

**Tic-Tac-Toe Game**

**Project Title: Tic-Tac-Toe Game**

**Student Name: Shani Kumar**

**Roll No.: 04**

**Branch: CSE AI**

**AKTU ROLL NO.: 202401100300224**

**Date: 11/03/2025**

**Tic-Tac-Toe Game Report**

**Introduction**

Welcome to the Tic-Tac-Toe game! This project is a simple yet intelligent version of the classic game, where you (the player) go up against an AI opponent. The AI is designed to make smart decisions using the Minimax algorithm, making the game both challenging and fun.

**What’s the Goal?**

The main aim of this project is to:

* Create an engaging Tic-Tac-Toe game in Python.
* Develop an AI that plays optimally using the Minimax algorithm.
* Ensure a smooth and fair gaming experience by validating user inputs.
* Make learning about AI decision-making fun and interactive.

**How Does the Game Work?**

**Basic Rules**

1. The game board is a 3x3 grid.
2. You play as 'X', and the AI plays as 'O'.
3. Players take turns placing their symbols in empty spots on the board.
4. The game ends when:
   * A player forms a line of three symbols (horizontally, vertically, or diagonally) and wins.
   * The board is full, resulting in a draw.

**Step-by-Step Breakdown**

1. **Game Starts:** The board is displayed empty.
2. **Your Move:** Enter the row and column numbers where you want to place your 'X'.
3. **AI’s Turn:** The AI calculates the best possible move and places its 'O'.
4. **Check for Winner:** The program checks if there’s a winner after each move.
5. **Repeat:** Steps 2-4 continue until the game ends with a win or a draw.
6. **Game Over:** The board is displayed with the final result.

**How the AI Works (Minimax Algorithm)**

The AI is smart because it uses something called the Minimax algorithm. Here’s how it works:

* The AI looks at all possible future moves and their outcomes.
* It assigns a score to each move: **+1** (AI wins), **-1** (You win), or **0** (Draw).
* The AI picks the move that gives it the best possible outcome while minimizing your chances of winning.
* This means the AI **never makes a mistake**—unless you play perfectly, it won’t lose!

**Code Implementation**

Here is the complete Python code for the Tic-Tac-Toe game with an AI opponent:

import math

# Tic-Tac-Toe board

board = [

[' ', ' ', ' '],

[' ', ' ', ' '],

[' ', ' ', ' ']

]

# Function to print the board

def print\_board():

for row in board:

print('|'.join(row))

print('-' \* 5)

# Check for a winner

def check\_winner():

for row in board:

if row[0] == row[1] == row[2] and row[0] != ' ':

return row[0]

for col in range(3):

if board[0][col] == board[1][col] == board[2][col] and board[0][col] != ' ':

return board[0][col]

if board[0][0] == board[1][1] == board[2][2] and board[0][0] != ' ':

return board[0][0]

if board[0][2] == board[1][1] == board[2][0] and board[0][2] != ' ':

return board[0][2]

return None

# Check if the board is full (draw)

def is\_full():

return all(board[i][j] != ' ' for i in range(3) for j in range(3))

# Minimax algorithm

def minimax(is\_maximizing):

winner = check\_winner()

if winner == 'X': # Human wins

return -1

if winner == 'O': # AI wins

return 1

if is\_full(): # Draw

return 0

if is\_maximizing: # AI's turn (O)

best\_score = -math.inf

for i in range(3):

for j in range(3):

if board[i][j] == ' ':

board[i][j] = 'O'

score = minimax(False)

board[i][j] = ' ' # Undo move

best\_score = max(best\_score, score)

return best\_score

else: # Human's turn (X)

best\_score = math.inf

for i in range(3):

for j in range(3):

if board[i][j] == ' ':

board[i][j] = 'X'

score = minimax(True)

board[i][j] = ' ' # Undo move

best\_score = min(best\_score, score)

return best\_score

# Find the best move for AI (O)

def best\_move():

best\_score = -math.inf

move = (-1, -1)

for i in range(3):

for j in range(3):

if board[i][j] == ' ':

board[i][j] = 'O'

score = minimax(False)

board[i][j] = ' ' # Undo move

if score > best\_score:

best\_score = score

move = (i, j)

return move

# Main game loop

def play():

print("Tic-Tac-Toe: You (X) vs AI (O)")

while True:

print\_board()

