```
PLAYER_X, PLAYER_O, EMPTY = 1, -1, 0


def evaluate(b):
  for i in range(3):
      if b[i][0] == b[i][1] == b[i][2] != EMPTY: return b[i][0]
      if b[0][i] == b[1][i] == b[2][i] != EMPTY: return b[0][i]
  if b[0][0] == b[1][1] == b[2][2] != EMPTY: return b[0][0]
  if b[0][2] == b[1][1] == b[2][0] != EMPTY: return b[0][2]
  return 0


def isMovesLeft(b):
  return any(EMPTY in row for row in b)


def minimax(b, isMax):
  score = evaluate(b)
  if score != 0 or not isMovesLeft(b): return score
  best = -float('inf') if isMax else float('inf')
  for i in range(3):
      for j in range(3):
          if b[i][j] == EMPTY:
              b[i][j] = PLAYER_X if isMax else PLAYER_O
              val = minimax(b, not isMax)
              b[i][j] = EMPTY
              best = max(best, val) if isMax else min(best, val)
  return best


def findBestMove(b):
  bestVal, move = -float('inf'), (-1, -1)
  for i in range(3):
      for j in range(3):
          if b[i][j] == EMPTY:
              b[i][j] = PLAYER_X
              val = minimax(b, False)
              b[i][j] = EMPTY
              if val > bestVal:
                  bestVal, move = val, (i, j)
  return move


def printBoard(b):
  for row in b:
      print(" ".join("X" if x == PLAYER_X else "O" if x == PLAYER_O else "." for x in row))


board = [
  [PLAYER_X, PLAYER_O, PLAYER_X],
  [PLAYER_O, PLAYER_X, EMPTY],
  [EMPTY, PLAYER_O, PLAYER_X]
]


print("Current Board:")
printBoard(board)


move = findBestMove(board)
print(f"Best Move: {move}")
board[move[0]][move[1]] = PLAYER_X


print("\nBoard after best move:")
printBoard(board)
```