make your code easy to read!

Example of additional functions:

- *ConsoleGame : a function to print the board on console*
- *Board : a function to get the target row when a player drop a disk on a column*

GIT/GITHUB:

- Commit as much as possible (every 2 hours...)
- The commit comment should be related to the changes performed and should be max 20 characters
  Example: *Added the clearBoard function*

- One the first iteration (console mode) has been finished : tag you repository to define the release version V1.0

CLEAN CODE

<u>The following rule must be respected:</u>

- **Meaning** full variable / function names
- **CAMEL** case
- **Comments** your functions and sections of code

  Example: # 1 - Check if word contains a "b" character
- **Avoid function with more than 20/30 line of code**
- Avoid **duplication of code**
- Each function should have a **single responsibility** (i.e. role)

PLANNING

This project is estimated about 5/6 days of development (5/6 weeks)

We want a release V1.0 for the console mode and then a release V2.0 with the graphic mode