huji moodle 2018/19

Search this forum

Amir Krause • 1 January 2019, 21:13 • Edited by the author on 15 January 2019, 12:43

Important questions and clarifications - Read this first, do not reply here

The subjects are ordered based on their importance, but you can see which bulletins were last updated by the date:

- **(15/01/19) - "I got an exception in the presubmit. What happened??"** - The presubmit, besides checking for the correct file and folder structures and making sure you have a readme and authors file, also attempts two additional tasks: The first is to initialize a new Game object and checking (via get_player_at) that all the cells contain the correct value (None), the second is to create a new Game object and two ai players (with two distinct AI objects) then making them play one against the other. In this section there are two common problems that you might encounter:
    - The ai players try to put a disc at a column that is already full (you can see that, for example, with sequences like "1-3, 2-3, 1-3, 2-3, 1- 3, 2-3, 1-3, 2-3, 1-3, 2-3, 1-3, 2-3, 1-3", where the first number if the player and the second is the column - we tried to place 7 discs at column 3.
    - You AI object, either in its constructor or find_legal_move, used some of your own created methods on the game object given to the AI in the constructor "our game". You can only use get_player_at, get_current_player and get_winner on that object, nothing more.

- (07/01/19) - "Can I (add methods / add functions / add classes / add variables or constants / add exceptions / place new files / use images / buy ice-cream) ?" - **yes, you can design the program however you want**. The only restrictions are:

-
    - You must not change the expected behavior of the API methods already detailed in the exercise description, including the exceptions they raise. Of course, these methods could use other methods you create.
    - game.py and ai.py cannot depend (by any means) on any graphics modules (as is clearly written in the exercise).
    - You cannot use networking, threading, signaling, bytecode manipulation or code in other programming languages (all of which were not taught).
    - You cannot use anything that isn't already installed on the cs labs computers (for example, pygame).
    - You have to keep the folder structure as-is (you cannot add additional folders). You are allowed to create additional files (given that they are not too large, the moodle submission has a cap of

several MBs), as long as these files are included in your zip file inside the ex12 folder. You may only create files in advance, you cannot create new files via a python code during your program's run (but you can alter existing files which you submitted). If you are using external files, you need to know how to use the __file__ variable to get the path to the file, even if it is in the same folder (if you don't know what that is, we recommend not using files). Note that working with files is only an option and *is not needed to solve the exercise, including the AI part, including sophisticated AI algorithms*.

- (13/01/19) - What can you do with the Game object inside AI, and what can you not do?
  - You can only use its exercise defined public API methods *get_player_at, get_winner* and *get_current_player*. You cannot use *make_move* on it (you cannot change the Game object's state from within the AI). For the purpose of finding a simple legal move, this is enough.
  - (This section is probably relevant only to the students who want to devise sophisticated AI algorithms) - But what if you do want to change the state, access other types or data, or use your own Game object (possibly with additional features) when calculating the AI's move? As explained in the exercise description, in this case, you could "clone" the Game object stored in the AI object. You can create another Game object of your own implementation (without changing the Game object stored in the AI object), then use the Game object stored in the AI object to "set" the data in your own Game object to reflect the same game state (after all, the game is a 6x7 board and you can get the type of disc at each location). Then, you can use the created Game object, with your implementation and methods, to decide the AI's next move. This will work in the following way: at each invocation of find_legal_move, you will create a new Game object *of your design*. Then, you will take the Game object stored in the AI object (which you cannot assume anything about, other than its representation of a valid game state), and use the methods you have access to (mainly get_player_at), to change the state of your Game object, such that it reflects the same "board" as the Game object in the AI object. Then, you can use your own Game object, in order to devise your solution. The Game object in the AI object will remain unchanged, and you do not need (and must not!) change it at all.
  - Remember: In the tournament, we will first call *find_legal_move* then assume that it places some kind of value (a result move) inside your AI object, such that calling *get_last_found_move* will give us that value. There is no need to actually find a move, again, inside *get_last_found_move* (note the method's name: *get*, it just fetches the value already found and stored by *find_legal_move*).

- Note that the AI object is responsible for one AI player (thus the player parameter is received in the constructor). A slight mistake has occurred in the description, stating that find_legal_move also receives a player parameter - this is wrong, find_legal_move should only receive the default timeout parameter (which could also be removed if you do not implement the bonus). The player for which the AI needs to find a move is inferred from the player that was received by the constructor. Also, the AI cannot find a move if the board is full or a winner already exists in the game.

- (13/01/19) - Remember you need to mark the winning streak when the game ends! If placing a disc resulted in two kinds of winning streaks, just mark one of them (or all of them - you choose).

- (13/01/19) - Some of you encountered a problem with displaying images. It might be solved by adding the line in bold here:

  ```
  l = tk.Label(root, image=photo_file)
  ```

  `l.photo = photo_file`

  (where photo_file is the location of the file).

- You need to make your import commands relative, as the exercise specifies ("from . import Game" for .py files within the ex12 folder or "import ex12.game" for four_in_a_row.py). However, this results in inability to run any code via "if __name__ == "__main__"" in any .py file inside the ex12 folder. If you try it, for example, from within game.py, you will get an error (The error will look something like this: *"ModuleNotFoundError: No module named '__main__.game';"*). The solution is to either not run any __main__ block code from any file within ex12.py (your only main, in the submission, should be in four_in_a_row.py which is outside), or - if you do wish to have main blocks inside ex12 for testing (remember to remove them upon submission) - just alter the imports to not relative ones (back to "import game" or "from game import Game"), but remember to change back prior to submission.

- The game board will always (including the contest) be a 7-columns 6-rows board. This is a global assumption and every object can assume that.

- You may assume the Game object given to the AI constructor is legal (chips are stacked correctly), but you cannot assume there isn't a winner already in the initial state.

- (09/01/19) - When we ask to raise an exception, the type of the exceptions raised does not matter and is up to you. The only thing we require is that the exceptions' string message be what we expect.

- During a game, the program should not crash due to exceptions (even though the methods possibly raise exceptions on some cases). Think about how to achieve that goal (ignore the exceptions and not doing the illegal action; limiting the GUI such that it is just impossible to make an exception, etc). Remember you have complete control.

Subscription

You do not currently receive messages from this discussion by email. If you would like to, click ‘Subscribe to discussion’. Subscribe to discussion

Atom RSS