S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR

Practical No. 03

Aim:- Build a Tic-Tac-Toe Game Engine with an AI Player in Python.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: V/III

Academic Session: 2024-2025

Date of Performance:

Date of Submission:

AIM: Build a Tic-Tac-Toe Game Engine with an AI Player in Python.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To be able to understand the concept of Tic-Tac-Toe.
- To be able to acquire the puzzle solving capability.
- To be able to acquire the game development ability.

THEORY:

Tic-tac-toe is a simple and popular two-player game where the objective is to get three of your symbols in a row on a 3x3 grid. The game is typically played with Xs and Os. Here's how it works:

How to Play:

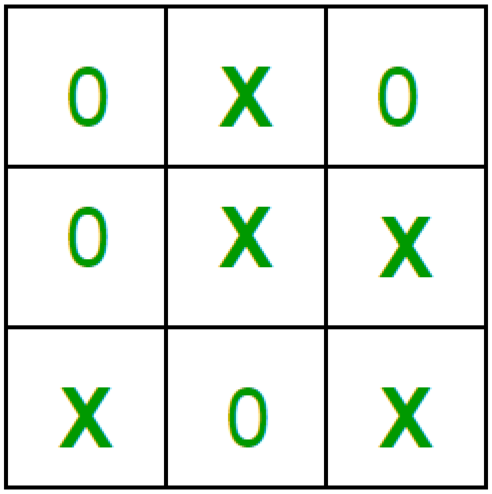
1. **Players:** Two players take turns, one playing as "X" and the other as "O."
2. **Grid:** The game board consists of a 3x3 grid.
3. **Objective:** The first player to place three of their marks (either Xs or Os) in a horizontal, vertical, or diagonal row wins the game.
4. **Turns:** Players take turns placing their mark in an empty square on the grid.
5. **Winning:** The game ends when one player gets three of their marks in a row or when all squares are filled, resulting in a draw if no one has won.

Strategies:

- **First move advantage:** Going first can be an advantage if played strategically.
- **Blocking:** Always watch your opponent's moves and block them if they are about to get three in a row.
- **Center square:** The center square is often the best first move, as it provides more opportunities to win.

Rules of the Game

- The game is to be played between two people (in this program between HUMAN and COMPUTER).
- One of the player chooses 'O' and the other 'X' to mark their respective cells.
- The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X').
- If no one wins, then the game is said to be draw



Implementation In our program the moves taken by the computer and the human are chosen randomly. We use rand() function for this. **What more can be done in the program?** The program is in not played optimally by both sides because the moves are chosen randomly. The program can be easily modified so that both players play optimally (which will fall under the category of Artificial Intelligence). Also the program can be modified such that the user himself gives the input (using scanf() or cin). The above changes are left as an exercise to the readers. **Winning Strategy – An Interesting Fact** If both the players play optimally then it is destined that you will never lose (“although the match can still be drawn”). It doesn’t matter whether you play first or second.In another ways – “ Two expert players will always draw ”. Isn’t this interesting ?

PROGRAM CODE:

ALGORITHM:

 Initialize the Game Board:

- Create a 3x3 grid (a 2D array or list of lists) filled with empty values.

 Display the Game Board:

- Create a function to print the current state of the grid to the console.

 Set the Players:

- Assign one player as "X" and the other as "O."
- Decide randomly or allow one player to choose who goes first.

