

Understanding and modifying code for lists

In this session you will be working on an Alice program that plays Connect Four. First your peer leader will demonstrate the program so you can see what it does. Next the peer leader will give you a version of the program that is missing the code for two methods. The first step is to look at the code provided to figure out what it is doing. Look at the world variables. Figure out what they are keeping track of (including the lists). Figure out how the board is laid out and if it makes sense in how the circles are numbered. Then, look at my first method. See what it calls and untangle the code. This will take a little while so be patient and ask other group(s) for help.

Once you are comfortable with the code, design a way to fill in the move method. It is designed to make the chip drop when it is clicked. A few things to consider:

- Don't drop chips below where ones already have been placed
- Don't drop onto a full column. In that case they lose their turn.

Once you think you have a working idea, try it in the code. (Note for testing at this point you can make wonColumn return false so the game never ends.) If you have problems getting Alice to insert the statement you want, ask for some help. Test what you have and make sure it works for all cases.

The next step is to fill in the code for wonColumn. It should return true if all the items on the supplied list (which are groups of circles) are owned by one player. (Note it does columns, rows & diagonals so the name column isn't the best – group would be better.) If not, it returns false. The won method will make sure it checks all the columns needed. If it works then you can have fun playing the game.

If you finish early, you could consider what changes you need to do Connect Five and if lists make it easier.