TIC TAC TOE

A PROJECT REPORT

SUBMITTED BY: PRAVEEN GURJAR
ROLL NUMBER: 25BCG10032

SUBJECT: FUNDAMENTAL IN AI/ML

Introduction

The game of Tic-Tac-Toe is an example of a simple task which can be programmed in Python by a beginner.

Atypical Loops, Functions, Conditional statements and Lists are the main features of Python language that are illustrated in the project to have a working game.

Problem Statement

Design and develop a functional Python program that allows two human players to play Tic-Tac-Toe on a 3×3 grid. The system verifies the moves, updates the board, detects the winner, and determines if the match ends in a draw.

Project Overview

Tic-Tac-Toe is a classic game played on a 3×3 grid between two players. Player X and Player O take turns

We first prompt the users to select the grid positions on 3×3 by entering numbers between 1 and 9. The cell is then put in

The board is updated after every input, and the program verifies a player's win.

Functional Requirements

FR1 – Player Input Module

- Accepts moves (1–9) from Player X and Player O.
- Validates if the position is empty.

FR2 – Board Management Module

- Stores and updates the 3×3 grid using a list.
- Displays board after each turn.

FR3 – Winner/Draw Evaluation Module

- Checks 8 winning combinations (3 rows, 3 columns, 2 diagonals)
  TicTacToe_Project_Report
- Declares winner or draw

Non-Functional Requirements

1. **Usability:**
   The game must be easy to understand, with clear prompts.
2. **Reliability:**
   Should correctly identify all winning patterns and avoid invalid moves.
3. **Performance:**
   The program should respond instantly as it is lightweight.
4. **Maintainability:**

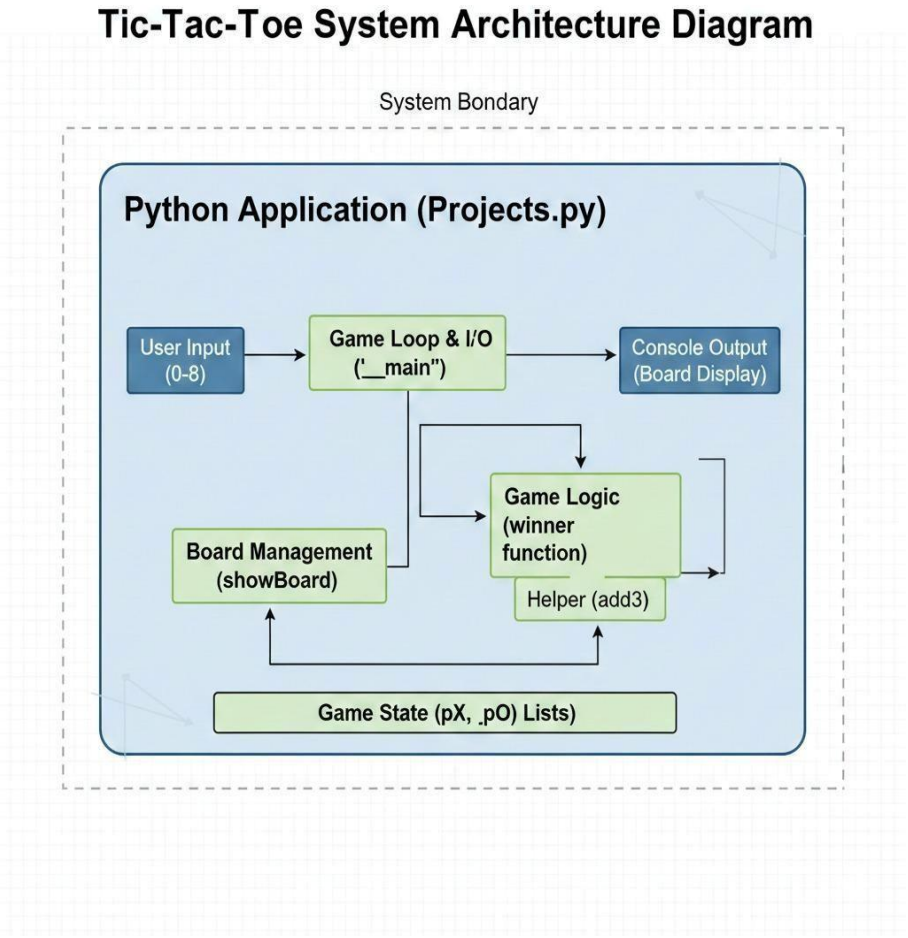
Code is modular with separate functions for board display and winner checking.

5. **Error-Handling:**
   Prevents invalid inputs (non-numbers / already occupied cells).