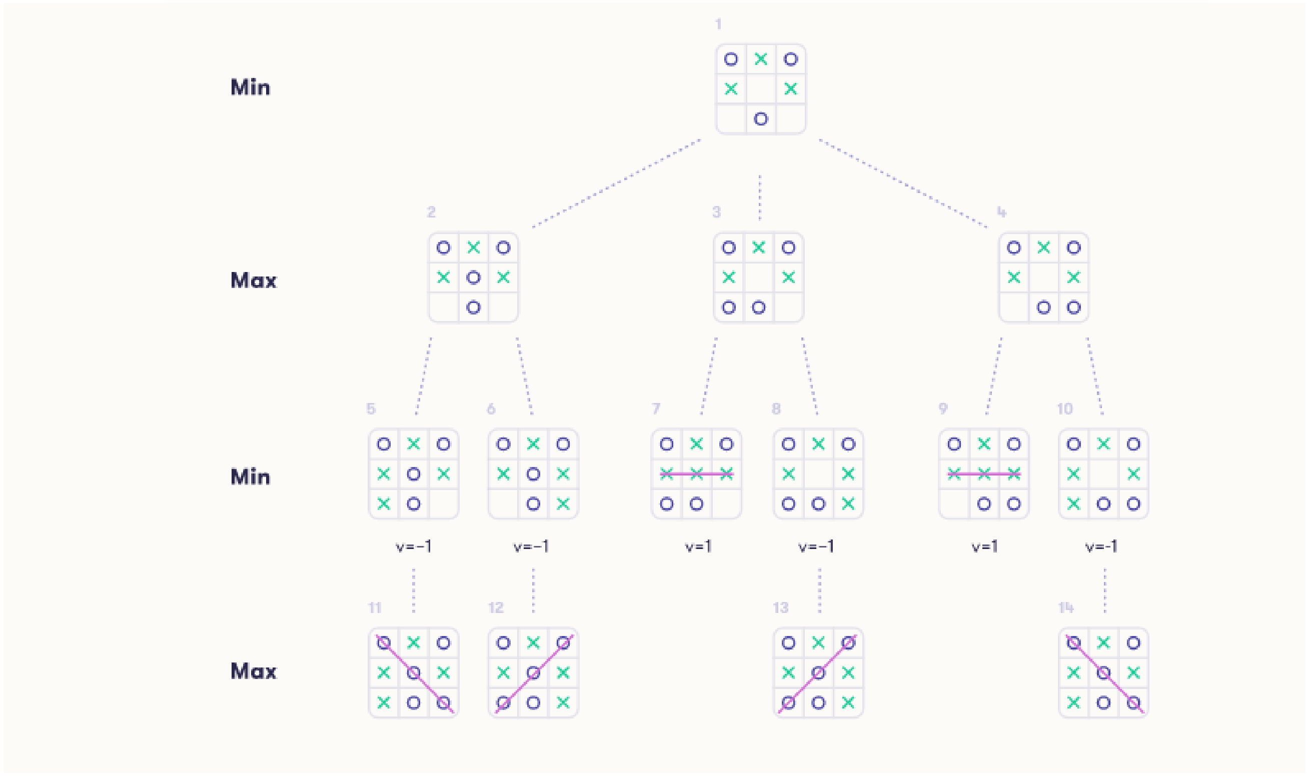
 Main Game Loop:

- Loop until the game ends (either a player wins or the grid is full):
    1. **Display the Board:**
        - Print the current state of the grid.
    2. **Player Input:**
        - Prompt the current player to select a row and column (1-3) for their move.
        - Validate the input to ensure the chosen cell is empty and within bounds.
    3. **Update the Board:**
        - Place the current player's symbol (X or O) in the selected cell.
    4. **Check for a Win:**
        - Create a function to check if the current player has won:
            - Check all rows, columns, and both diagonals to see if any have three of the same symbols.
    5. **Check for a Draw:**
        - If the grid is full and no one has won, declare a draw.
    6. **Switch Players:**
        - Alternate between players (X -> O, O -> X).

 End the Game:

- If a player wins, announce the winner.
- If the game ends in a draw, announce that the game is tied.

Search Space:
Consider nodes (5)–(10) on the second level from the bottom. In nodes (7) and (9), the game is over, and Max wins with three X's in a row. The value of these positions is +1. In the remaining nodes, (5), (6), (8), and (10), the game is also practically over, since Min only needs to place her O in the only remaining cell to win. In other words, we know how the game will end at each node on the second level from the bottom. We can therefore decide that the value of nodes (5), (6), (8), and (10) is also –1.



