COMP161
Project 2
Tic-Tac-Toe - Python Edition

Spring 2019

> Second verse, same as the first. We'll use the tic-tac-toe game to focus a
> study of programming in python.

Starting with Python

Our first project has all the basic functionality we expect from a pro-
gram: some pure functions, some effect driven procedures, use in-
teractivity via I/O, and even some randomization. By redoing these
things in python we can get started with that language and have a
very concrete, *familiar* set of programming problems for focus on.
As you progress it is important to remember that we want to learn
to do things the Python way and not attempt a literal translation of
our C++ design to Python. It's not just about learning new syntax
but learning how to make practical use of a given language and work
within its expected norms. Don't worry about formal testing but do
use the interactivity of python test your code as your write it.

Version 0

The first version of your program is about developing procedures
that visualize the game state, test for win/loss conditions, and find
the locations of all the allowable moves. There is no running program
here, just some very necessary functionality for tic-tac-toe.

- A procedure, or suite of procedures, to determine if the game has
  been won and by whom.

- Three procedures for converting board positions between 1D and
  2D coordinates: 2D to 1D, 1D to 2D row coordinate, and 1D to 2D
  column coordinate.

- A procedure that produces a vector containing the 1D coordinates
  for all the currently allowable moves.

- A procedure that writes a visual representation of the board to the
  terminal.

Version 1

This version of the program adds procedures for getting and han-
dling a move from a player or, in the case of a random move, an AI.

After completing this version of the program you still only have a library of helpful procedures, but it's worth noting that these procedures more or less cover what you need to get the full game moving.

- A procedure, or series of procedures for getting a legal move from a player.

- A procedure to select a random legal move.

- A procedure to apply a given move, specified as a 1D coordinate, to the board.

Version 2

You will now take the library you've developed and implement a simple tic-tac-toe program that allows the user to play one game against an AI opponent. Don't worry about anything other than being able to play one complete game against the computer.

- A program that allows for a single PvAI game of tic-tac-toe. The AI goes first and uses the strategy of picking any allowable move at random. The program terminates as soon as the game is over.

Version 3

The final version of the program adds repeatable plays, the option to play against other people, and a few other UI bells and whistles.

- Modify the program so that it begins by asking if you'd like to play PvP or PvAI. When the game is over, announce the winner and ask the user if they'd like to play again. If they choose to keep playing, go back to asking what kind of game they want to play.

Optional: Putting the I in AI

Currently, your AI is not very intelligent. It simply picks a possible move at random rather than attempting to determine which of those moves is best. Intelligence implies at least some evaluation of move quality.

We can judge the quality of a move based on how good the board looks to the player after they've made their move. Towards this end we can define a simple function that assigns a numerical value to a given board configuration.

- 200 if the AI has won

- 20 for each two in a row the AI has on the board

- 5 for each one in a row the AI has on the board

The AI now does the following: determine all possible moves, determine the value of the board that results from each of those moves, and choose the move that results in the board state with the highest value. Feel free to play around with other ways of evaluating a board position. If you want to go all out on this then look into the following algorithms: minimax, alpha-beta pruning, and A*-search.

Grading

Your grade is a function of the overall quality of the code you submit and how much of the program you completed as measured by the version requirements listed above. Completeness will determine the broad, letter-range of your grade while quality determines where you fall within that range.

Quality Points

To earn full credit for code quality you must use good programming style, have well written documentation, and a full set of tests for all procedures. Further more, your design should be making good use of helper procedures in order to manage the complexity of the program. Cramming everything into a small number of procedures is likely to cause you to lose some points on design quality but you can earn a good amount of points if the procedures you have are well documented and tested using gTest. On the other hand, you can do an excellent job breaking things down into basic procedures but lose points to sparse or missing documentation and tests. Finally, you should make some effort at producing a decent user experience. You don't need to go overboard one visual elements, but the experience of playing your game should not be unpleasant or difficult to figure out.

Correctness and Completeness Points

Your overall grade range is based on the highest version number you complete as listed below. For example if you complete all of the features listed in version 0 and 1 then you can expect to get something in the C range based on the quality and correctness of what you've submitted. In the event that you also completed some of the features in version 2, then it is possible you'll get as high as a B- so long as your code is of sufficient quality and correctly implements all the required features. However, you cannot boost your grade by means of cherry picking bits and pieces of each version. Having only parts of version 0, 1, and two complete will be treated more or less like an incomplete version 0 project. Finally, above all else, be certain that

your code compiles without error. Code that does not compile will
not receive a passing grade.

Version	Grade
0	D
1	C
2	B
3	A

Timeline

When it's due, submit your code as assignment *proj2* using handin.

Date	Assignment Due
4/18	Begin Project in Lab
4/25	Work time in Lab
5/1	Project Due by end of day.

Table 1: Project Due Dates