System Architecture

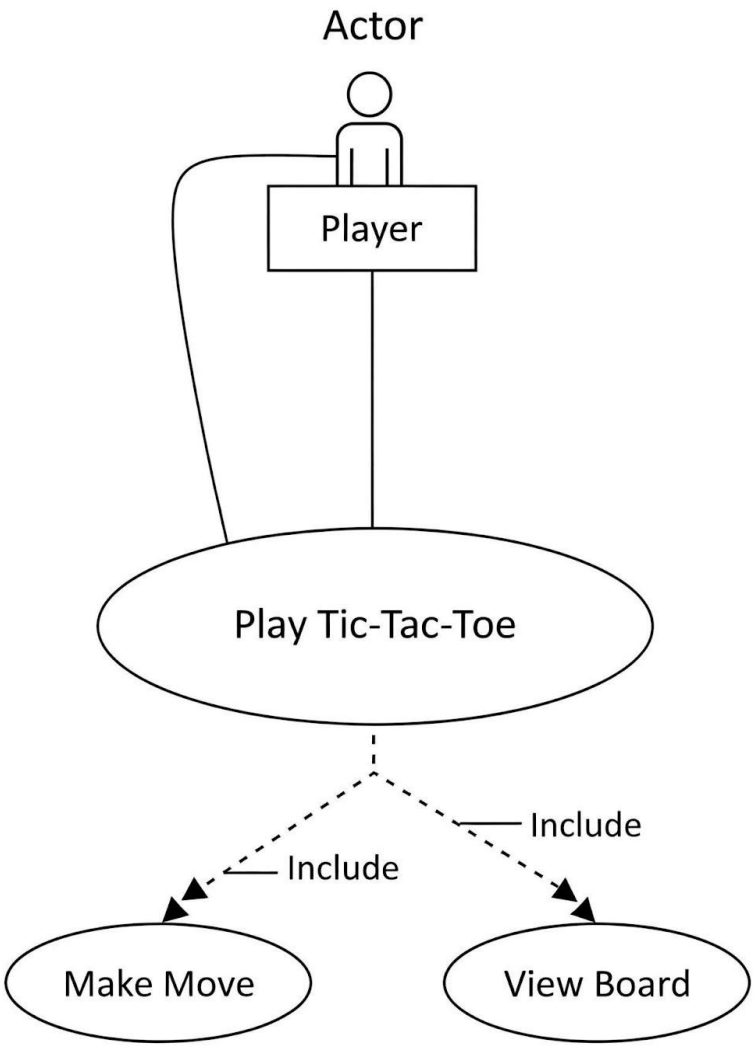
Architecture Description

The system follows a simple linear structure:
Player Input → Validation → Update Board → Check Winner → Repeat

