**Tic\_Tac\_Toe game in Python**

This is my **Python** version code for a player board game featuring a human player, **MG**, and a computer player, **AI**. The game uses a 5x5 grid, with each cell initially empty. Here is the design and the explanation of the code.

**Class Board:**

Responsibilities

A screenshot of a computer screen

Description automatically generated

Picture 1: Storing game state.

* **Storing game state:** it maintains the state of the game board using the list of 25 elements representing a 5x5 grid.

A screenshot of a computer screen

Description automatically generated

Picture 2: Displaying the board.

* **Displaying the board:** the display method prints the current state of the board in a grid format.

A screenshot of a computer screen

Description automatically generated

Picture 3: Checking for the winner.

* **Checking for winner:** the winner method checks if any player has achieved a winning combination.

A screenshot of a computer screen

Description automatically generated

Picture 4: Handling the moves.

* **Handling moves:** the add method updates the board with the player’s move if the chosen position is available.

**Class Game:**

Responsibilities

A screenshot of a computer screen

Description automatically generated

Picture 5: The game control flow.

* **Game flow control:** this class orchestrates the overall flow of the game.

A screenshot of a computer screen

Description automatically generated

Picture 6: Resetting the game.

* **Game reset:** the **reset\_game** method resets the game to its initial state.

A screenshot of a computer screen

Description automatically generated

Picture 7: The turn management.

* **Turn management:** It manages players' turns using a list (turn). **In the while loop,** the **play** method coordinates a single play cycle like a display board and take turn.

**Note:** if player MG did not enter the valid number, it will show the invalid move and let the player AI play tell you to enter the correct number(1 to 25).

**Main function:**

Responsibilities:

A screenshot of a computer screen

Description automatically generated

Picture 8: Game initialisation.

* **Game initialisation:** it initialises the game by creating **board** and **game** objectives.

A screenshot of a computer

Description automatically generated

Picture 9: The game looping.

* **Game loop**: controls the main game loop, determining when the game ends and if it should restart, and the Winning condition check will check if either player has won after each turn. When we have no winner, it’s a Tie!

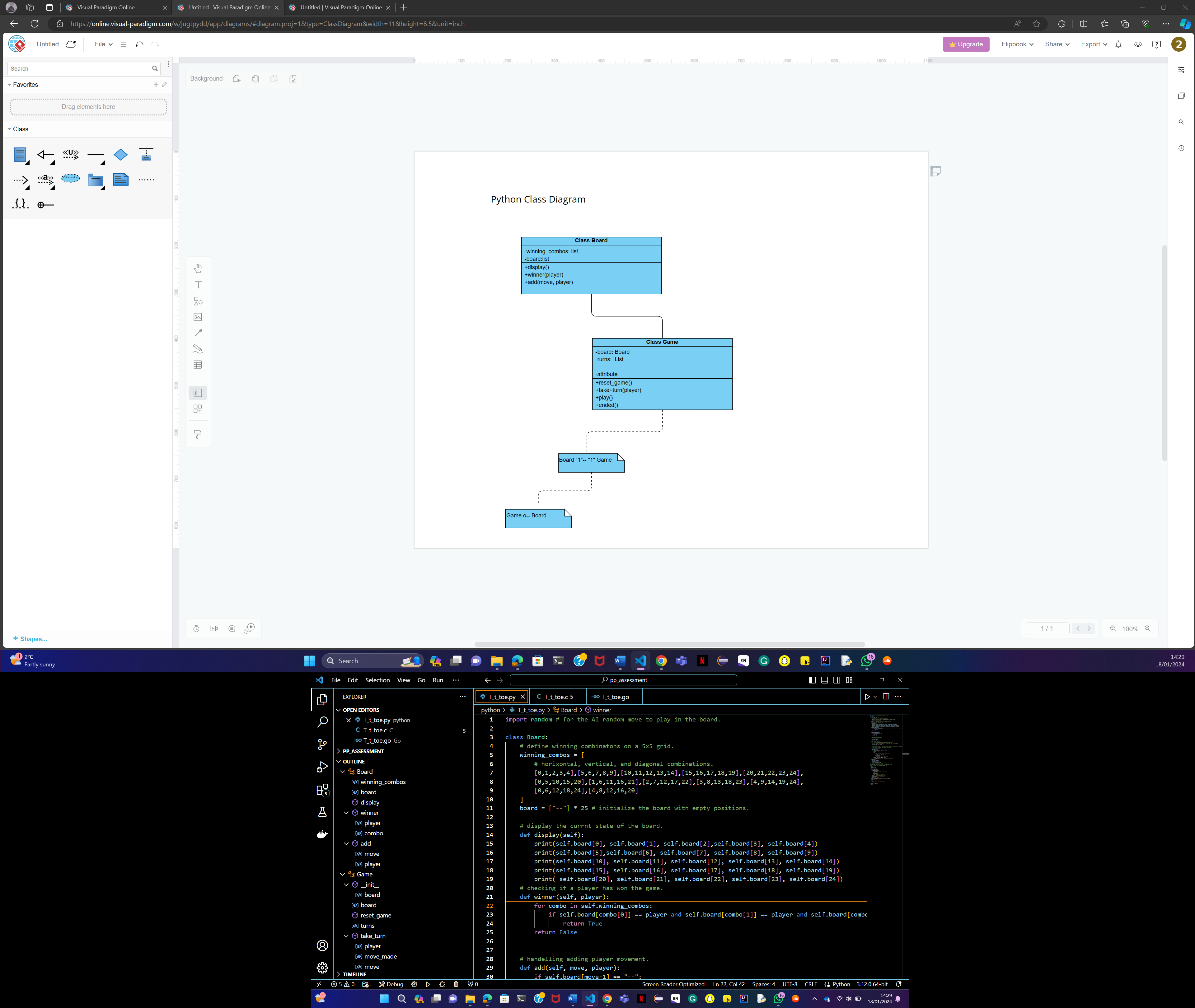
A screenshot of a computer screen

Description automatically generated

Picture 10: The restarting logic.

* **Restart logic**: offers the player a choice to restart the game. Otherwise, end the game.

**Python UML Diagram**



Picture 11: the drawing of the class diagram.