```
# Tic-Tac-Toe with Minimax (Google Colab Ready)
# Demonstrates the concept of Adversarial Search
# Human = O (MIN player), AI = X (MAX player)

import math

# Initialize board
board = [" " for _ in range(9)]

# Helper: Print board
def print_board(board):
  print("\n")
  for row in [board[i*3:(i+1)*3] for i in range(3)]:
    print(" | ".join(row))
    print("-"*5)

# Check for winner
def check_winner(board):
  win_combos = [(0,1,2),(3,4,5),(6,7,8),  # rows
         (0,3,6),(1,4,7),(2,5,8),  # cols
         (0,4,8),(2,4,6)]         # diagonals
  for a,b,c in win_combos:
    if board[a] == board[b] == board[c] and board[a] != " ":
      return board[a]
  return None

# Check if board is full
def is_full(board):
  return " " not in board

# Minimax algorithm (adversarial search core)
def minimax(board, depth, is_maximizing):
  # Terminal condition: check if game is won/lost/drawn
  winner = check_winner(board)
  if winner == "X":
    return 1    # Utility: MAX (AI) wins
  elif winner == "O":
    return -1   # Utility: MIN (Human) wins
    elif is_full(board):
        return 0    # Utility: Draw

    # If it is MAX's turn (AI)
    if is_maximizing:
        best_score = -math.inf
        for i in range(9):
            if board[i] == " ":
                board[i] = "X"  # Try move for MAX
                score = minimax(board, depth+1, False)  # Recurse: opponent's turn
                board[i] = " "  # Undo move
                best_score = max(score, best_score)    # MAX chooses highest value
        return best_score

    # If it is MIN's turn (Human)
    else:
        best_score = math.inf
        for i in range(9):
            if board[i] == " ":
                board[i] = "O"  # Try move for MIN
                score = minimax(board, depth+1, True)   # Recurse: opponent's turn
                board[i] = " "  # Undo move
                best_score = min(score, best_score)    # MIN chooses lowest value
        return best_score

# AI move (X)
def ai_move(board):
    best_score = -math.inf
    move = None
    # Explore all possible moves
    for i in range(9):
        if board[i] == " ":
            board[i] = "X"  # Simulate AI move
            score = minimax(board, 0, False)  # Evaluate using minimax
            board[i] = " "  # Undo move
            if score > best_score:
                best_score = score
                move = i
    # Choose the move that maximizes AI's outcome
    board[move] = "X"

# Human move (O)
def human_move(board):
    move = int(input("Enter your move (1-9): ")) - 1
    if board[move] == " ":
        board[move] = "O"
    else:
        print("Invalid move, try again.")
        human_move(board)

# Game loop
print("Welcome to Tic Tac Toe! You are O (MIN), AI is X (MAX).")
print_board(board)

while True:
    # Human turn (MIN)
    human_move(board)
    print_board(board)
    if check_winner(board) == "O":
        print("You win! (Utility = -1)")
        break
    if is_full(board):
        print("It's a draw! (Utility = 0)")
        break

    # AI turn (MAX)
    ai_move(board)
    print("AI plays:")
    print_board(board)
    if check_winner(board) == "X":
        print("AI wins! (Utility = +1)")
        break
    if is_full(board):
        print("It's a draw! (Utility = 0)")
        break
```