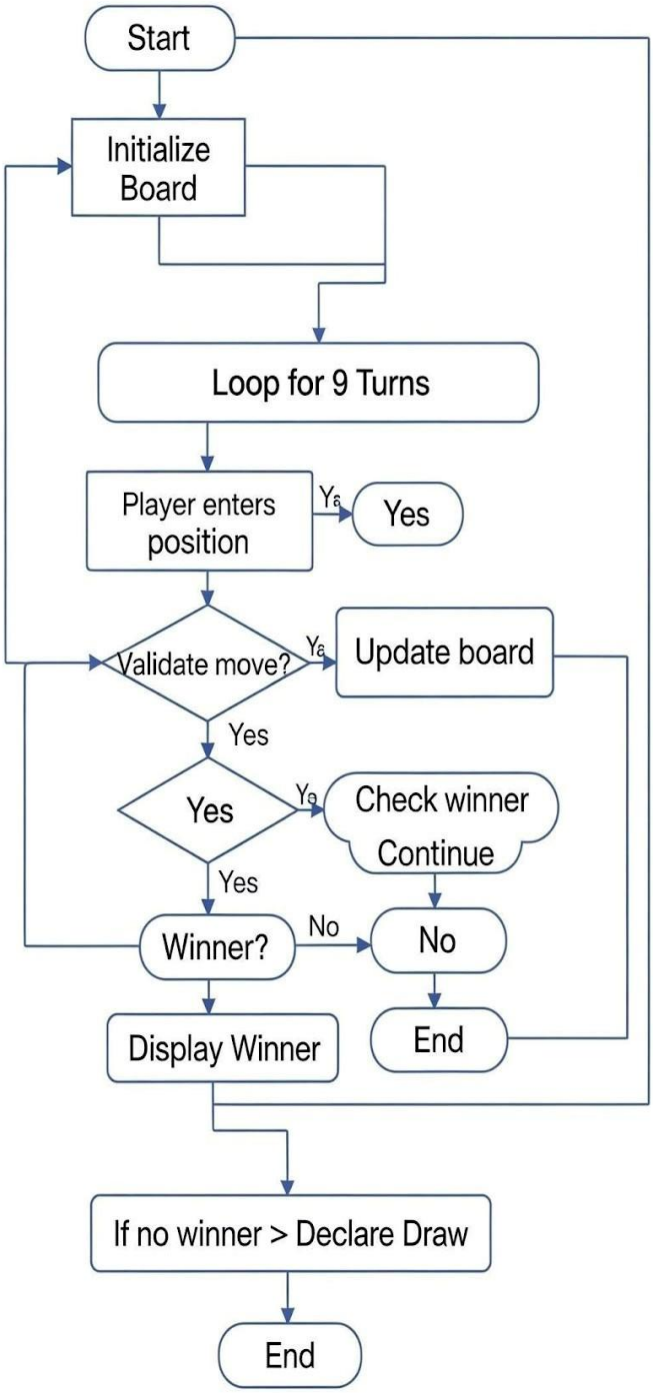


Design Diagram

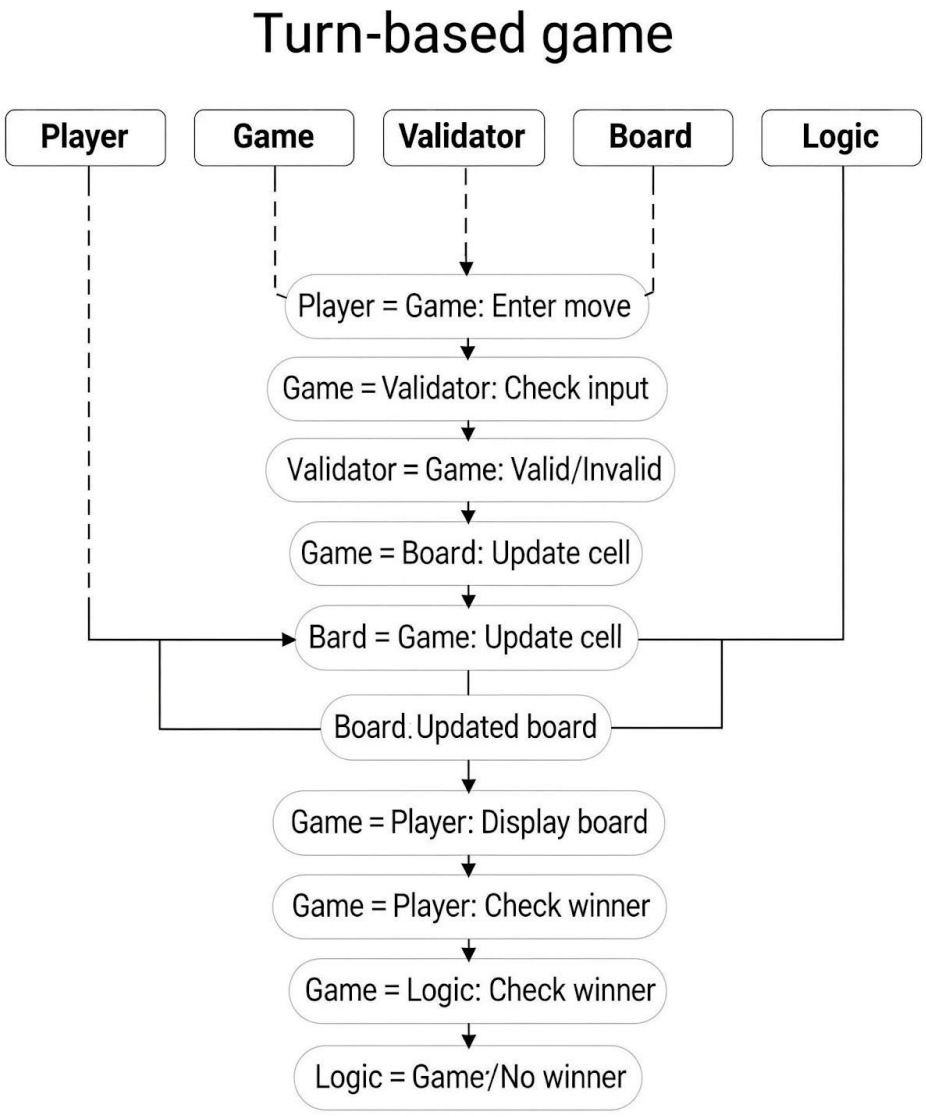
1. Use Case Diagram



2. Work Flow Diagram

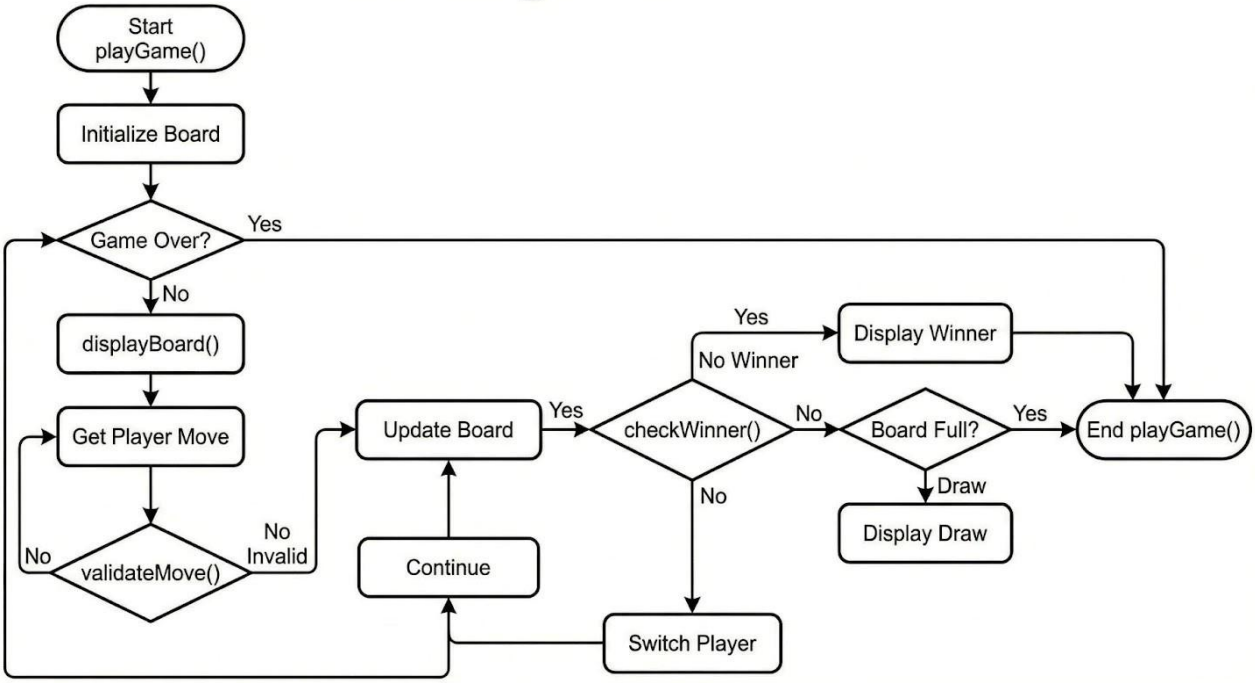


3. Sequence Diagram

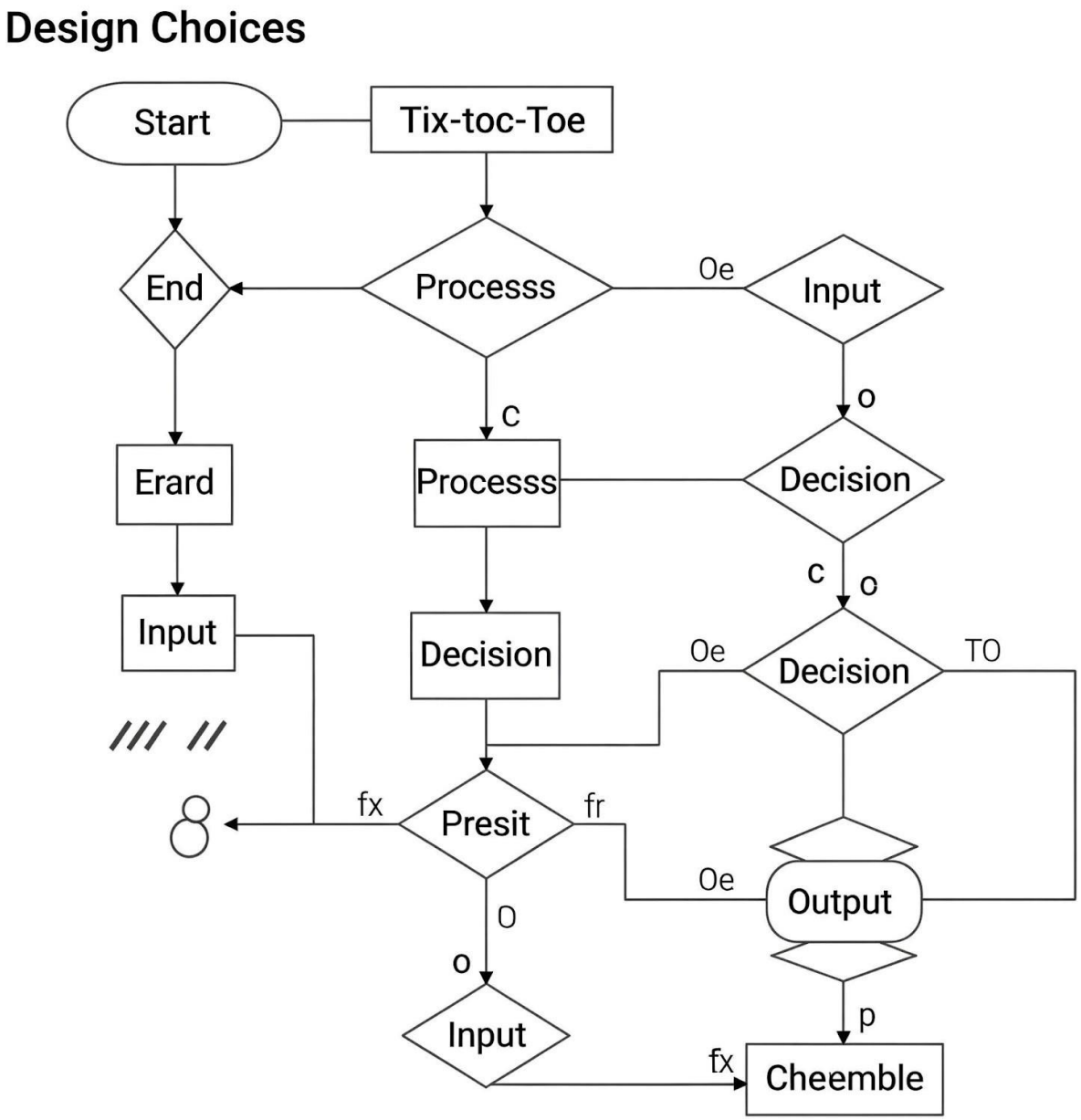


