Homework #3

Due: Nov. 2, 2022

The third homework assignment will be to implement an AI agent to play connect four: https://en.wikipedia.org/wiki/Connect_Four. It doesn't matter to me if you have the agent play player one or player two, but let's arbitrarily choose that the person will be player 1 (the red player as per my convention) and that player 2 will be the agent (blue player). However, if you want player one to be a computer agent – or choose who goes first randomly, that is fine with me—especially if that is done for testing – see below.

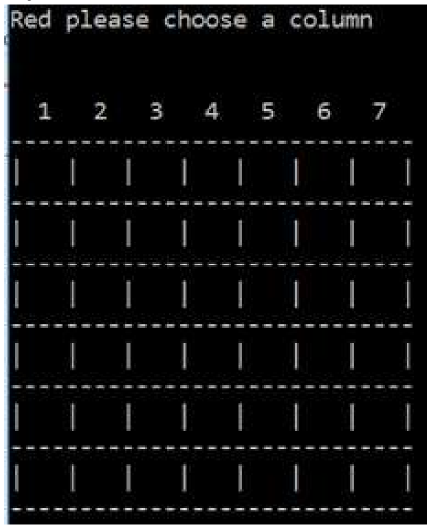
The grade breakdown will be:

30% -- creating a basic agent that plays the game as per the rules with some basic, heuristic logic

20% -- documentation about the logic of your solution

50% -- the creativity of your solution—defined below

The interface need not be anything fancy. This is what I had imagined and previously implemented myself:



```
Red please choose a column


  1   2   3   4   5   6   7
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
```

The human player (red) can then choose a column 1-7. To make things easier, I'm including a working version of the game where the agent plays randomly. It is written assuming a white background and with the termcolor package (which you likely will have to install). No problem if you take it out!

To get the full 20% (20 points) for documentation, there should be both comments in your Python solution and in a separate document (up to one page) of general explanation of your code.

To get the 50% (50 points) for creativity you will need to justify that your code is in fact creative. My own agent for this game wasn't clever enough to beat my wife. She kept on double trapping it. What could you do to make your agent plan ahead for that type of move? There is no one correct answer here, and minimax/alpha-beta pruning is one of the possibilities (especially if the search tree looks ahead 2-3 moves). You don't *have* to go that route, but I'll be grading the agent based on how often the agent you write will beat the random agent I included. Some baseline evaluation techniques you should consider:

1. What heuristics do you use to evaluate a game state? What "helper functions" do you want to create? You might want to consider subtask planning functions like, "Should_win", "Should_Block" and others.
2. How often does your game beat an agent that plays randomly? (It should be near 100%)
3. **Why** does the agent you wrote sometimes lose?
4. Truly for extra credit: what happens when you have the agent play itself.
5. Can you design agents of varying skill levels (e.g. agent1 and agent2 both nearly always beat the random agent, but agent1 usually beats agent2)?

Submission will be via Git (like you did last time).