Connect Four - The game

Este proyecto consiste en implementar el juego **Cuatro en Raya** en la terminal, con una estructura modular orientada a objetos y lógica escalable para distintos niveles de dificultad de la CPU.

Objetivos de Aprendizaje

- Uso de listas bidimensionales
- Control de flujo con estructuras `if, for, while`
- Diseño e implementación de clases y subclases
- Modularización del código en múltiples archivos
- Implementación y uso de decoradores
- Desarrollo incremental, pruebas funcionales y depuración
- Lógica algorítmica: detección de victoria y estrategia de juego de la CPU

Reglas del Juego

- El tablero tiene 6 filas y 7 columnas
- Cada jugador coloca su ficha (`X` o `0`) por turnos, eligiendo una columna
- Las fichas caen hasta la posición libre más baja de la columna seleccionada
- Gana quien consiga 4 fichas consecutivas en:
    - Horizontal

    - Vertical
    - Diagonal descendente `(/)`
    - Diagonal ascendente `(\)`
- Si se llena el tablero sin ganador, el resultado es empate

Estructura del Código

El proyecto está dividido en tres archivos principales, cada uno con una clase independiente:

1. `board.py` → Clase `Board`: gestiona el estado del tablero y su visualización.
2. `menu.py` → Clase `Menu`: gestiona la interacción con el usuario y el flujo del juego.
3. `cpu_logic.py` → Clase `CPULogic`: contiene la lógica de dificultad `(easy, normal, hard)`. Puede incorporar estrategias reutilizables o recursivas según el nivel de dificultad.

Estructura del Desarrollo

Paso 1: Clase `Board (board.py)`

Responsable de la representación y operaciones del tablero.

Métodos esperados:

- `__init__()` → inicializa una matriz 6x7 con valores
- `print_board()` → muestra el estado actual del tablero
- `insert_piece(column, piece)` → inserta una ficha en la columna válida

- `is_valid_column(column)` → comprueba si la columna está disponible
- `check_win(piece)` → comprueba condiciones de victoria (horizontal, vertical, diagonal)

Paso 2: Clase `Menu (menu.py)`

Gestiona la interfaz textual, los turnos y el flujo de ejecución.

Métodos esperados:

- `__init__()` → inicializa los componentes (tablero y lógica de CPU)
- `show_menu()` → permite seleccionar nivel de dificultad
- `start_game()` → ejecuta el bucle principal:
    - Alterna turnos
    - Valida movimientos
    - Muestra tablero tras cada jugada
    - Detecta victoria o empate
    - Permite reiniciar el juego

Paso 3: Clase `CPULogic (cpu_logic.py)`

Implementa las estrategias automáticas según el nivel elegido.

Métodos esperados:

- `cpu_easy(board)` → devuelve una columna aleatoria válida
- `cpu_normal(board, piece, opponent_piece)` → intenta ganar o bloquear
- `cpu_hard(board, piece)` → aplica heurística o algoritmo recursivo (opcionalmente minimax)