4. Class / Component Diagram

GameLogic Flowchart



Design Decisions & Rationale



Implementation Details

1. Lists: The 3×3 board is a list of 9 elements.

2. Functions: Employ to show the board and find the winners.

3. Conditionals: Ensure that moves are legit, identify winners, and take care of games that end in a draw.

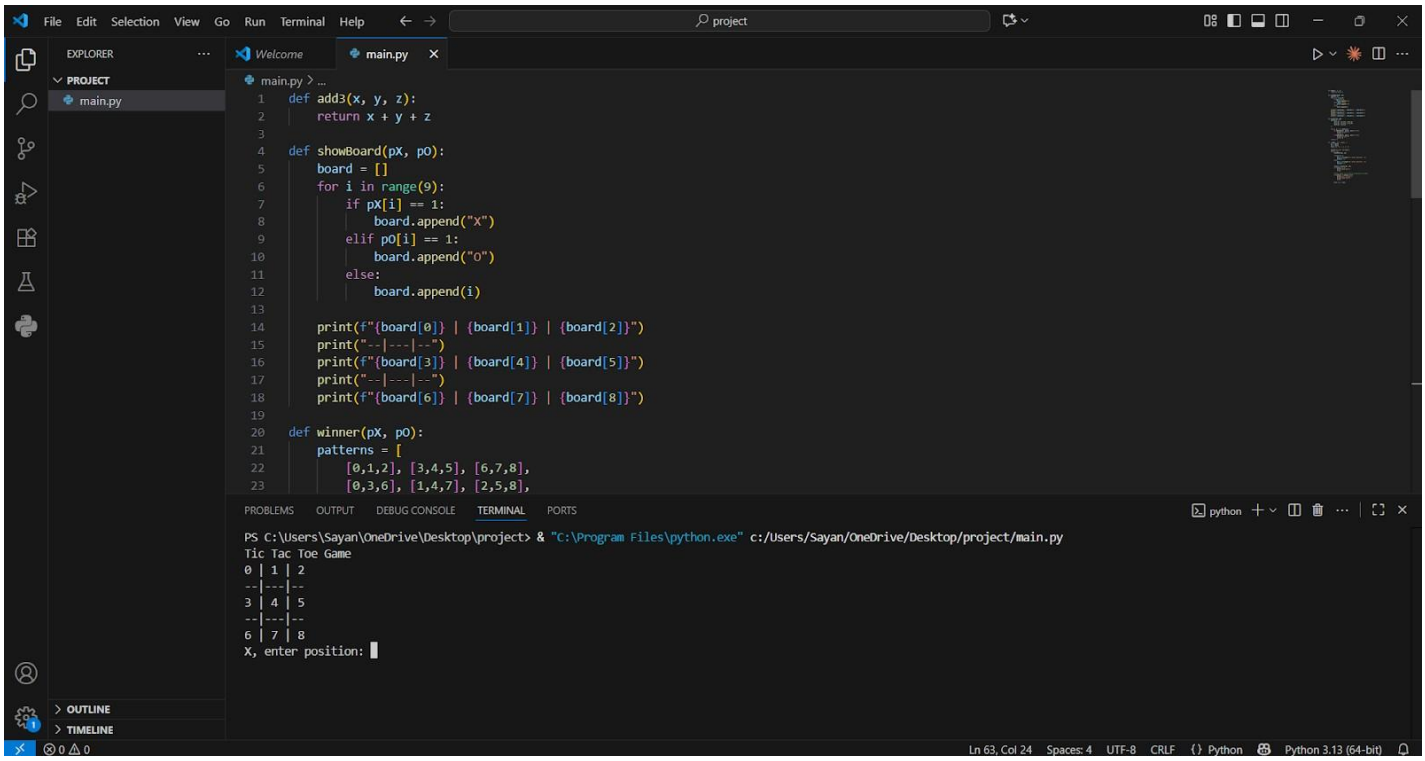
4. Loops: There can be a maximum of 9 turns, thus the game may loop 9 times. Someone may also win before that.

