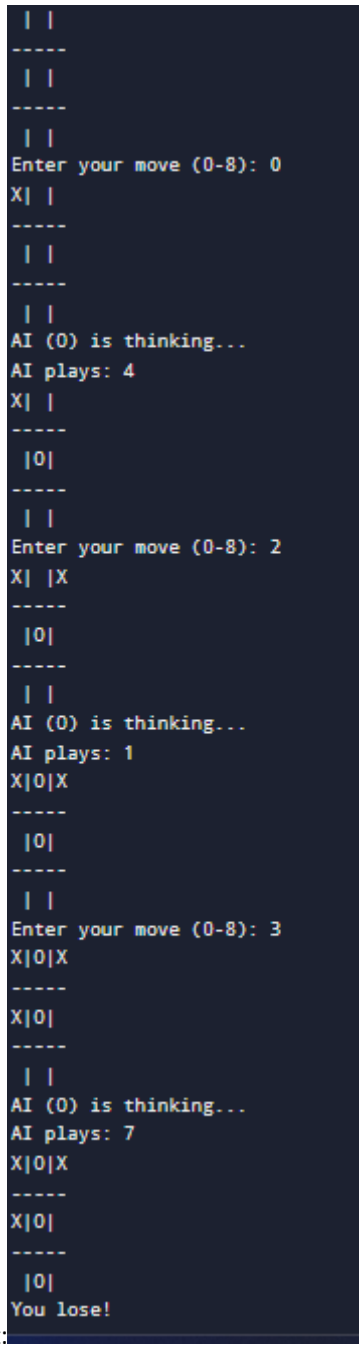
Tic tac toe code:

```
class TicTacToe:
    def __init__(self):
        self.board = [' '] * 9
        self.current_player = 'X'

    def print_board(self):
        for i in range(0, 9, 3):
            print("|".join(self.board[i:i + 3]))
            if i < 6:
                print("-----")

    def is_winner(self, player):
        # Check rows
        for i in range(0, 9, 3):
            if all(self.board[j] == player for j in range(i, i + 3)):
                return True

        # Check columns
        for i in range(3):
            if all(self.board[j] == player for j in range(i, 9, 3)):
                return True

        # Check diagonals
        if all(self.board[i] == player for i in [0, 4, 8]):
            return True
        if all(self.board[i] == player for i in [2, 4, 6]):
            return True

        return False

    def is_full(self):
        return ' ' not in self.board

    def is_game_over(self):
        return self.is_winner('X') or self.is_winner('O') or self.is_full()

    def get_available_moves(self):
        return [i for i, v in enumerate(self.board) if v == ' ']

    def make_move(self, move):
        self.board[move] = self.current_player
        self.current_player = 'O' if self.current_player == 'X' else 'X'

    def undo_move(self, move):
        self.board[move] = ' '
        self.current_player = 'O' if self.current_player == 'X' else 'X'


def minimax(board, maximizing_player):
    if board.is_game_over():
        if board.is_winner('X'):
            return -1
        elif board.is_winner('O'):
            return 1
        else:
            return 0
```

```
    if maximizing_player:
        max_eval = float('-inf')
        for move in board.get_available_moves():
            board.make_move(move)
            eval = minimax(board, False)
            board.undo_move(move)
            max_eval = max(max_eval, eval)
        return max_eval
    else:
        min_eval = float('inf')
        for move in board.get_available_moves():
            board.make_move(move)
            eval = minimax(board, True)
            board.undo_move(move)
            min_eval = min(min_eval, eval)
        return min_eval


def get_best_move(board):
    best_move = None
    best_eval = float('-inf')

    for move in board.get_available_moves():
        board.make_move(move)
        eval = minimax(board, False)
        board.undo_move(move)

        if eval > best_eval:
            best_eval = eval
            best_move = move

    return best_move


# ========================
# Play the Game
# ========================

game = TicTacToe()

while not game.is_game_over():
    game.print_board()

    if game.current_player == 'X':
        try:
            move = int(input("Enter your move (0-8): "))
        except ValueError:
            print("Invalid input! Enter a number 0-8.")
            continue

        if move not in game.get_available_moves():
            print("Invalid move! Try again.")
            continue

        game.make_move(move)
```

```
    else:
        print("AI (O) is thinking...")
        move = get_best_move(game)
        print(f"AI plays: {move}")
        game.make_move(move)

game.print_board()

if game.is_winner('X'):
    print("You win!")
elif game.is_winner('O'):
    print("You lose!")
else:
    print("It's a draw!")
```

Output:


| |
---
| |
---
| |
Enter your move (0-8): 0
X| |
---
| |
---
| |
AI (O) is thinking...
AI plays: 4
X| |
---
|O|
---
| |
Enter your move (0-8): 2
X| |X
---
|O|
---
| |
AI (O) is thinking...
AI plays: 1
X|O|X
---
|O|
---
| |
Enter your move (0-8): 3
X|O|X
---
X|O|
---
| |
AI (O) is thinking...
AI plays: 7
X|O|X
---
X|O|
---
|O|
You lose!