Tic Tac Toe

Planning

Rules:

- Choose a symbol, ‘O’, or ‘X’.
- Take turns to place your symbol on the grid.
- The first player to get a horizontal, diagonal or vertical line of 3 symbols, wins.

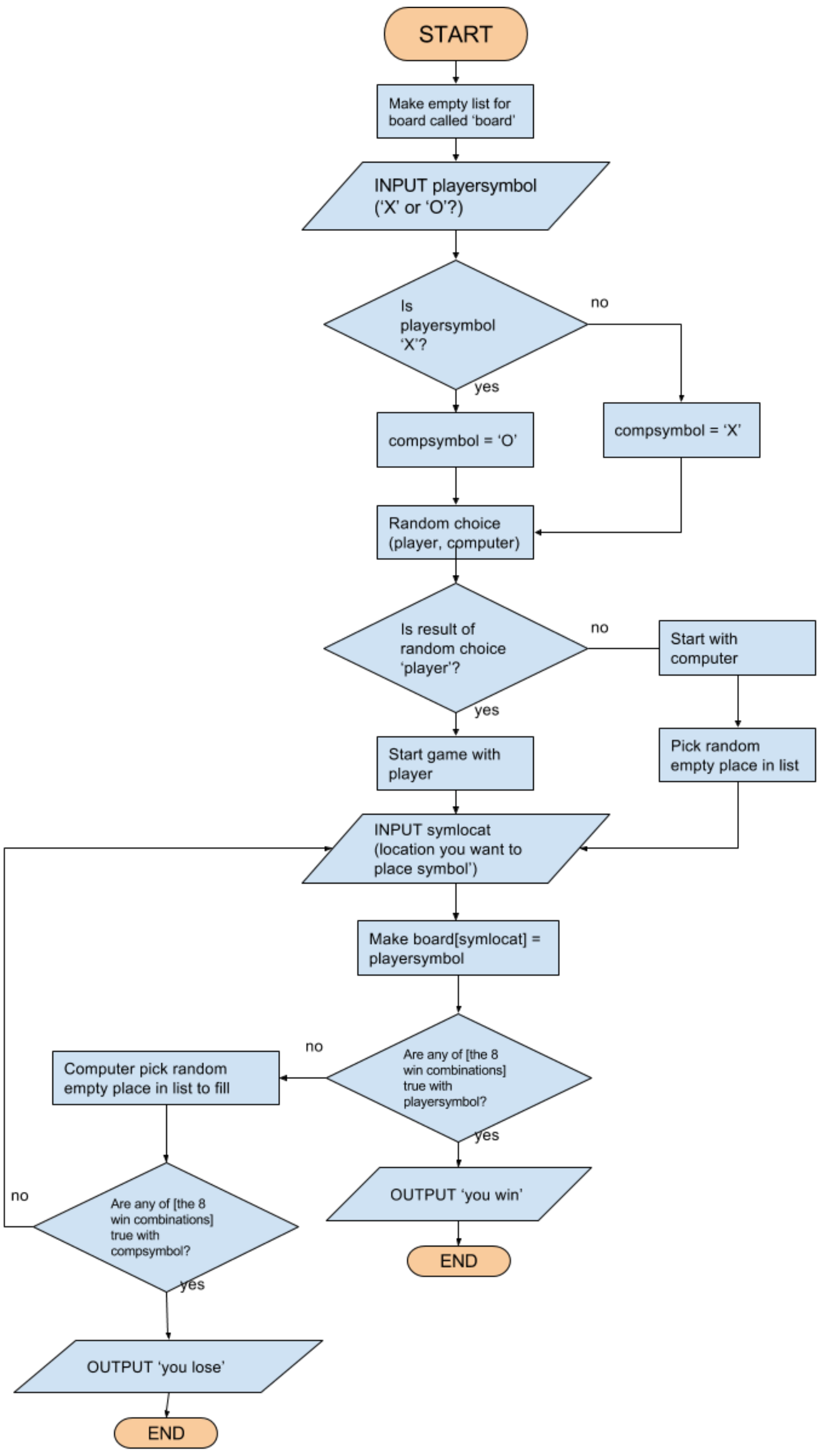
Data structure:

Grid location will use a list, from 1-9 (0-8 index)
- List will be called ‘board’

Constants:
- playersymbol (the symbol the player will use)
- compsymbol (the symbol the computer uses)

Variables:
- whofirst (it will either be ‘player’ or ‘computer’)
- symlocat (number from 1-9 for 3x3 grid)

Flowchart


START
Make empty list for board called 'board'
INPUT playersymbol ('X' or 'O'?)
Is playersymbol 'X'?
no
yes
compsymbol = 'O'
compsymbol = 'X'
Random choice (player, computer)
Is result of random choice 'player'?
no
yes
Start with computer
Start game with player
Pick random empty place in list
INPUT symlocat (location you want to place symbol')
Make board[symlocat] = playersymbol
Are any of [the 8 win combinations] true with playersymbol?
no
yes
Computer pick random empty place in list to fill
OUTPUT 'you win'
END
Are any of [the 8 win combinations] true with compsymbol?
no
yes
OUTPUT 'you lose'
END

Test plan:

Test Num	Description of Test	Test data	Expected outcome
1	Enter ‘x’ as an input	valid	It accepts the ‘x’ as an input
2	Enter ‘X’ as an input	valid	It accepts the ‘X’ as an input
3	Enter ‘o’ as an input	valid	It accepts the ‘o’ as an input
4	Enter O’ as an input	valid	It accepts the ‘O’ as an input
5	Enter ‘e’ (or any other character) as an input	invalid	It does not accept the ‘e’ (or any other character) as an input
6	The game should end when the player wins	valid	The game prints ‘you win’ when the player makes a winning move
7	The game should end when the computer wins.	valid	The game prints ‘you lose’ when the computer makes a winning move.

Evaluation

The coding conventions that you used
Identifier naming conventions
Use of external libraries/modules
I have used the random module - the randint, and random.choice.
Decomposition of code - use of functions and parameters
I used functions to run the code separately for the input after the functions had been defined once.

The quality of your solution for the user
How easy it it to use and play?
It is quite easy to use, by making the numbers correspond to the number pad, it’s easy to visually connect where to place the mark. Any incorrect inputs are recognised, and the player is given instructions and guidance on what to enter.

What features does it provide?
Choice of symbol, random selection of first move, intelligent computer moves,

Can the user crash? Did you take unexpected events into account and write
I took into account unexpected inputs and so the program is unlikely to crash, it will however crash if you enter nothing.

The quality of your code for programmers

How easy it it to read?
I have commented frequently, and the variable and function names I have used are appropriate.

How easy it is to modify or extend?
As it uses functions, it’s easy to trace back to a certain part of code, and modify the code.

Is the code robust? (Can the user crash the program?)
The user can’t crash the program under normal circumstances, (when the code functionality is not reliant on the server.)

Use of memory - global and local variables, use of functions and parameters

The performance of your program - how efficient is the code? Where could it be improved?
The code may not be efficient, as I have used a lot of functions, and have seperated the code into two main cells, one for the defining the functions, and the other for input and running of the game. I could find a more efficient way of programming the AI rather than programming in every possible win pattern to check for a possibility of winning.