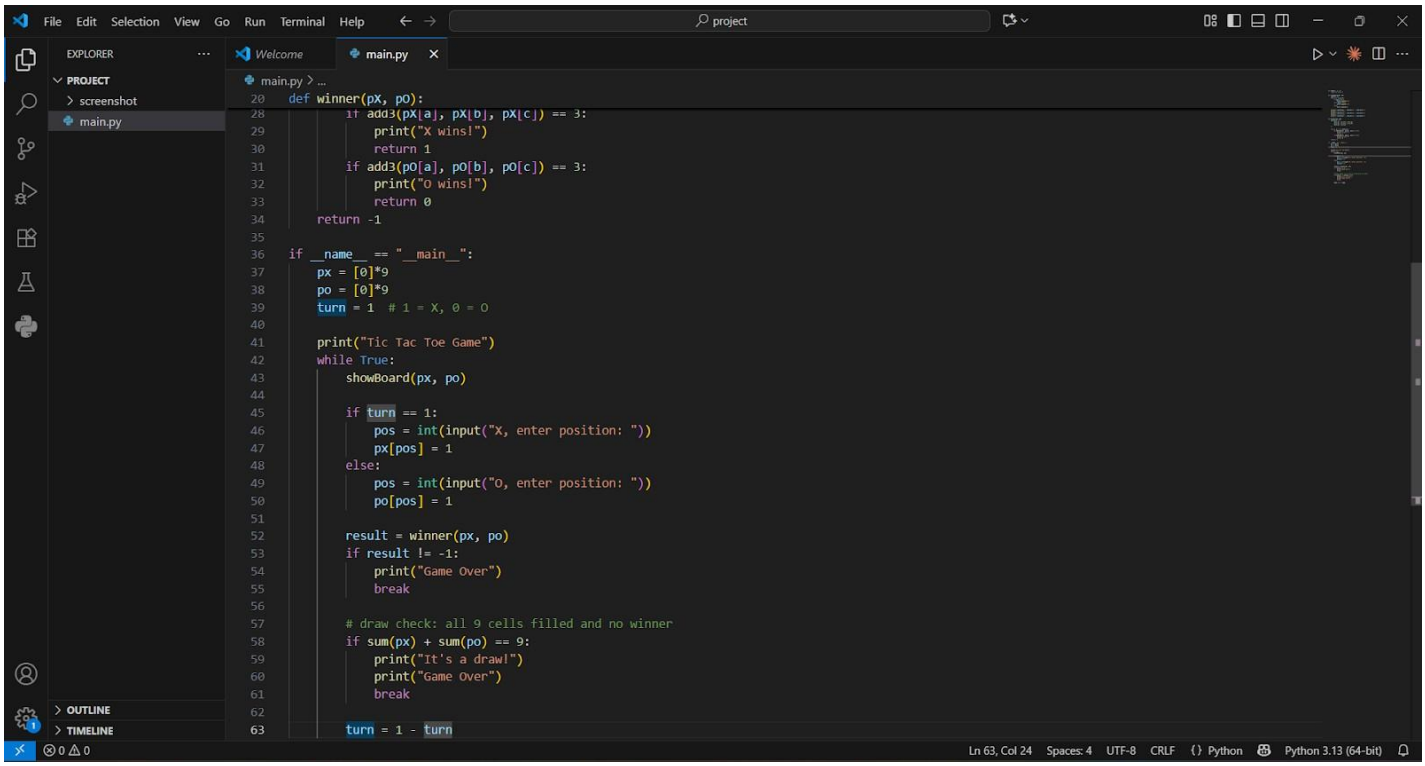
5. After the game is initialized, the board is empty. The players decide which numbered position is going to be their move.

