Intro to Python Computer Programming with Z. Miller

Tic Tac Toe Game by Shawn Musengo

For my project I wanted to make an interactive game with my limited python abilities. I was trying to keep the rock, paper, and scissor game in mind when I was deciding which game to code in python. I looked for a few days online at different game ideas that I thought were in my skill level and I narrowed them down to tic tac toe, hangman, a word scramble game and finally a text based adventure game. I then narrowed these down by effectiveness and by what I thought would be interesting for my presentation. The text based adventure game I immediately ruled out because I thought it would be boring and the least audience interactive. Then I looked at hang man and ruled this out because in order to make this fun and multi-player ready, I would have had to have made a ton of different hangmen and had a huge word bank. I then weighted the remaining two choices: tic tac toe and word scramble. The word scramble was my initial choice because I thought it would be a better choice to present to the class. I was going to make three or four rounds with three words scrambled in each round. After attempting this, I realized that I thought it was boring and that tic tac toe was a better choice. So, I started over completely.

My tic tac toe game has been an evolving process. I have made four versions so far and am still perfecting the final version. My first version I struggled with making the board print out in the python shell. I got the game to run and for the x or o character to go to the appropriate space, but the entire board didn’t show up until someone won. I fixed this easily but then realized my next problem. I made the board by using a list with characters 0 through 8. I then split these characters with a vertical line piping that made the individual spaces. This numbers were confusing to the player because the first space already looks like an o is placed in it, so I had to redo the entire board. I struggled through a few versions of my game to fix this problem. In my next attempt I tried to make the board starting with the number 1 and go through 9, but python did not like this either. For my third attempt, I tried to make the board out of a 3 x 3 grid that contained [0, 0] in each grid space. This solved the number confusion problem for the player, but confused me in my later programming, so it was eventually changed. I then finally remembered about the %s. Using this allowed me to place exactly what I wanted into each space on my game board. This %s saved me a lot of time once I remember how to use it. This string formatting is very easy and allowed me to place the exact notation that I wanted into these spaces.

The next step in developing my game is to make a function that checks to see if someone has won, and if so ends the game. In order to do this I first had to determine how many ways there were to win tic tac toe. The answer to this is 8, three vertically, three horizontally and two diagonally. I then made a function that checked to see if any of these eight combinations was fulfilled and if so it ended the game and a winner was announced. This process, just as the board development, was evolving the entire time. I first made a function that checked the first horizontal line and if that was full it ended the game. I then made another function that was called checkAll that checked the other seven ways to win. I used tuples as my way of writing the winning combinations. In my final version I made a variable called test into my is-winner function and stated that if any of these eight ways were equal to test that then the function was equal to true. This true then stopped the game and announced a winner. If the function came back false then the program goes on to the next function which was a while loop. I basically stated that while any of these eight winning combinations were not satisfied that the game would continue to allow the user to place x’s or o’s in spaces that were not already filled.

The next step was to import random and write a function that allowed the computer to play against a player. I did this in all of my version by using randomseed (). This all allowed for the computer to choose a number. Then I set the range to that of my board. I used the random.randrange(1, 9). This allowed the computer to pick a number space on my board and then I set it to integer so that it had to be a whole number and not a fraction. This step also took me numerous attempts to get it the way I wanted it. I still have not perfected this by any means. It is very easy to beat the computer at this game because it does not know human strategy or any strategy for that matter. It simply picks a number and places an o in that space until they are all taken or until a winner is decided.

I have added a lot of raw input springs into my game. I first have the title Tic Tac Toe printed out at the top of the game. I have one that asks for the player’s name. I have one that tells the player that it’s their move. I have one that tells the player if they choose a spot on the board that is already taken. I have one that says they have won to game, tied the game or lost the game. I have one that asks if you want to play another game and finally, I have one that says goodbye once you are finished playing.

I made my game a multiplayer game to that it would be more interactive than a single player game. I added a second player into my while loop and even made it so that if you enter the name python the computer will be the second player. This is a nice feature because it is single player friendly as well as multiplayer accepting. This is actually the part that includes the random string formatting. It is only activated once python is typed in for the player 2 name. If player 2 is another name besides python it will be just as player 1 is and allows the user to make their own board choices. The only limitation is that player one is always x and player two is always o. I don’t think this is a huge problem so I left it.

When deciding where each player choses to place their letter I had to take into account the fact that python starts with the number 0 and that my list started with the letter 1. I had to write a function that subtracted one from and space that was chosen. This was very tricky and confused me up until my latest version of the game. I kept forgetting about the zero index and constantly struggled with the wrong placing of x or o.

Other features in my game that one might overlook are the function that only allows for 9 moves in a given game. This prohibits doubling up on a single space or the game failing to end with a winner. I also used to % to replace players names throughout the entire coding process. This makes it so that the game remembers the players name and that they don’t have to keep entering it. The game also uses this to be more personal and tell the players who’s move and who wins or loses. At the end of the game I made a function that asks the player(s) if they want to play again. This was then set to lower case and if initiated started up another game.

This is a spot where I failed in my first few attempts. I could never get a second player to work quite right. I even struggled with the computer generated random guessing for a few versions. I finally figured this out with help and a lot of determination. I also failed in prior versions with getting another game to start after the first one was complete. This was a simple fix that I made after reviewing my python book and loops. I just had to write some code that went back to the beginning of the function once it was complete.

I have learned a lot about coding by making this game. I never thought I would be able to make a game, even one this simple, with python. I am pretty happy with my progress as a while considering I have zero programming experience except this intro class. I look forward to continuing with my programming and learning to make more complex and meaningful programs/objects.