```
def is_game_over(board):
    return is_winner(board, 'X') or is_winner(board, 'O') or is_full(board)

def get_best_move(board):
    best_move = None
    best_eval = float('-inf')
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = 'O'
                eval = minimax(board, 0, False)
                board[i][j] = ' '
                if eval > best_eval:
                    best_eval = eval
                    best_move = (i, j)
    return best_move

def print_board(board):
    print("\nCurrent Board:")
    for row in board:
        print(' | '.join(row))
        print('-' * 9)
    print()

def is_winner(board, player):
    for i in range(3):
        if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j in range(3)):
            return True
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3)):
        return True
    return False

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

def minimax(board, depth, maximizing_player):
    if is_winner(board, 'X'):
        return -1
    elif is_winner(board, 'O'):
        return 1
    elif is_full(board):
        return 0

    if maximizing_player:
        max_eval = float('-inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'O'
                    eval = minimax(board, depth + 1, False)
                    board[i][j] = ' '
                    max_eval = max(max_eval, eval)
        return max_eval
    else:
        min_eval = float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'X'
                    eval = minimax(board, depth + 1, True)
                    board[i][j] = ' '
                    min_eval = min(min_eval, eval)
        return min_eval

def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    current_player = 'X'

    while not is_game_over(board):
        print_board(board)

        if current_player == 'X':
            print("Your turn:")
            try:
                row, col = map(int, input('Enter your move (row and column: 0 1 2): ').split())
                if 0 <= row < 3 and 0 <= col < 3 and board[row][col] == ' ':
                    board[row][col] = 'X'
                    current_player = 'O'
                else:
                    print('Invalid move, try again.\n')
            except ValueError:
                [illegible]
                print('Invalid input. Enter two numbers between 0 and 2.\n')
        else:
            best_move = get_best_move(board)
            if best_move:
                board[best_move[0]][best_move[1]] = 'O'
            print("\nComputer has played its move.\n")
            current_player = 'X'

    print_board(board)
    if is_winner(board, 'X'):
        print('Player Wins!\n')
    elif is_winner(board, 'O'):
        print('AI Wins!\n')
    else:
        print('It is a draw!\n')

if __name__ == '__main__':
    play_game()
```

```
Current Board:
  |   |  
---------
  |   |  
---------
  |   |  
---------

Your turn:
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-4-11e4451cead0> in <cell line: 0>()
     97 
     98 if __name__ == '__main__':
---> 99     play_game()

2 frames
/usr/local/lib/python3.11/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
    893             except KeyboardInterrupt:
    894                 # re-raise KeyboardInterrupt, to truncate traceback
--> 895                 raise KeyboardInterrupt("Interrupted by user") from None
    896             except Exception as e:
    897                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```