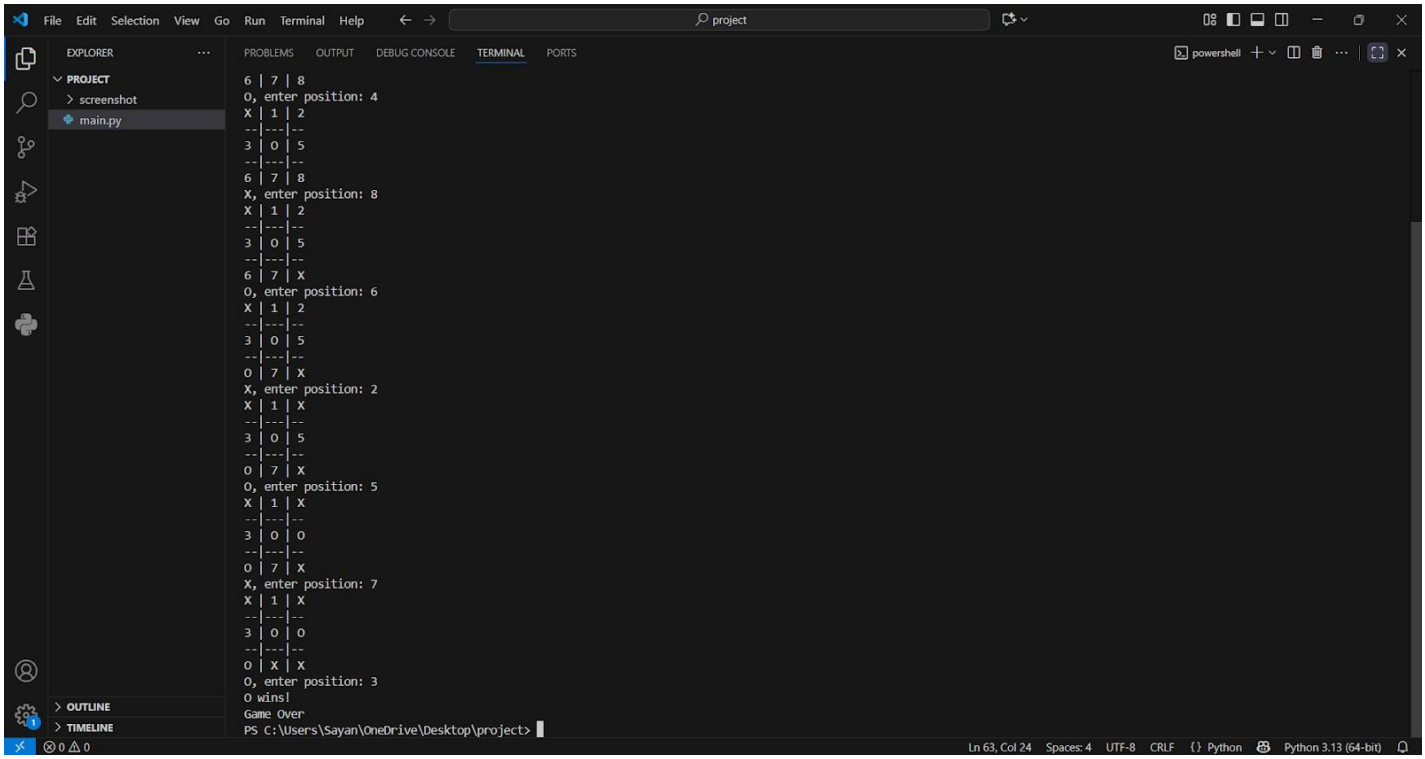
6. The Program assures the move is legal, and hence the board is updated. It checks all possible winning combinations.

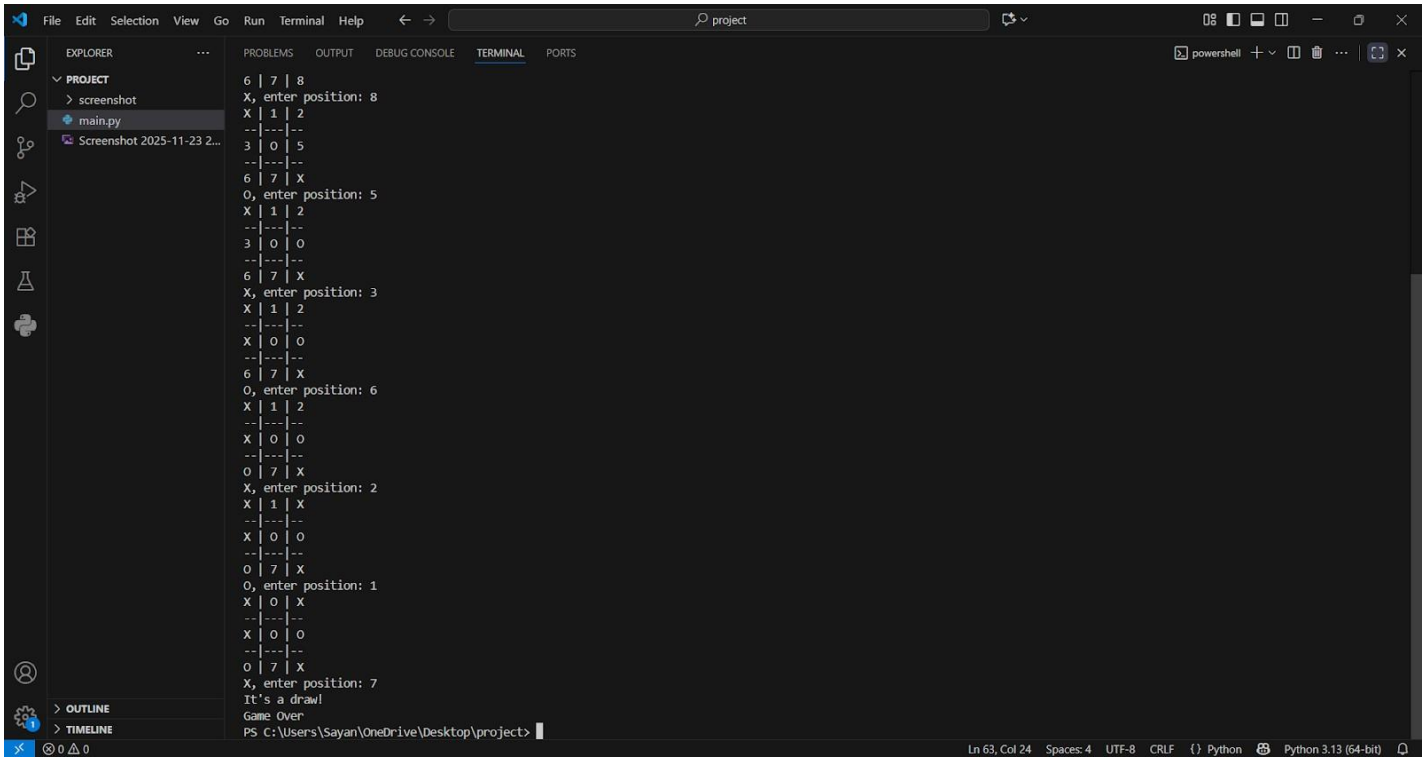
7. If none of the combinations satisfy the winning condition and the board is full, the result is a draw.

