TIC-TAC-TOE

Title: The TIC-TAC-TOE project is a python-based command-line game for two players. Players take turns marking spaces in a 3\*3 grid with ‘X’ or ‘0’.

The first player to align three marks horizontally, vertically, or diagonally wins the game. If all cells are filled and no player wins , the game ends in a row.

ABSTRACT

A Tic-Tac-Toe game project in python involves creating a program that allows two players to play the classic game.

The core components include:

Game Board Representation:

A list or a 2D array is used to represent the TIC-TAC-TOE board. Each element in the list/array corresponds to a cell on the board, and stores either ‘X’ or ‘0’ or an empty space(‘ ‘).

Player Input:

The program takes input from players to determine their moves. This involves prompting players to enter the position (e. g.., row and column number or a number from 0 to 9 ) where they want to place their mark.

Move Validation:

Before updating the game board, the program checks if the move is valid. A valid move means the selected cell in within the boards boundaries and is currently empty.

Updating The Game Board:

If the move is valid , the program updates the game board by placing the players mark(‘ X ’ or ‘ 0 ’) in the chosen cell.

Checking For The Winner:

After each move, the program checks if there is a winner. It examines all possible winning combinations (rows, columns, diagonals) to see if any player has three of their marks in a row.

Checking For A Tie:

If no player has won and all the cells on the board are filled, the game is declared as a tie.

Game Loop:

The game continues until either a player wins or the game ends in a tie. The game loop handles player turns, input, move validation, board updates and win/tie checking.

User Interface:

The user interface can be text-based(using the console) or graphical(using libraries like Tkinter or pygame). A text-based interface displays the board and prompts for input in the console. A graphical interface provides a visual representation of the board and allows players to interact with it using the mouse.

Artificial Intelligence:

For a single player mode, an AI opponent can be implemented. This can range from simple random move selection to more advanced algorithms like Minimax to ensure a challenging game.

INTRODUCTION

TIC-TAC-TOE is not a very challenging game for human beings. If you are an enthusiast, you have probably moved from the basic game to some variant like three dimensional tic-tac-toe on a larger grid. If you sit down right now to play ordinary three-by three tic-tac-toe with a friend , what will probably will happen is that every game will come out a tie. Both you and your friend can probably play perfectly, never making a mistake that would allow your opponent to win. Most of the time you probably aren’t even aware of alternative possibilities; you just look at the board and instantly know where you want to move. That kind of instant knowledge is great for human beings, because it makes you a fast player. But it isn’t much help in writing a computer program. For that, you have to know very explicitly what your strategy is.

The highest and the lowest priority rules seemed obvious to me right away. The highest priority are these:

1. If I can win on this move, do it.
2. If the other player can win on the next move, block that winning square.

Here are the lowest priority rules, used only if there is nothing suggested more strongly by the board position:

1. N-2 take the center square if its free
2. N-1 take a corner square if one is false
3. N take whatever is available

IMPLEMENTATION

* Print\_ board(board):

Displays the current state of the board in a formatted way.

* Check\_ win(board, player):

Checks if the current player has three of their marks in a row, column ,diagonal.

* Check\_ draw(board):

Checks if the board is full without any winner(resulting in a draw)

* Tic\_ Tac\_ Toe():

Main function that runs the game:

1. Initializes an empty board
2. Alternates turns between ‘ x ’ and ‘ 0 ’
3. Handles player inputs with validation
4. Checks for a win or a draw after each move

Program Flow:

1. Initialization:

* Create an empty 3\*3 board filled with spaces ’ ‘
* Set the first player

1. Game Loop:

* Display the board
* Ask the current player for a row and column number
* Validate the input (check if it’s a number, within bounds, and if the cell is empty)
* Update the board with the players move
* Check if the current move wins the game or if the board is full(draw)
* If neither , switch to the other player

1. End Game:

* Print the final board
* Announce the winner or declare a draw

Program:

def print\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 5)

def check\_win(board, player):

# Check rows, columns and diagonals

for row in board:

if all(cell == player for cell in row):

return True

for col in range(3):

if all(board[row][col] == player for row in range(3)):

return True

if all(board[i][i] == player for i in range(3)):

return True

if all(board[i][2-i] == player for i in range(3)):

return True

return False

def check\_draw(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)

print(f"Player {current\_player}'s turn")

try:

row = int(input("Enter row (0, 1, 2): "))

col = int(input("Enter column (0, 1, 2): "))

except ValueError:

print("Invalid input! Please enter a number.")

continue

if row not in range(3) or col not in range(3):

print("Invalid position! Try again.")

continue

if board[row][col] != ' ':

print("Cell already taken! Try another one.")

continue

board[row][col] = current\_player

if check\_win(board, current\_player):

print\_board(board)

print(f"Player {current\_player} wins!")

break

if check\_draw(board):

print\_board(board)

print("It's a draw!")

break

# Switch player

current\_player = 'O' if current\_player == 'X' else 'X'

# Run the game

tic\_tac\_toe()

OUTPUT

Sample Gameplay:

| |

-----

| |

-----

| |

Player X's turn

Enter row (0, 1, 2): 0

Enter column (0, 1, 2): 0

X| |

-----

| |

-----

| |

Player O's turn

Enter row (0, 1, 2): 1

Enter column (0, 1, 2): 1

X| |

-----

|O|

-----

| |

...

Player X wins!

CONCLUSION

The TIC\_ TAC\_ TOE game project successfully demonstrates the fundamental concepts of python programming, including functions, loops, and user input handling. Through this project, we created a fully functional two player game that handles player turns checks for winning conditions, and detects draws efficiently.

It also sets a strong foundation for more advanced improvements such as AI integration, Graphical User Interfaces(GUI’s), or expanding the board size.

Overall, the TIC-TAC-TOE project serves as an excellent beginner – level programming exercise that strengthens problem-solving skills and logical thinking in python.