

WASHINGTON STATE UNIVERSITY VANCOUVER

PROGRAM DESIGN AND DEVELOPMENT - CS 121

Assignment 10 (Final Prep)

Professor:
Ben MCCAMISH

November 17, 2020

Overall Assignment

Your code should all be contained in a single file titled `assignment10.py`. Make sure that they all work on the Ubuntu OS in the labs. Your program must be written for Python 3.6+.

Warning: This assignment will be used for the final project. You will want to make sure that it works before moving onto the final assignment.

This week you will implement the Player class and edit your existing Connect 4 class. There are two areas to be affected by adding an “Artificially Intelligent” player to your Connect 4 simulator from Assignment 9. First, you will need to implement a class `Player` which can generate moves, given a connect4 board. Next, you will write a method named `playGameWith(aiPlayer)` for your `connect4` board class. This method will be very similar to `hostGame()` except it will regard the `aiPlayer` parameter as an object which will compute its own next move, rather than asking for input from a user.

1 Getting started

Create a new file called `assignment10.py` and copy your class `Connect4` you made in assignment 9 into this file. Then, create a class called `Player` with a constructor method and a representation method. Your representation method is up to you. A suggestion might be to print out all the current stats of the player (ply, tie breaker, etc).

2 Constructing the Player Class

Some things it will need to know:

- Its checker type... X or O
- How far to look ahead when computing the next move (ply)
- Possibly, how to resolve ties between seemingly equally good moves

What must class player know how to do? There is really only one method it must have (other than `__init__`) and that is `nextMove(self, board)`. When invoked, this method examines the connect4 object board and returns an integer indicating its desired move (column number). This method will be used by the connect4 object's `playGameWith()` method to obtain the AI player's move, rather than asking for an interactive user's move.

For this assignment, you will primarily be on your own. Here are some tips (in addition to the slides) to get you started.

- It is likely that you may want to implement some of the suggested methods for class player from the lecture slides. However, how you implement class Player is up to you, but your code must be your own. However, you MUST implement the `nextMove(self, board)` method.
- Your class player should have the capability of looking ahead at least 3 moves (ply = 3). More is OK, but you will find that your program slows down significantly for ply > 3.
- You may find it interesting (not required) to implement a capability in class connect4 to play a game between two different player objects. Then have multiple instances of your players fight it out, possibly with different values for ply or other parameters.
- Tiebreaker should be either 'Left', 'Right', or 'Random'.

3 Modifying the Connect4 Class

How does the connect4 class change? Well, one new method is needed called `playGameWith(self, aiPlayer)`. This method is similar to `hostGame()` except for the following:

- The first player to move is 'X' and X's moves are solicited from the user, just as in `hostGame()`
- The second player to move is 'O' and O's moves are obtained and made by obtaining a move from the aiPlayer, something like `oMove = aiPlayer.nextMove(self) # get the move self.addMove(oMove, 'O') # make the move`

Autolab Notes:

- Submit your python code as you go to determine whether you answered the sections correctly.
- The tests for the `playGameWith` will be performed by the TA.
- Only submit a single .py file.
- 60 points for autolab tests (however, they are not complete)
- 40 points for `\playGameWith()`
- 'x' and 'o' must be lowercase from now on.