Screen Shots and Results




File Edit Selection View Go Run Terminal Help
project
EXPLORER
Welcome
main.py
PROJECT
screenshot
main.py
main.py > ...
def winner(px, po):
if add3(px[a], px[b], px[c]) == 3:
print("X wins!")
return 1
if add3(pO[a], pO[b], pO[c]) == 3:
print("O wins!")
return 0
return -1
if __name__ == "__main__":
px = [0]*9
po = [0]*9
turn = 1 # 1 = X, 0 = O
print("Tic Tac Toe Game")
while True:
showBoard(px, po)
if turn == 1:
pos = int(input("X, enter position: "))
px[pos] = 1
else:
pos = int(input("O, enter position: "))
po[pos] = 1
result = winner(px, po)
if result != -1:
print("Game Over")
break
# draw check: all 9 cells filled and no winner
if sum(px) + sum(po) == 9:
print("It's a draw!")
print("Game Over")
break
turn = 1 - turn
OUTLINE
TIMELINE
Ln 63, Col 24
Spaces: 4
UTF-8
CRLF
Python
Python 3.13 (64-bit)


File Edit Selection View Go Run Terminal Help
project
EXPLORER
PROJECT
screenshot
main.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
6 | 7 | 8
O, enter position: 4
X	1	2
3	O	5
6	7	8
X, enter position: 8
X	1	2
3	O	5
6	7	X
O, enter position: 6
X	1	2
3	O	5
O	7	X
X, enter position: 2
X	1	X
3	O	5
O	7	X
O, enter position: 5
X	1	X
3	O	O
O	7	X
X, enter position: 7
X	1	X
3	O	O
O	X	X
O, enter position: 3
O wins!
Game Over
PS C:\Users\Sayan\OneDrive\Desktop\project>
OUTLINE
TIMELINE
Ln 63, Col 24 Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit)


File Edit Selection View Go Run Terminal Help
project
EXPLORER
PROJECT
screenshot
main.py
Screenshot 2025-11-23 2...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
6 | 7 | 8
X, enter position: 8
X	1	2
3	O	5
6	7	X
O, enter position: 5
X	1	2
3	O	O
6	7	X
X, enter position: 3
X	1	2
X	O	O
6	7	X
O, enter position: 6
X	1	2
X	O	O
O	7	X
X, enter position: 2
X	1	X
X	O	O
O	7	X
O, enter position: 1
X	O	X
X	O	O
O	7	X
X, enter position: 7
It's a draw!
Game Over
PS C:\Users\Sayan\OneDrive\Desktop\project>
OUTLINE
TIMELINE
Ln 63, Col 24 Spaces: 4 UTF-8 CRLF Python Python 3.13 (64-bit)

Testing And Approach

1. Test valid moves
2. Test invalid moves
3. Test winning rows/columns/diagonals
4. Test draw condition
5. Test early termination when winner found

Challenges Faced

1. Checking all winning combinations correctly
2. Ensuring the program rejects illegal moves
3. Making the board display clean and readable

Learnings & Key Takeaways

1. Understanding of functions and modular programming
2. Improved debugging skills
3. Realizing how simple logic builds complete systems

4. Experience with conditional branching and loops

Future Enhancements

1. Add computer AI opponent
2. Add scoring system
3. Add GUI using Tkinter
4. Add multiplayer over network

Reference

1. Vityarthi
2.
3. GitHub Open Source 4.
5. Youtube

Logic to Win

The program checks eight different sequences of cells that might bring victory:

- Three rows
- Three columns
- Two diagonals

- i.e., any three cells which contain the same symbol, X or O, are sufficient to declare that player the winner.

Conclusion

This Tic Tac Toe project is a demonstration of how simple concepts can be used to create a

working game which solves the problem. It is a good pool of exercises

for programming novices and may be developed further by adding

features like AI player, GUI, scoreboard, etc.