**TITLE:** CodTech IT Solutions Internship - Task Documentation: TIC TAC TOE

**INTERN INFORMATION:**

**Name:** AYUSH M C

**ID:** ICOD7066

**INTRODUCTION**

The provided program is a Python implementation of a simple tic-tac-toe game playable in the console. Tic-tac-toe, also known as noughts and crosses or Xs and Os, is a two-player game where players take turns marking spaces in a 3x3 grid with their respective symbols, typically 'X' and 'O'. The objective of the game is to be the first player to form a horizontal, vertical, or diagonal line of three of their symbols.

This program allows two players to take turns making moves by entering the row and column of the grid where they wish to place their symbol. The game continues until one player wins by forming a line of three symbols, or the entire grid is filled without a winner, resulting in a tie.

The main features of the program include:

1. Displaying the current state of the board after each move.

2. Validating player input to ensure it corresponds to a valid grid position.

3. Checking for a winner after each move to determine if the game should continue or end.

4. Providing an option for players to play again after the game concludes.

The program utilizes functions to organize different aspects of the game, such as displaying the board, checking for a winner, and determining if the board is full. It employs a while loop to continuously prompt players for input and update the game state until a winner is determined or the game ends in a tie.

Overall, this program provides a basic implementation of the tic-tac-toe game, demonstrating fundamental concepts of input handling, game logic, and flow control in Python.

**Implementation**

* The game board is represented as a 3x3 grid using a list of lists.
* Players are represented by 'X' and 'O', and their moves are alternated using a current player index.
* Input validation ensures that players enter valid row and column indices within the range [0, 2].
* The **check\_winner()** function checks for winning conditions by examining rows, columns, and diagonals.
* The **is\_board\_full()** function checks if all spaces on the board are filled.
* Recursion is used to allow players to play multiple games in succession.

**CODE EXPLAINATION**

This program allows two players to play the classic game of Tic-Tac-Toe on a 3x3 grid. Players take turns marking spaces on the grid with their respective symbols ('X' or 'O'), aiming to create a horizontal, vertical, or diagonal line of their symbols before their opponent.

1. display\_board(board)` function:

- This function takes the current state of the board as input and prints it to the console, displaying the current positions of 'X' and 'O' marks.

- It separates the cells with '|' symbols and adds horizontal lines to represent the grid.

2. check\_winner(board)` function:

- This function checks the current state of the board to determine if a player has won the game.

- It examines rows, columns, and diagonals to see if there is a complete line of 'X's or 'O's.

- If a winning condition is found, it returns the winning player's symbol ('X' or 'O'); otherwise, it returns `None`.

3. is\_board\_full(board)` function:

- This function checks if the board is full, indicating that the game has ended in a tie.

- It iterates through the board to see if there are any empty spaces (' ') left.

- If the board is full, it returns `True`; otherwise, it returns `False`.

4. main()` function:

- This is the main function that orchestrates the game.

- It initializes the game board, defines player symbols ('X' and 'O'), and sets the current player to Player 1 ('X').

- It enters a loop where players take turns making moves until a winner is determined or the board is full.

- After each move, it checks for a winner or a tie and updates the current player.

- Once the game ends, it prompts the players if they want to play again and restarts if they agree.

Note:

- Error handling is implemented to ensure valid player input for moves.

- Debugging print statements are included to assist in troubleshooting potential issues with player input and board state.

**USAGE**

1. Game Initialization:

- Upon running the program, the game initializes with an empty 3x3 board.

- Players are represented by 'X' and 'O'.

2. Gameplay:

- Players take turns entering their moves by specifying the row and column numbers where they want to place their symbol.

- Rows and columns are zero-indexed, meaning they range from 0 to 2.

- Players input their moves in the format "row col", separated by a space.

- For example, entering "1 2" would place the current player's symbol in the second row and third column of the board.

3. Error Handling:

- The program validates user inputs to ensure they consist of two integers separated by a space.

- It also checks if the specified row and column are within the valid range (0 to 2) and if the chosen position is already occupied.

4. Game End Conditions:

- The game ends when one player wins by completing a row, column, or diagonal with their symbol.

- If no player wins and the board is full, the game declares a tie.

- After the game ends, players are prompted to play again or exit.

5. Play Again:

- Players have the option to restart the game by entering "yes" when prompted.

- Entering any other input will exit the program with a farewell message.

6. Note:

- This program is designed for two human players. It does not include an AI opponent.

- It runs entirely on the command line interface, providing a simple and accessible gaming experience.

.**CONCLUSION**

This program is a simple implementation of the classic game Tic-Tac-Toe. Players take turns entering their moves as row and column coordinates, trying to get three of their symbols in a row, column, or diagonal. The program checks for a winner after each move and ends the game when there's a winner or the board is full (resulting in a tie). Players can choose to play again after the game ends. Overall, it provides a fun and interactive way to play Tic-Tac-Toe against another player on the computer.

**OUTPUT**





