Programming II

Exercise Performance: Connect Four

Markus Mayer
markus.mayer@th-deg.de

Overview

- All rules from the slide set `00_exercise_performance_general.pdf` apply.
- The main file is called `connect_four.py`
- Deadline is 2023-05-01 (Monday), 23.59 o'clock.
- There are 5 points to achieve:
  - One point for sticking to the general rules.
  - 3 Points for the console game against another human player.
  - One point for the implementation of the AI player.

The task

- Implement Connect Four such that it can be played on the Python console.
- The board size of the game is set by quasi-constants in the code and not fixed due to magical numbers.
- A player wins if they have 4 checkers in a row, column, or diagonal. If this is the case, the game ends and it is asked if it should be restarted or quit.
- After start, the game asks if the player wants to play against another human player, or against an AI. in case of an AI player, the human always has the first move.
- The (human) moves are made by asking for a column number (counting starting from 0 or 1) or if the player wants to restart or quit the game (which is then of course also possible).

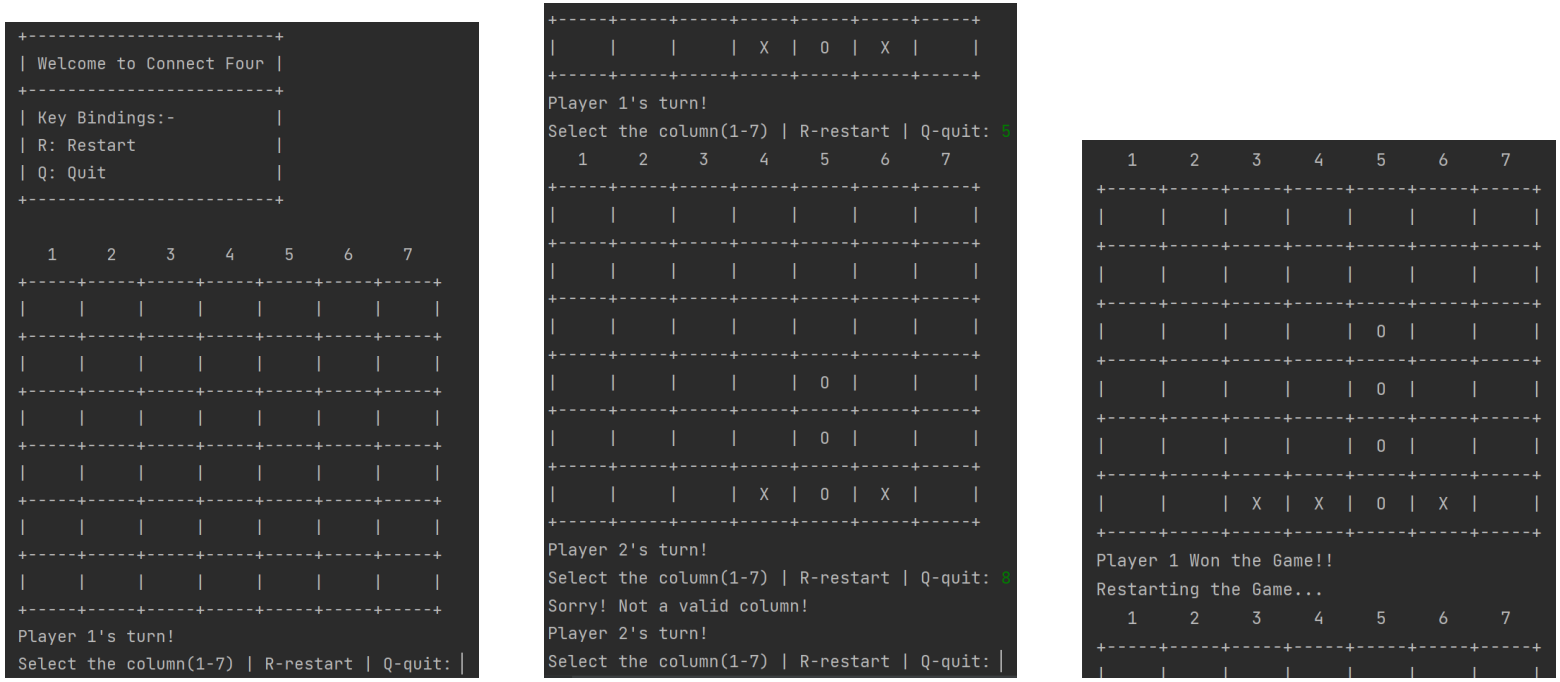
The task (2)

- If a non-valid move is made by a player, it is asked again until a valid move is given as user input (or the restart or quit options are selected).
- After each move, the board is printed to the console again.

The AI player

- Connect Four can be played (for small board sizes) optimally by an algorithm exploring all possible moves, the so called "Minimax" algorithm.
- You'll find various variants (coded in Python) for this algorithm on the internet. Possible search keywords are *"TicTacToe Python Optimal"* or *"Connect Four Minimax"* etc.
- You are allowed (and should) use a implementation you found (and adapt it), but cite the source in your code.
- We discuss Minimax shortly in the lecture (without slides).

The console play: Example



(a) (b) (c)

Figure: Connect four example console version (programmed by Dhruv Patel). Be aware: This version has no AI player, and therefore the welcome screen / restart options differ from what is required in this task (a) Welcome screen (b) Wrong move (c) End and automatic restart of the game (in your version you should ask for restart or quit).

Thoughts before you start coding

- You can implement the game with OOP or pprocedural (e.g. using functions only). No matter what programming paradigm you use, you have to decide:
  - How do you represent the board?
  - What are necessary functions/methods that work on the board?
  - What functions/methods do you need to define the win (or draw) state?
  - What functions/methods do you need to make the Minimax algorithm work?
- Your console printout does not have to reproduce the example given before - it may be simpler. The only condition is that the board is clearly visible.