TIC TAC TOE SOLVER

Name: Harshita Kumari
Branch: CSE AI
Section: B
Roll No: 45

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
from time import sleep

# Creates an empty board
def create_board():
 return(np.array([[0, 0, 0],
 [0, 0, 0],
 [0, 0, 0]]))

# Check for empty places on board
def possibilities(board):
 l = []

 for i in range(len(board)):
 for j in range(len(board)):

 if board[i][j] == 0:
 l.append((i, j))
 return(l)

# Select a random place for the player
def random_place(board, player):
 selection = possibilities(board)
 current_loc = random.choice(selection)
 board[current_loc] = player
 return(board)

# Checks whether the player has three
# of their marks in a horizontal row
def row_win(board, player):
 for x in range(len(board)):
 win = True

 for y in range(len(board)):
 if board[x, y] != player:
 win = False
 continue

 if win == True:
 return(win)
 return(win)
```

```
def col_win(board, player):
 for x in range(len(board)):
 win = True

 for y in range(len(board)):
 if board[y][x] != player:
 win = False
 continue

 if win == True:
 return(win)
 return(win)

# Checks whether the player has three
# of their marks in a diagonal row
def diag_win(board, player):
 win = True
 y = 0
 for x in range(len(board)):
 if board[x, x] != player:
 win = False
 if win:
 return win
 win = True
 if win:
 for x in range(len(board)):
 y = len(board) - 1 - x
 if board[x, y] != player:
 win = False
 return win
```

```
# Evaluates whether there is
# a winner or a tie
def evaluate(board):
winner = 0

for player in [1, 2]:
if (row_win(board, player) or
col_win(board, player) or
diag_win(board, player)):

winner = player

if np.all(board != 0) and winner == 0:
winner = -1
return winner

# Main function to start the game
def play_game():
board, winner, counter = create_board(), 0, 1
print(board)
sleep(2)

while winner == 0:
for player in [1, 2]:
board = random_place(board, player)
print("Board after " + str(counter) + " move")
print(board)
sleep(2)
counter += 1
winner = evaluate(board)
if winner != 0:
break
return(winner)
print("Winner is: " + str(play_game()))
```

output

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 1 0]
 [0 0 0]
 [0 0 0]]
Board after 2 move
[[2 1 0]
 [0 0 0]
 [0 0 0]]
Board after 3 move
[[2 1 0]
 [0 0 1]
 [0 0 0]]
Board after 4 move
[[2 1 2]
 [0 0 1]
 [0 0 0]]
Board after 5 move
[[2 1 2]
 [1 0 1]
 [0 0 0]]
Board after 6 move
[[2 1 2]
 [1 0 1]
 [0 2 0]]
Board after 7 move
[[2 1 2]
 [1 1 1]
 [0 2 0]]
Winner is: 1
```