Started at 7:00 pm, August 4
Code finished at 9:25

I took this challenge step by step. I didn’t end up writing notes in this google doc until I had finished because I worked the problem out in my head and through frequent code testing in pycharm itself.
First I wrote the function to make the grid of periods. I used a double for loop to nest lists inside each other, making a grid. Then I wrote a simple print grid function, just iterating over the grid spot by spot and printing out the value with a space after it so it looked neat. I then made a mark grid function which took two inputs and marked the grid I’d made at those inputs.
I then wrote a function to get coordinates from the player. Much of this function is devoted to sanitizing the inputs so that they come out as a clean tuple. Once I settled on having the function return the coordinates as a tuple, I realized that I would have to modify the mark grid function to unpack the tuple before marking the grid, so I changed that.
I then wrote a function to check if given coordinates are occupied or not. It returns a boolean. I planned to use this later to check for both the player and the computer if their inputs were occupied. I then wrote the function for the computer to provide coordinates. I made it select two random coordinates, and then I had it check if those coordinates were occupied, and if they were, to select two random coordinates again.
At this point, I started assembling the game function. I knew that I would have to create a turn variable that would switch back and forth. It was a bit tricky knowing how to start a given player first since I wanted the game to run in a while loop until broken, so I had the computer take an extra turn at the beginning if the player selected to start second. I then had the program determine whose turn it was and run that respective player’s turn function. Those functions (player_move and computer_move) were just an assembly of the previous functions in order, so that the coordinates are obtained from the given player and checked to see if occupied, then the grid is marked and printed.
This led me to the point in the game where I would check if anyone had won yet. There were four avenues to check: if columns or rows had three in a row of a given value, and if the diagonals had three in a row of a given value. For the row and column functions, I had it add up the number of times a value appeared in each row or column. If this number ever reached three, the function returned True, indicating someone had one. Otherwise it returned false. The diagonal functions run along the diagonals using a zipped range to check if they have three in a row. I then had a function (someone_won) to check if any of the four previous functions returned True, indicating someone had won. Finally, I had a function (winner_declared) that made the value checked in each of these functions set to the current player, and then it returned either that value (current turn’s player) or False if no one had won yet.
From here, my program was mostly finished. I implemented the turn counter to account for a tie, and then I wrote the part to loop games for the player.
Thank you for considering my application and program. I would love to work for you guys.