```python
# Tic-Tac-Toe
# Basic game loop structure
#       - move_input()
#       - update_board()
#       - render_board()
# WIN_SET contains sets of all the spots on the board that equal a win
import random

class Tictactoe:

    #constructor
    def __init__(self):

        #static set of valid 'win' moves
        self.WIN_SET = (
           (0,1,2), (3,4,5), (6,7,8),        #horizontal
           (0,3,6), (1,4,7), (2,5,8),        #vertical
           (0,4,8), (2,4,6))                     #diagonal

        self.board = [' '] * 9
        self.current_player = 'x'
        self.player_choice = ' '
        self.players = {'x': ' ', 'o': ' ' }

        self.winner = None
        self.move = None


    #methods
    def choose_side(self):
        while True:
            t.player_choice = input("""Please choose a side: 'X' or 'O': """)
            t.player_choice = t.player_choice.lower()

            #make sure its a valid choice
            if t.player_choice == 'x' or t.player_choice == 'o':
                print("You've chosen: " + t.player_choice)
                break
            else:
                print("Invalid entry.")
                t.player_choice = ' '
                continue

        #player 'x' goes first
        #player 'o' ai goes first
        if t.player_choice == 'x':
            self.players['x'] = "Human"
            self.players['o'] = "Super AI"
        else:
            self.players['x'] = "Super AI"
            self.players['o'] = "Human"


    def check_move_valid(self):
        global move

        if move <= 8 and move >= 0:
            if t.board[move] == ' ':
                return True
            else:
                print("That area is already filled. Please enter a different number.")
                return False
        else:
            print("Not a valid position. Must be a number between [0-8].")


    def check_win_result(self):

        #if matching symbols in WIN_SET (e.g. 0,4,8)
        #row[0] = 0, row[1] = 4, row[2] = 8
        for row in self.WIN_SET:
            if t.board[row[0]] == t.board[row[1]] == t.board[row[2]] != ' ':
                print(t.board[row[0]] + " wins!")
                t.winner = "true"

        if ' ' not in t.board and t.winner == None:
            print("Tie")
            t.winner = "false"


    #================================================================================
    # Human and AI functions

    def human_move(self):
        return int(input("Enter a number [0-8]: "))

    def ai_move(self):
        return random.randrange(9)

    def ai_move_hard(self):
        #best move if ai goes first
        if t.board[4] == ' ':
            return 4
        else:
            #get positions the player has taken so far
            player_pos = [i for i in range(len(t.board)) if t.board[i] !=
t.current_player and t.board[i] != ' ']

            #block the players winning move by comparing what squares the player has
against what they need to win
            if len(player_pos) == 2:
                for row in self.WIN_SET:
                    if player_pos[0] in row and player_pos[1] in row:
                        #find the last square the player needs to win and take it!
                        if player_pos[0] != row[0]:
                            return row[0]
                        elif player_pos[1] != row[1]:
                            return row[1]
                        else:
                            return row[2]
                    else:
                        return random.choice(t.choose_empty_pos())
            else:
                return random.choice(t.choose_empty_pos())

    def choose_empty_pos(self):
        #chooses a random, empty square from the remaining squares on the board
        empty_pos = [i for i in range(len(t.board)) if t.board[i] == ' ']
        return empty_pos


    #================================================================================
    # Game loop methods
    def next_move(self):
        global move

        if t.current_player == 'x':
            if t.players['x'] == "Human":
                move = t.human_move()
                print("--Player--")
            else:
                t.players['x'] == "Super AI"
                move = t.ai_move_hard()
                print("--Computer--")
        else:
            if t.players['o'] == "Human":
                move = t.human_move()
                print("--Player--")
            else:
                t.players['o'] == "Super AI"
                move = t.ai_move_hard()
                print("--Computer--")


    def update_model(self):
        if t.check_move_valid():

            #set the list element to match the current players symbol
            t.board[move] = t.current_player
            #print(t.board[0:8])

            #check to see if anyone has won
            t.check_win_result()

            #switch players
            if t.current_player == 'x':
                t.current_player = 'o'
            else:
                t.current_player = 'x'


    def render_board(self):
        print('-------------')
        print('| %s | %s | %s |' % (self.board[0], self.board[1], self.board[2]))
        print('-------------')
        print('| %s | %s | %s |' % (self.board[3], self.board[4], self.board[5]))
        print('-------------')
        print('| %s | %s | %s |' % (self.board[6], self.board[7], self.board[8]))
        print('-------------')


    def position_board(self):
        print("HOW TO PLAY: Enter a number when prompted to fill the corresponding 
space")
        print('-------------')
        print('| 0 | 1 | 2 |')
        print('-------------')
        print('| 3 | 4 | 5 |')
        print('-------------')
        print('| 6 | 7 | 8 |')
        print('-------------')


#===============================================================================
# Main
```

```python
#================================================================================

if __name__ == '__main__':

    t = Tictactoe()
    t.position_board()       #shows board for moveset
    t.choose_side()          #choose a side

    #game loop
    while t.winner is None:
        t.next_move()
        t.update_model()
        t.render_board()
```