Tic Tac Toe Evaluation

Coding conventions

Naming conventions

For my program I have used reasonable variable names so that anyone reading my code will be able to tell what they do. These include: “board”, “player” “counter” and “win”

Use of external libraries

I have used a few external programs and libraries in my code in order to reduce the number of bugs. The first library I used is the “re” library. I used this library to allow me to take user inputs and interpret them as Regex. This reduces the amount of possible bugs in the code as only valid inputs will be allowed meaning that I will not get any type errors. The second module I used is is the time module. This allows me to give a small wait time using the time.sleep() notation. This makes the program seem more natural as the user is not abruptly greeted with a new board as soon as they input their place.

Functions and parameters

I used one function to print the board to the terminal. I decided to use a function for this because printing the board is done multiple times per game. This makes it much more efficient to use a function and call it when needed that to rewrite the code every time. This function does not take any parameters since it does not require any inputs.

Upon reflection of my code, I now see that I would have been more efficient to use a function to test if the user has won a game. This would be much better than the current implementation of if trees.

Quality of solution for the user

Ease of use

My program is fairly easy to use because it is quite intuitive to anyone that already knows the rules of tic tac toe. I have made the program easy to use by adding numbers to the spots on the board as well as telling the use what to input. (numbers one to nine). To play the game, all the user has to do is type in a number for an appropriate spot. The user is prevented from putting a counter in an already taken spot by a short message. To improve the usability of the program, I could add in some features that state whose turn it is as well as displaying the rules at the start of the game.

Features

My program only provides the basic functionality of tic tac toe with no additional features. Users can play the game and can see who wins but that's about it. To improve this, I can add in systems that count the amount of wins as well as getting the user to customise their counter.

Crashes

In previous versions of the code, the user is able to crash the program by inputting a number larger than 9 or by inputting a letter. In version 3, the program no longer crashes because of the regex implementation. I have an else statement that prevents the program crashing because if an error occurs then the user is greeted with a message and is then told to replay the go with a valid input. This prevents the program attempting to work with invalid or erroneous inputs.

Quality of code

How easy is the code to read

I would say that the code is fairly easy to read because it is indented properly and for the most part, has adequate spacing between the lines and sections. To improve the readability I would add in comments.

How easy is it to modify

Some parts of the code are easy to modify because of the use of the use of functions. This means that only one part of the code needs to modified and then that change will be applied to the rest of the code when the function is used. Other parts such as the win condition are not as easy to modify or extend because they do not use function and are instead hardcoded.

How robust is the code

The code is extremely robust and is basically uncrashable. This is because I have used regex to only allow valid user inputs rather than simply hope that the user will follow the instructions given. Furthermore, all end conditions have been accounted for, these being: X wins, O wins, and it’s a draw.

Use of memory

I have only used on function for the layout meaning that I am wasting some memory. Additionally all of my variables are global which again wastes memory. My function does not take any parameters

Efficiency and performance

The layout and user input section of the code is quite efficient as it only uses necessary variables to save memory space. Also because the layout is a function, it is very memory efficient. The win condition section is not as efficient because it uses an if tree rather than a for loop to check if the user has won the game. This is not efficient as the program completes lots of very similar tasks instead of just one loop that runs once per go.