***FUTURENSE TECHNOLOGIES INTERNSHIP***

TIC-TAC-TOE

By Josh Hanley Thomas, Abhilash Santhosh, Arun Siby

# INTRODUCTION

The Tic-Tac-Toe game is a well-known and straightforward two-player game that involves placing 'X' or 'O' symbols on a 3x3 grid. This project focuses on implementing a console-based version of Tic-Tac-Toe in Python. Players take turns making moves until there is a winner with three symbols in a row, column, or diagonal, or the game ends in a tie when the entire board is filled.

# IMPLEMENTATION

The game board is represented as a 3x3 grid using a 2D list, with each cell initially containing an empty space (' '). A `print\_board` function is created to visually display the current state of the board in the console. Players interact with the game by inputting row and column indices to make their moves. Input validation ensures that the entered move is within the board boundaries and that the selected cell is unoccupied.

The game logic is encapsulated within the `tic\_tac\_toe` function, which serves as the main game loop. Players alternate turns, and after each move, the game checks for a winner by examining rows, columns, and diagonals. If a winner is found, the game announces the winning player. If the board is full with no winner, the game declares a tie.

# CODE

|  |
| --- |
| **def** **print\_board**(board):  **for** row **in** board:  print(" | ".join(row))  print("-" \* 5)  **def** **check\_winner**(board):  # Check rows  **for** row **in** board:  **if** all(cell == row[0] **and** cell != ' ' **for** cell **in** row):  **return** **True**   # Check columns  **for** col **in** range(3):  **if** all(board[row][col] == board[0][col] **and** board[row][col] != ' ' **for** row **in** range(3)):  **return** **True**   # Check diagonals  **if** all(board[i][i] == board[0][0] **and** board[i][i] != ' ' **for** i **in** range(3)) **or** \  all(board[i][2 - i] == board[0][2] **and** board[i][2 - i] != ' ' **for** i **in** range(3)):  **return** **True**   **return** **False**  **def** **is\_board\_full**(board):  **return** all(cell != ' ' **for** row **in** board **for** cell **in** row)  **def** **tic\_tac\_toe**():  board = [[' ' **for** \_ **in** range(3)] **for** \_ **in** range(3)]  current\_player = 'X'   **while** **True**:  print\_board(board)   # Get player move  **while** **True**:  row = int(input(f"Player {current\_player}, enter row (0, 1, or 2): "))  col = int(input(f"Player {current\_player}, enter column (0, 1, or 2): "))   **if** 0 <= row < 3 **and** 0 <= col < 3 **and** board[row][col] == ' ':  board[row][col] = current\_player  **break**  **else**:  print("Invalid move. Try again.")   # Check for a winner  **if** check\_winner(board):  print\_board(board)  print(f"Player {current\_player} wins!")  **break**   # Check for a tie  **if** is\_board\_full(board):  print\_board(board)  print("It's a tie!")  **break**   # Switch player  current\_player = 'O' **if** current\_player == 'X' **else** 'X'  **if** \_\_name\_\_ == "\_\_main\_\_":  tic\_tac\_toe() |

# LOGIC

1. Main Game Loop:

The game revolves around a continuous loop that executes until a winner is determined or the board is full. Players make moves, and the game alternates between displaying the current state of the board and prompting players for their input.

2. Player Moves:

Players input their moves by specifying row and column indices. The game ensures that the move is valid by checking if it is within the board boundaries and if the selected cell is unoccupied.

3. Checking for a Winner:

After each move, the game checks for a winner by examining the rows, columns, and diagonals of the board. It looks for a sequence of three identical symbols in a row, column, or diagonal to declare a winner.

4. Checking for a Tie:

To determine a tie, the game checks if the board is completely filled with symbols and no winner has been declared. If all cells are occupied and no winner is found, the game concludes as a tie.

5. Switching Players:

After each move, the current player is switched to the other symbol ('X' to 'O' or vice versa). This ensures that players alternate turns throughout the game.

6. Conclusion of the Game:

The game loop concludes when either a winner is determined or the board is full, resulting in a tie. The game announces the winner or declares a tie and then exits the loop.

# CONCLUSION

In conclusion, this implementation of Tic-Tac-Toe in Python provides a fundamental understanding of game development concepts. The game offers an interactive and engaging experience for players while adhering to the classic rules of Tic-Tac-Toe. It serves as a foundation for further exploration into more advanced game development concepts and features.