# Human move

x, y = map(int, input("Enter your move (row col): ").split())

if board[x][y] != ' ':

print("Invalid move. Try again.")

continue

board[x][y] = 'X'

if check\_winner():

print\_board()

print("You win! 🎉")

break

if is\_full():

print\_board()

print("It's a draw! 🤝")

break

# AI move

i, j = best\_move()

board[i][j] = 'O'

print(f"AI plays: {i}, {j}")

if check\_winner():

print\_board()

print("AI wins! 🤖")

break

if is\_full():

print\_board()

print("It's a draw! 🤝")

break

# Run the game

play()

**Cool Features**

* **AI Opponent:** The AI makes smart, calculated moves.
* **Live Board Updates:** The board updates in real-time as you play.
* **Input Validation:** You can’t place your move in an already occupied spot.
* **Win/Draw Detection:** The program knows when the game is over and announces the result.
* **Fair & Competitive Gameplay:** The AI ensures a fun challenge.

**What You Need to Run the Game**

* **Programming Language:** Python 3.x
* **Required Library:** Only Python’s built-in math module (no extra installations needed)
* **Algorithm Used:** Minimax (a decision-making strategy used in AI)

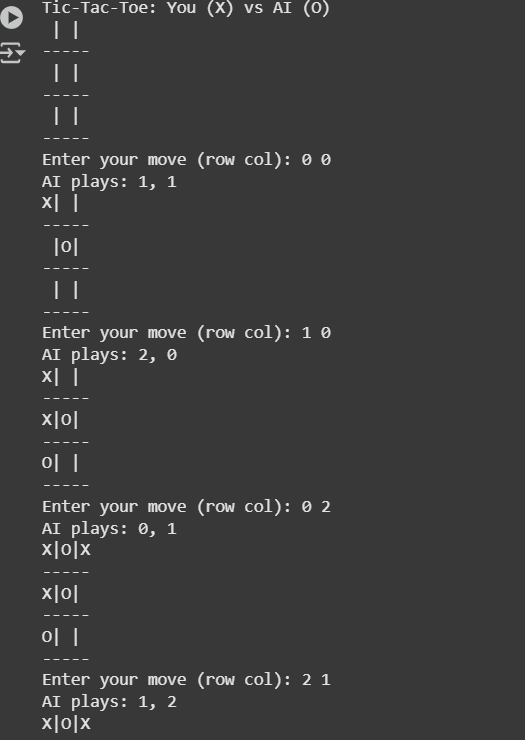
**How to Play**

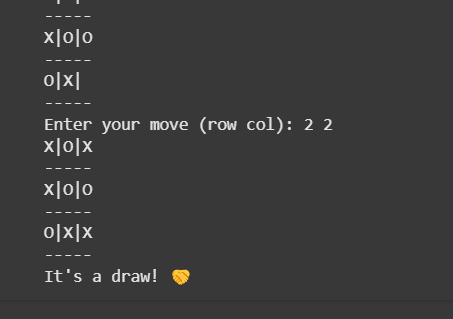
1. Open your terminal or command prompt.
2. Run this command:

python tic\_tac\_toe.py

1. Follow the on-screen instructions to play.

**Sample Game Output**

****

****

**Final Thoughts**

This game brings Tic-Tac-Toe to life with an AI opponent that plays optimally. It’s a great example of how AI decision-making works in real-time. Want to take it a step further? Consider adding a **GUI** for a better experience or adjusting the AI difficulty for different challenges!

**Future Enhancements**

* A **Graphical User Interface (GUI)** for a more interactive experience.
* Different **difficulty levels** to make it easier or harder.
* Game statistics tracking to see how often you win against the AI.

**References**

* Basics of the Minimax Algorithm
* Python Programming Documentation