Tictoctoe build

Step1 -Game Criteria and Mode

1. Game Over Conditions

- Player 1 wins if the result is -1, Player 2 wins if the result is 1.
- The game ends in a tie if no valid moves remain.

2. Player Symbols

- Player 1 = 1
- Player 2 = -1

3. Turn Order

- Player 1 plays first by default, but this can be user-defined.

4. Board State

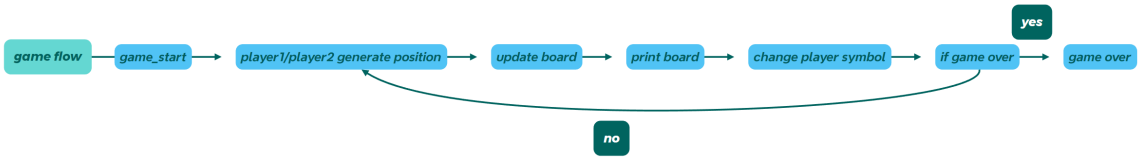
- The board starts clear at the beginning of the game.

5. Agent Modes

- **Random Mode:** The computer randomly selects a position.
- **Q-Learning Mode:** The computer chooses a position using a Q-learning algorithm.
- **User Mode:** The user inputs positions via the command window.

Step2 - Game flow

- The game flow present in following



  - Game Flow Start: The game begins.
  - Game Start: Initialize the game, setting up the board and any necessary settings.

- Player 1/Player 2 Generate Position: Each player takes their turn by selecting a position, depending on the game mode (random, Q-learning, or user input).
- Update Board: Modify the board to reflect the player's move.
- Print Board: Display the current state of the board to the players.
- Change Player Symbol: Switch to the other player's turn.
- If Game Over: Check conditions to determine if the game has ended.
- Yes: The game ends, and the winner is announced, or a tie is declared.
- No: Repeat the process by returning to the generate position step.

Step 3 -Tictactoe in python

Initialization

- `__init__` : Initializes the game with two players, specifying their modes (`'user'` , `'random'` , `'qlearning'`) and symbols (`1` for Player 1, `-1` for Player 2). The `first_player` parameter determines who plays first.

Methods

- `print_game_start` : Clears the board, prints the starting game message, player modes, and the initial board state.

- `print_board` : Displays the current 3x3 board state in a readable format.

- `define_position_available` : Returns a list of indices for positions that are currently unoccupied on the board.

- `reset_board` : Clears the board to its initial state and sets the game as not over.

- `switch_player` : Alternates the current player between Player 1 and Player 2.

- `generate_position` : Generates the next move based on the current player's mode:
  - `user` : Prompts the user to input a position.

    - **`random`** : Selects a position randomly.
    - **`qlearning`** : Uses a placeholder method for Q-learning based move selection.

- **`update_board`** : Sets the current player's symbol at the chosen position on the board.

- **`gameover`** : Announces the result based on the winner (or tie) and indicates the game is complete.

- **`if_game_over`** : Checks for a winning condition or a tie:
    - Evaluates rows, columns, and diagonals for a win.
    - Checks for a tie if no positions are available.
    - Updates `game_over` status and returns the winner.

- **`print_game_result`** : Outputs the result of the game, declaring a draw or a win by a player.

- **`play`** : Manages the game loop, processing each turn until the game ends, then announces the winner.

- In the main section, a game instance is created with specific player modes. However, note that the `train` method is called, which is not defined in the class and should be removed or implemented if necessary.

Step 4 - Qlearning implement in tictactoe

Q learning algorithm:

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

1. **Action Selection**:
    - The agent selects an action based on the current state to maximize the expected reward.

$$\pi^* = \arg\max Q^*(s,a)$$

2. **New State and Reward**:
    - The agent performs the action, producing a new state and receiving a reward.

3. **Q-Value Update**:
    - The Q-value for the specific state-action pair is updated using the formula:

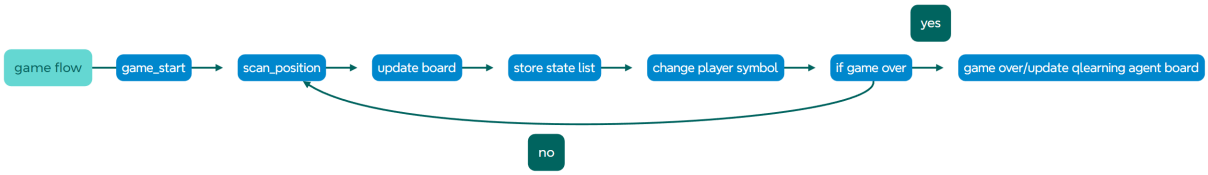
$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a)\right]$$

where:

- α is the learning rate
- R_{t+1} is the reward received after taking action (a_t).
- γ is the discount factor, determining the importance of future rewards.
- $\max_a Q(s_{t+1}, a)$ is the maximum expected future reward for the next state (s_{t+1})

Add workflow to tictactoe game

- we need to add two step in tictactoe game flow: store state list ,update board, these function build in agent class



- in training workflow we need to add two step store current board, action to agent state list,and update q learning board in game over

- in agent we need to add 3 function store state list, update board, choose q learning action

Different in train,test -exploration rate:

- Training Mode : With a probability defined by exploration_rate, randomly choose an action (exploration).
  Otherwise, select the action with the highest Q-value from the Q-table (exploitation).
  Decay the exploration rate after action selection.

- Testing Mode :
  Select the action with the highest Q-value from the Q-table for the given board state.

Game demo: