**Experiment - 3**

Samkit Shah

2019130060

TE Comps

Batch - C

**Aim**:

To build a Tic-Tac-Toe using A\* Algorithm.

Code:

def printBoard(board):

    print(board[1] + '|' + board[2] + '|' + board[3])

    print('-+-+-')

    print(board[4] + '|' + board[5] + '|' + board[6])

    print('-+-+-')

    print(board[7] + '|' + board[8] + '|' + board[9])

    print("\n")

def spaceIsFree(position):

    if board[position] == ' ':

        return True

    else:

        return False

def insertLetter(letter, position):

    if spaceIsFree(position):

        board[position] = letter

        printBoard(board)

        if (checkDraw()):

            print("Draw!")

            exit()

        if checkForWin():

            if letter == 'X':

                print("Bot wins!")

                exit()

            else:

                print("Player wins!")

                exit()

        return

    else:

        print("Can't insert there!")

        position = int(input("Please enter new position:  "))

        insertLetter(letter, position)

        return

def checkForWin():

    if (board[1] == board[2] and board[1] == board[3] and board[1] != ' '):

        return True

    elif (board[4] == board[5] and board[4] == board[6] and board[4] != ' '):

        return True

    elif (board[7] == board[8] and board[7] == board[9] and board[7] != ' '):

        return True

    elif (board[1] == board[4] and board[1] == board[7] and board[1] != ' '):

        return True

    elif (board[2] == board[5] and board[2] == board[8] and board[2] != ' '):

        return True

    elif (board[3] == board[6] and board[3] == board[9] and board[3] != ' '):

        return True

    elif (board[1] == board[5] and board[1] == board[9] and board[1] != ' '):

        return True

    elif (board[7] == board[5] and board[7] == board[3] and board[7] != ' '):

        return True

    else:

        return False

def checkWhichMarkWon(mark):

    if board[1] == board[2] and board[1] == board[3] and board[1] == mark:

        return True

    elif (board[4] == board[5] and board[4] == board[6] and board[4] == mark):

        return True

    elif (board[7] == board[8] and board[7] == board[9] and board[7] == mark):

        return True

    elif (board[1] == board[4] and board[1] == board[7] and board[1] == mark):

        return True

    elif (board[2] == board[5] and board[2] == board[8] and board[2] == mark):

        return True

    elif (board[3] == board[6] and board[3] == board[9] and board[3] == mark):

        return True

    elif (board[1] == board[5] and board[1] == board[9] and board[1] == mark):

        return True

    elif (board[7] == board[5] and board[7] == board[3] and board[7] == mark):

        return True

    else:

        return False

def checkDraw():

    for key in board.keys():

        if (board[key] == ' '):

            return False

    return True

def playerMove():

    position = int(input("Enter the position for 'O':  "))

    insertLetter(player, position)

    return

def compMove():

    bestScore = -800

    bestMove = 0

    for key in board.keys():

        if (board[key] == ' '):

            board[key] = bot

            score = astar(board, 0, False)

            board[key] = ' '

            if (score > bestScore):

                bestScore = score

                bestMove = key

    insertLetter(bot, bestMove)

    return

def astar(board, depth, isMaximizing):

    if (checkWhichMarkWon(bot)):

        return 1

    elif (checkWhichMarkWon(player)):

        return -1

    elif (checkDraw()):

        return 0

    if (isMaximizing):

        bestScore = -800

        for key in board.keys():

            if (board[key] == ' '):

                board[key] = bot

                score = astar(board, depth + 1, False)

                board[key] = ' '

                if (score > bestScore):

                    bestScore = score

        return bestScore

    else:

        bestScore = 800

        for key in board.keys():

            if (board[key] == ' '):

                board[key] = player

                score = astar(board, depth + 1, True)

                board[key] = ' '

                if (score < bestScore):

                    bestScore = score

        return bestScore

board = {1: ' ', 2: ' ', 3: ' ',

         4: ' ', 5: ' ', 6: ' ',

         7: ' ', 8: ' ', 9: ' '}

printBoard(board)

print("Computer goes first! Good luck.")

print("Positions are as follow:")

print("1, 2, 3 ")

print("4, 5, 6 ")

print("7, 8, 9 ")

print("\n")

player = 'O'

bot = 'X'

global firstComputerMove

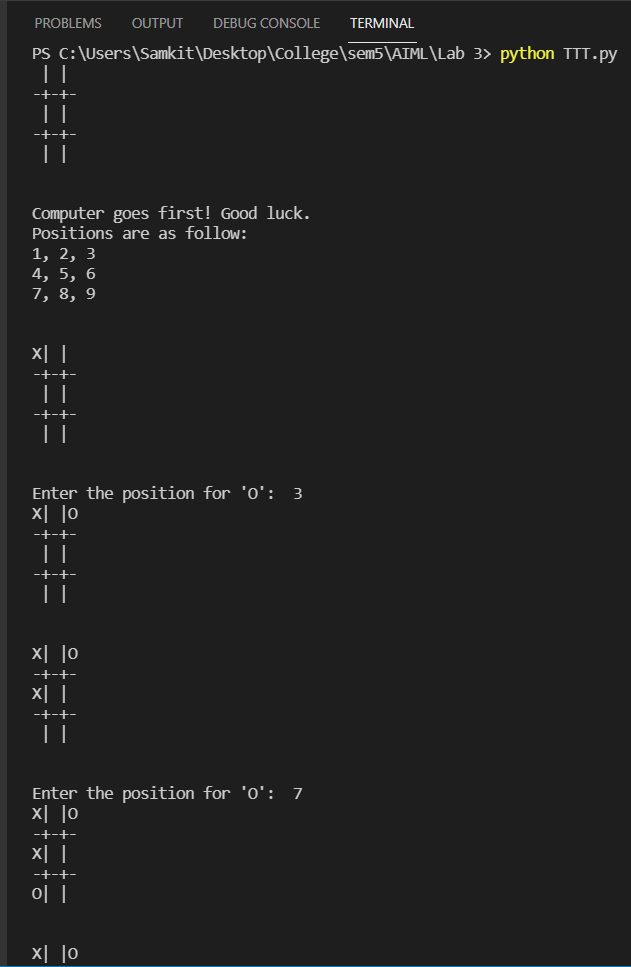
firstComputerMove = True

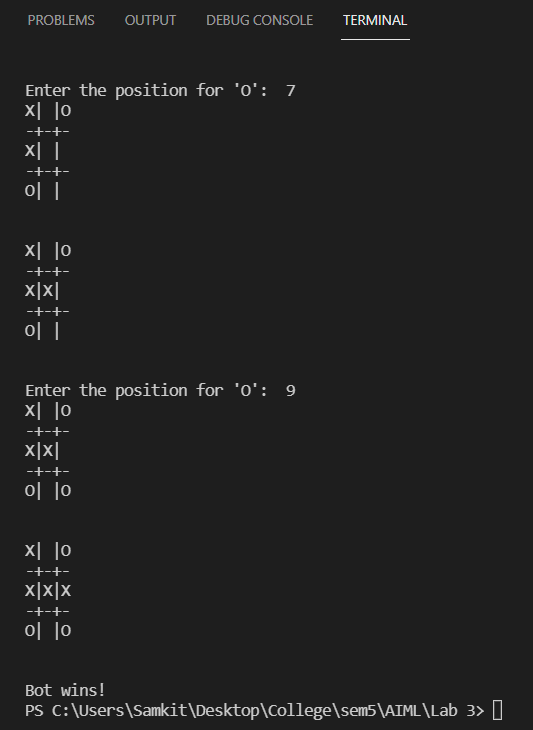
while not checkForWin():

    compMove()

    playerMove()

**Output**:





Conclusion:

1) With the help of above code I developed a Tic-Tac-Toe game using A\* Algorithm