TIC TAC TOE SOLVER REPORT

By Harshita Kumari
Roll No: 2024011100300119
Course: BTECH- Computer Science Engineering with Artificial Intelligence
Institution: KIET Group of Institutions

Date: 11 MARCH 2025

Introduction

The Tic Tac Toe Solver is a Python-based program that simulates a simple game of Tic Tac Toe. In this implementation, two players make moves by randomly selecting an available spot on a 3×3 board until a win condition is met or the game results in a tie. The program uses basic NumPy arrays to represent the board, evaluates win conditions (rows, columns, and diagonals), and prints the board state after every move. The objective of this project is to demonstrate the use of fundamental programming constructs such as loops, conditionals, and functions, while also providing an example of simple game AI logic.

Methodology

- Board Creation:
- The create_board() function initializes a 3×3 NumPy array filled with zeros. This represents an empty board where 0 indicates an unoccupied space.
- Determining Possible Moves:
- The possibilities(board) function scans the board and returns a list of all empty positions (cells with a 0).
- Placing a Move:
- The random_place(board, player) function selects a random empty cell from the list provided by possibilities and marks it with the player's number (1 or 2).
- Checking for Win Conditions:
- Three separate functions (row_win, col_win, and diag_win) check whether a player has filled an entire row, column, or diagonal with their marker. These functions iterate over the board and use boolean checks to validate if the player has won.
- Evaluating the Game State:
- The evaluate(board) function checks whether any player has met the win condition or if the board is completely filled (indicating a tie).
- Game Simulation:
- The play_game() function drives the game by alternating between the two players. After each move, it prints the current state of the board and evaluates whether there is a winner or if the game has ended in a tie.

Code Typed

```
import numpy as np
import random

# Creates an empty 3x3 Tic-Tac-Toe board
def create_board():
 return np.zeros((3, 3), dtype=int)

# Returns a list of available empty positions on the board
def possibilities(board):
 return [(i, j) for i in range(3) for j in range(3) if board[i, j] == 0]

# Allows the user to input their move
def user_place(board, player):
 while True:
 try:
 row, col = map(int, input("Enter row and column (0-2) separated by space: ").split())
 if (row, col) in possibilities(board):
 board[row, col] = player # Place the user's move
 break
 else:
 print("Invalid move! Try again.")
 except ValueError:
 print("Invalid input! Enter two numbers between 0 and 2.")
 return board

# AI places a random move
def random_place(board, player):
 row, col = random.choice(possibilities(board)) # Select a random empty spot
 board[row, col] = player
 return board
```

```
# Checks if a player has won the game
def check_win(board, player):
 return any(all(board[i, j] == player for j in range(3)) for i in range(3))
or \
 any(all(board[j, i] == player for j in range(3)) for i in range(3)) or \
 all(board[i, i] == player for i in range(3)) or \
 all(board[i, 2 - i] == player for i in range(3))

# Evaluates the game state: returns the winner (1 or 2), -1 for a tie,
or 0 if the game is ongoing
def evaluate(board):
 for player in [1, 2]:
 if check_win(board, player):
 return player
 return -1 if not possibilities(board) else 0

# Main function to play the game
def play_game():
 board, winner = create_board(), 0 # Initialize the game board
 print(board)
 while winner == 0:
 board = user_place(board, 1) # User's turn
 print(board)
 winner = evaluate(board)
 if winner != 0:
 break
 board = random_place(board, 2) # AI's turn
 print("AI Move:")
 print(board)
 winner = evaluate(board)
 print("Winner is:", "User" if winner == 1 else "AI" if winner == 2 else
"It's a tie!")

play_game()
```

output

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Enter row and column (0-2) separated by space: 1 1
[[0 0 0]
 [0 1 0]
 [0 0 0]]
AI Move:
[[0 0 0]
 [0 1 0]
 [0 0 2]]
Enter row and column (0-2) separated by space: 0 0
[[1 0 0]
 [0 1 0]
 [0 0 2]]
AI Move:
[[1 2 0]
 [0 1 0]
 [0 0 2]]
Enter row and column (0-2) separated by space: 2 0
[[1 2 0]
 [0 1 0]
 [1 0 2]]
AI Move:
[[1 2 0]
 [2 1 0]
 [1 0 2]]
Enter row and column (0-2) separated by space: 2 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 1
Invalid move! Try again.
Enter row and column (0-2) separated by space: 0 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 2
```

```
[[1 2 0]
 [2 1 0]
 [1 0 2]]
Enter row and column (0-2) separated by space: 2 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 1
Invalid move! Try again.
Enter row and column (0-2) separated by space: 0 0
Invalid move! Try again.
Enter row and column (0-2) separated by space: 1 2
[[1 2 0]
 [2 1 1]
 [1 0 2]]
AI Move:
[[1 2 0]
 [2 1 1]
 [1 2 2]]
Enter row and column (0-2) separated by space: 2 1
Invalid move! Try again.
Enter row and column (0-2) separated by space: 0 2
[[1 2 1]
 [2 1 1]
 [1 2 2]]
Winner is: User
```