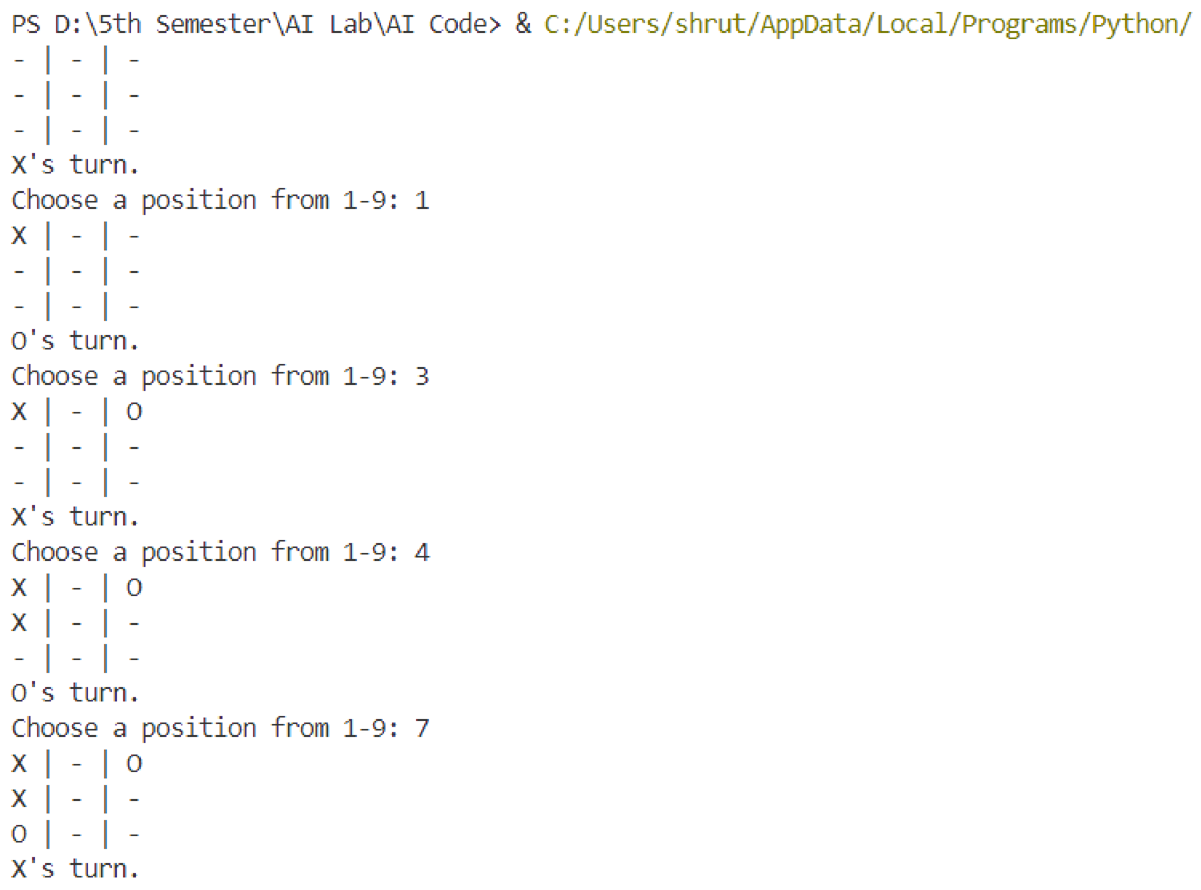
Code:-

Complete code:

