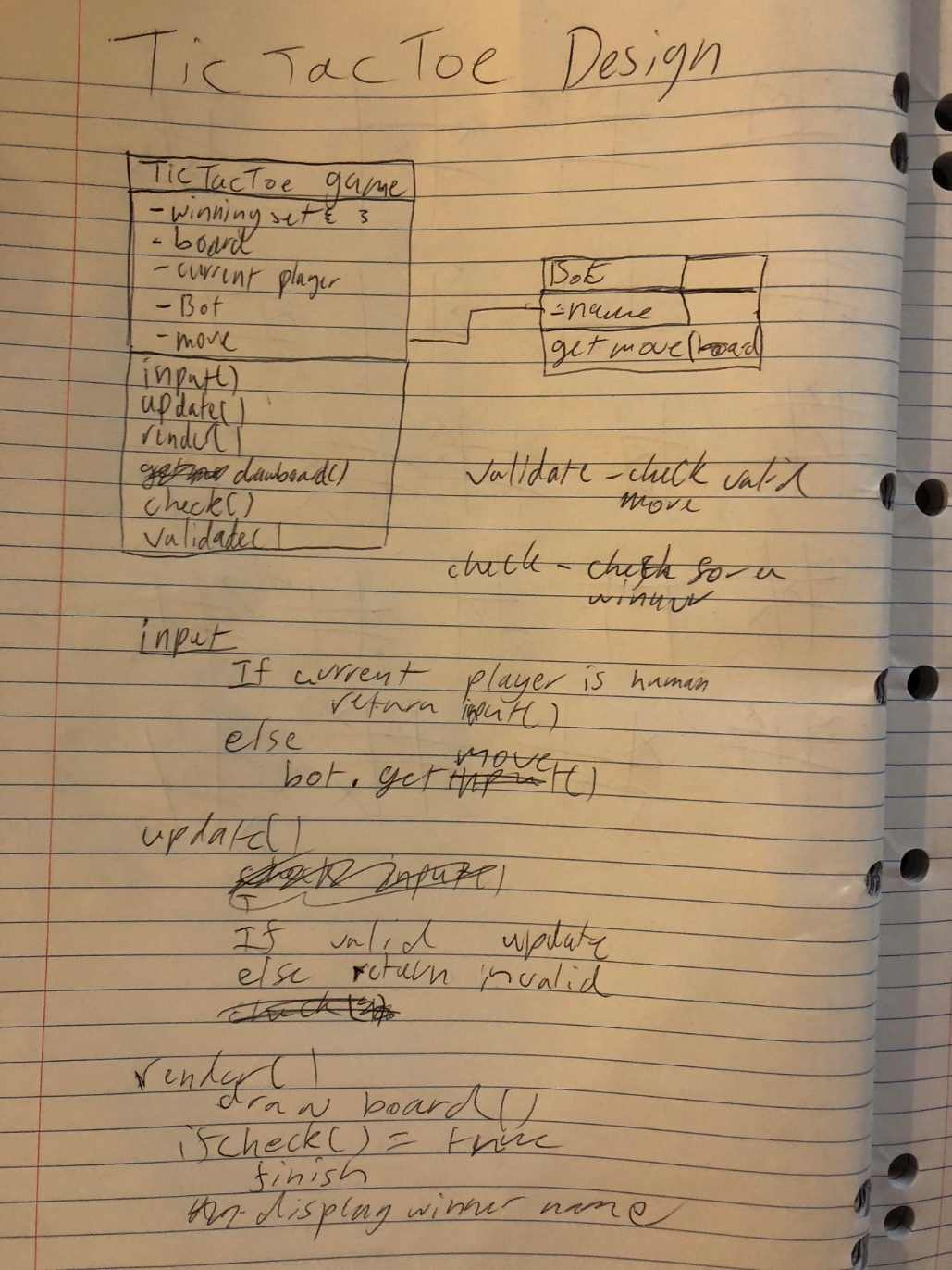
**COS30002**

**Lab Report – Task 3 (14/3/2021)**

Ryan Chessum 102564760

* Designed TicTacToe program
* Learnt about basic game architecture and game loops
* Designed 2 AI bots
* Implemented designs into code
* Made both AI bots play against each other

**Software Design**



**AI design**

**Bot 1 (centre strategy):**

* If winning move available select the last spot
* If opponent can make a winning move select the 3rd space to block them
* If centre space isn’t taken select the centre space
* Else random

**Bot 2 (defensive):**

* If winning move available select the last spot
* If opponent can make a winning move select the 3rd space to block them
* Pick a random space adjacent to opponents last move
* Else random

**Bot Code**

    def get\_definately\_not\_an\_ai\_move(self):

        '''Get a "human" players raw input '''

        if self.go\_last\_space() <= 8:

            return self.go\_last\_space()

        #go to a random space next to the last move

        if self.move != None:

            return choice(self.adjacent.get(self.move, (randrange(9))))

        return randrange(9)

    def get\_ai\_move(self):

        '''Get the AI's next move '''

        if self.go\_last\_space() <= 8:

            return self.go\_last\_space()

        if self.board[4] == ' ':

            return 4

        return randrange(9)

    def go\_last\_space(self):

        #search the board for potential winning spaces and return the last one

        board = self.board

        for row in self.WIN\_SET:

            if (board[row[0]] == board[row[1]] != ' ') and (board[row[2]] == ' '):

                return row[2]

            if (board[row[1]] == board[row[2]] != ' ') and (board[row[0]] == ' '):

                return row[0]

            if (board[row[0]] == board[row[2]] != ' ') and (board[row[1]] == ' '):

                return row[1]

        #return an invalid move otherwise to tell the ai to make a different move

        return 9

    adjacent = {

        0: (1, 3),

        1: (0, 4, 2),

        2: (1, 5),

        3: (0, 4, 6),

        4: (1, 3, 5, 7),

        5: (2, 4, 8),

        6: (3, 7),

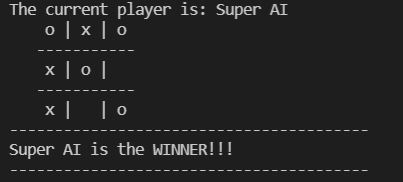
        7: (6, 4, 8),

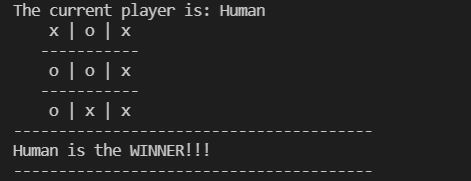
        8: (7, 5)

    }

**AI battles**

When the AI played against each other it would always end in a tie. But after I changed the program so the starting player was randomised things became more interesting. It would still often be a tie but bot 1 (the one that goes for the centre) would win every couple of rounds. I ran the program many times and I only managed to get the other AI to win once.

****

****