AI Experiment-1

February 4, 2024

[ ]:

```
#Brute Force Approach
```

[ ]:

```
import random

board = [' ' for x in range(9)]
def main():
    print('Game started')
    print_board()
    game_end = False
    while not game_end:
        print('Player turn')
        player_turn()
        print_board()
        if check_winner(board):
            print('Player won')
            game_end = True
            break

        print('Computer turn')
        computer_move = computer_turn()
        if computer_move != -1:
            board[computer_move] = 'O'
            print_board()
            if check_winner(board):
                print('Computer won')
                game_end = True
                break

        if board.count(' ') < 1:
            print('Tie game')
            game_end = True

    print('Game ended')

def print_board():
    print(board[0] + ' | ' + board[1] + ' | ' + board[2])
    print('---------')
    print(board[3] + ' | ' + board[4] + ' | ' + board[5])
    print('---------')
    print(board[6] + ' | ' + board[7] + ' | ' + board[8])

def check_winner(board):
    if ((board[0] == board[1] == board[2] != ' ') or
        (board[3] == board[4] == board[5] != ' ') or
        (board[6] == board[7] == board[8] != ' ')):
        return True

    if ((board[0] == board[3] == board[6] != ' ') or
        (board[1] == board[4] == board[7] != ' ') or
        (board[2] == board[5] == board[8] != ' ')):
        return True

    if ((board[0] == board[4] == board[8] != ' ') or
        (board[2] == board[4] == board[6] != ' ')):
        return True

    return False

def player_turn():
    made_move = False
    while not made_move:
        player_input = input('Enter a position (1-9) ')
        try:
            player_move = int(player_input)
            if player_move < 1 or player_move > 9:
                print('Enter a valid position')
            else:
                player_position = player_move - 1 # player index in board
                if board[player_position] != ' ':
                    print('Position is already taken')
                else:
                    board[player_position] = 'X'
                    made_move = True

        except:
            print('Enter a valid number')


def computer_turn():

    available_moves = [pos for pos, value in enumerate(board) if value == ' ']

    move = -1
    for i in available_moves:
        new_board = board[:]
        new_board[i] = 'O'
        if check_winner(new_board):
            move = i
            return move

    for i in available_moves:
        new_board = board[:]
        new_board[i] = 'X'
        if check_winner(new_board):
            move = i
            return move

    avalable_corners = []
    for i in available_moves:
        if i in [0, 2, 6, 8]:
            avalable_corners.append(i)

    if len(avalable_corners) > 0:
        random_index = random.randrange(0, len(avalable_corners))
        move = avalable_corners[random_index]
        return move

    if 4 in available_moves:
        move = 4
        return move

    avalable_edges = []
    for i in available_moves:
        if i in [1, 3, 5, 7]:
            avalable_edges.append(i)

    if len(avalable_edges) > 0:
        random_index = random.randrange(0, len(avalable_edges))
        move = avalable_edges[random_index]
        return move

    return move

if __name__ == '__main__':

    main()
```

```
Game started
  |   |
---------
```

```
  |   |
---------
  |   |
Player turn
Enter a position (1-9)  2
  | X |
---------
  |   |
---------
  |   |
Computer turn
O | X |
---------
  |   |
---------
  |   |
Player turn
Enter a position (1-9)  5
O | X |
---------
  | X |
---------
  |   |
Computer turn
O | X |
---------
  | X |
---------
  | O |
Player turn
Enter a position (1-9)  3
O | X | X
---------
  | X |
---------
  | O |
Computer turn
O | X | X
---------
  | X |
---------
O | O |
Player turn
Enter a position (1-9)  4
O | X | X
---------
X | X |
---------
```

```
O | O |
Computer turn
O | X | X
---------
X | X |
---------
O | O | O
Computer won
Game ended
```

[ ]:

```
#Heuristic Approach
```

[ ]:

```
import random

class TicTacToe:
    def __init__(self):
        self.board = [' ' for _ in range(9)]  # 3x3 Tic Tac Toe board
        self.current_winner = None  # Keep track of the winner

    def print_board(self):
        for row in [self.board[i * 3:(i + 1) * 3] for i in range(3)]:
            print('| ' + ' | '.join(row) + ' |')

    @staticmethod
    def print_board_nums():
        number_board = [[str(i) for i in range(j * 3, (j + 1) * 3)] for j in␣
 ↪range(3)]
        for row in number_board:
            print('| ' + ' | '.join(row) + ' |')

    def available_moves(self):
        return [i for i, spot in enumerate(self.board) if spot == ' ']

    def empty_squares(self):
        return ' ' in self.board

    def num_empty_squares(self):
        return self.board.count(' ')

    def make_move(self, square, letter):
        if self.board[square] == ' ':
            self.board[square] = letter
            if self.winner(square, letter):
                self.current_winner = letter
            return True
        return False
    def winner(self, square, letter):
        # Check row
        row_ind = square // 3
        row = self.board[row_ind*3:(row_ind+1)*3]
        if all([spot == letter for spot in row]):
            return True
        # Check column
        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
        if square % 2 == 0:
            diagonal1 = [self.board[i] for i in [0, 4, 8]]
            if all([spot == letter for spot in diagonal1]):
                return True
            diagonal2 = [self.board[i] for i in [2, 4, 6]]
            if all([spot == letter for spot in diagonal2]):
                return True
        return False

def play(game, x_player, o_player, print_game=True):
    if print_game:
        game.print_board_nums()

    letter = 'X'  # Starting letter
    while game.empty_squares():
        if letter == 'O':
            square = o_player.get_move(game)
        else:
            square = x_player.get_move(game)

        if game.make_move(square, letter):
            if print_game:
                print(letter + f' makes a move to square {square}')
                game.print_board()
                print('')  # Empty line

            if game.current_winner:
                if print_game:
                    print(letter + ' wins!')
                return letter  # Ends the loop and exits the game
            letter = 'O' if letter == 'X' else 'X'  # Switch player

        elif print_game:
            print('It\'s a tie!')
```

```
class HumanPlayer:
    def __init__(self, letter):
        self.letter = letter

    def get_move(self, game):
        valid_square = False
        val = None
        while not valid_square:
            square = input(self.letter + '\'s turn. Input move (0-8): ')
            try:
                val = int(square)
                if val not in game.available_moves():
                    raise ValueError
                valid_square = True
            except ValueError:
                print('Invalid square. Try again.')
        return val

class RandomComputerPlayer:
    def __init__(self, letter):
        self.letter = letter

    def get_move(self, game):
        square = random.choice(game.available_moves())
        return square

if __name__ == '__main__':
    x_player = HumanPlayer('X')
    o_player = RandomComputerPlayer('O')
    t = TicTacToe()
    play(t, x_player, o_player, print_game=True)
```

```
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
X makes a move to square 7
|   |   |   |
|   |   |   |
|   | X |   |

O makes a move to square 3
|   |   |   |
| O |   |   |
|   | X |   |

X makes a move to square 1
|   | X |   |
| O |   |   |
|   | X |   |

O makes a move to square 0
| O | X |   |
| O |   |   |
|   | X |   |

X makes a move to square 2
| O | X | X |
| O |   |   |
|   | X |   |

O makes a move to square 4
| O | X | X |
| O | O |   |
|   | X |   |

Invalid square. Try again.
X makes a move to square 6
| O | X | X |
| O | O |   |
| X | X |   |

O makes a move to square 8
| O | X | X |
| O | O |   |
| X | X | O |

O wins!
```