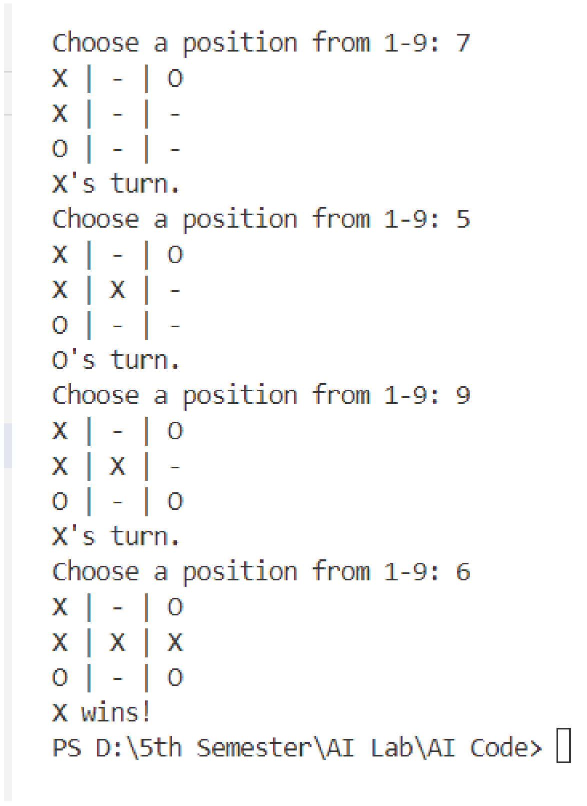
```
# Set up the game board as a list
board = ["-", "-", "-",
         "-", "-", "-",
         "-", "-", "-"]

# Define a function to print the game board
def print_board():
    print(board[0] + " | " + board[1] + " | " + board[2])
    print(board[3] + " | " + board[4] + " | " + board[5])
    print(board[6] + " | " + board[7] + " | " + board[8])

# Define a function to handle a player's turn
def take_turn(player):
    print(player + "'s turn.")
    position = input("Choose a position from 1-9: ")
    while position not in ["1", "2", "3", "4", "5", "6", "7", "8",
"9"]:
        position = input("Invalid input. Choose a position from 1-
9: ")
    position = int(position) - 1
    while board[position] != "-":
        position = int(input("Position already taken. Choose a
different position: ")) - 1
    board[position] = player
    print_board()
```

```
# Define a function to check if the game is over
def check_game_over():
    # Check for a win
    if (board[0] == board[1] == board[2] != "-") or \
       (board[3] == board[4] == board[5] != "-") or \
       (board[6] == board[7] == board[8] != "-") or \
       (board[0] == board[3] == board[6] != "-") or \
       (board[1] == board[4] == board[7] != "-") or \
       (board[2] == board[5] == board[8] != "-") or \
       (board[0] == board[4] == board[8] != "-") or \
       (board[2] == board[4] == board[6] != "-"):
        return "win"
    # Check for a tie
    elif "-" not in board:
        return "tie"
    # Game is not over
    else:
        return "play"

# Define the main game loop
def play_game():
    print_board()
    current_player = "X"
    game_over = False
    while not game_over:
        take_turn(current_player)
        game_result = check_game_over()
        if game_result == "win":
            print(current_player + " wins!")
            game_over = True
        elif game_result == "tie":
            print("It's a tie!")
            game_over = True
        else:
            # Switch to the other player
            current_player = "O" if current_player == "X" else "X"

# Start the game
play_game()
```

INPUT & OUTPUT:



```
PS D:\5th Semester\AI Lab\AI Code> & C:/Users/shrut/AppData/Local/Programs/Python/
- | - | -
- | - | -
- | - | -
X's turn.
Choose a position from 1-9: 1
X | - | -
- | - | -
- | - | -
O's turn.
Choose a position from 1-9: 3
X | - | O
- | - | -
- | - | -
X's turn.
Choose a position from 1-9: 4
X | - | O
X | - | -
- | - | -
O's turn.
Choose a position from 1-9: 7
X | - | O
X | - | -
O | - | -
X's turn.
```



```
Choose a position from 1-9: 7
X | - | O
X | - | -
O | - | -
X's turn.
Choose a position from 1-9: 5
X | - | O
X | X | -
O | - | -
O's turn.
Choose a position from 1-9: 9
X | - | O
X | X | -
O | - | O
X's turn.
Choose a position from 1-9: 6
X | - | O
X | X | X
O | - | O
X wins!
PS D:\5th Semester\AI Lab\AI Code>
```

CONCLUSION:

Thus, build a Tic-Tac-Toe Game Engine with an AI Player in Python.

DISCUSSION QUESTIONS:

- Which Technique is best for above game?
- Give the best rules to win the game.
- Give the best rules to draw the game.

REFERENCES:

- https://course.elementsofai.com/2/3
- https://www.geeksforgeeks.org/implementation-of-tic-tac-toe-game/
- https://www.geeksforgeeks.org/low-level-design-of-tic-tac-toe-system-design/
- https://react.dev/learn/tutorial-tic-tac-toe