In the previous day, we worked on creating the player classes and then we defined a Tic-Tac-Toe class and initialized it with a board parameter and we also defined a method to get available moves. Moving on, the next thing we are going to do is to create a method inside Tic-Tac-Toe that actually makes the move after we have selected the square we want to move to. If it is a valid move, then we make the move i.e. assign the square to the letter.

def make\_move(self, square, letter):

        if self.board[square] == " ":

            self.board[square] = letter

            return True

        return False

That simple enough function would help us make the moves. Next we want to define a function to help us decide the winner. A player wins if they have three in a “row”. They can have this in three ways – three in a row, three in a column or three in a diagonal. To check for the winner we are going to need the square and we are going to need the letter do our method would look like this.

def winner(self, square, letter):

        # winner if 3 in a row anywhere (row, column or diagonal)

        # first check the row

        row\_ind = square // 3 #divide whatever square is by 3 and then round it down so its always going to be either 0, 1 or 2

        row = self.board[row\_ind \* 3 : (row\_ind + 1) \* 3]

        if all([spot == letter for spot in row]): #returns true if everything is true else returns false

            return True

Here, we check the rows first. We get the row index by performing a floor division (i.e. //). This would divide the square by three and round it down so it is always going to be 0, 1 or 2 for example if we play in square 8 (keep in mind that it’s a 9 game box so index is 0 up to 8), our row index is going to be 8/3 which is 2.667 but the since we are doing a floor division (8//3), we would get two (2) so last row (three row means index 0, 1, 2). Next, we use that index number we have gotten to get the row. Since a row consists of three values, we need to slice the board so we can have just those values. For example, if our row index was zero (0), to get our row, the starting value for the slice would be 0 \* 3 and the end value would be (0 + 1) \* 3. So we would have <code>row = self.board[0 : 3]</code> and of course this would return items in position 0, 1 and 2 of the board (remember when slicing a list, it is start from the initial value provided up to but not including the final value provided). Next we loop through our row list. For spot in row, if every spot is equal to letter return true. The way <code>all()</code> works is it returns true if everything is true, else it returns false. <br />

After checking the rows, we check the column next. This is similar to the rows, but this time, we use the modulus (%) operator. When you divide numbers using the modulus operator, we get back the remainder. For example, 3%3 returns zero (0), 2%3 returns two (2), 7%3 returns one (1) etc. so any square on the first column is going to have the index of 0 etc.

col\_ind = square % 3

        column = [self.board[col\_ind+i\*3] for i in range(3)]

        if all([spot == letter for spot in column]):

            return True

The rest of the code is pretty similar to that of the row checking. Next we check the diagonals. Since we have a square consisting of 9 spaces, our diagonals if divided with modulus 2, should give us zero i.e. our diagonals are in positions, 0, 2, 4, 6, 8. If the square position modulus 2 is not equal to zero then it is not on a diagonal.

if square % 2 == 0: #if it's an even number

            diagonal1 = [self.board[i] for i in [0, 4, 8]] #left to right diagonal

            if all([spot == letter for spot in diagonal1]):

                return True

            diagonal2 = [self.board[i] for i in [2, 4, 6]] #right to left diagonal

            if all([spot == letter for spot in diagonal2]):

                return True

We have two diagonals, from top left to bottom right and then from top right to bottom left. For diagonal1, we create a list of the squares in positions 0, 4 and 8 and then we check if all of them are equal. If yes, we return true. We do the same for the second diagonal. At the end, if all fails, i.e. if there is no match on the rows, columns or on the diagonals, then we return false.

def winner(self, square, letter):

        # winner if 3 in a row anywhere (row, column or diagonal)

        # first check the row

        row\_ind = square // 3 #divide whatever square is by 3 and then round it down so its always going to be either 0, 1 or 2

        row = self.board[row\_ind \* 3 : (row\_ind + 1) \* 3]

        if all([spot == letter for spot in row]): #returns true if everything is true else returns false

            return True

        #check the column next

        col\_ind = square % 3

        column = [self.board[col\_ind+i\*3] for i in range(3)]

        if all([spot == letter for spot in column]):

            return True

        #check diagonals

        #but only if the square is an even number (0, 2, 4, 6, 8)

        #these are the only moves possible to win a diagonal

        if square % 2 == 0: #if it's an even number

            diagonal1 = [self.board[i] for i in [0, 4, 8]] #left to right diagonal

            if all([spot == letter for spot in diagonal1]):

                return True

            diagonal2 = [self.board[i] for i in [2, 4, 6]] #right to left diagonal

            if all([spot == letter for spot in diagonal2]):

                return True

        # if all of these fail

        return False

Now that we have a method that check for winners, we would want to check if there is a winner after every move is played so we would update our <code>make\_move()</code> method.

def make\_move(self, square, letter):

        #if valid move, then make the move (assign square to letter)

        #then return true. if invalid, return false

        if self.board[square] == " ":

            self.board[square] = letter

            if self.winner(square, letter):

                self.current\_winner = letter

            return True

        return False

After we make the move to the square, we check if there is a winner. If there is a winner, we set the current winner to the letter.