AI EXP - 2: AGENT PROGRAM

Team : Automata Lab

RA1911031010131 - Namrata Mishra

RA1911031010138 - Husna Qasim

RA1911031010130 - Thinakar R

RA1911031010128 - Veer Viswajeet Swami

RA1911031010129 - Vijay Rengaraj R

*Problem Name : TIC-TAC-TOE

***Problem Statement :** Program a two-person game of Tic -Tac- Toe. The game is played on a three by three board. Each player has a marker. One player has an 'X', the other an 'O'. Players alternate turns to place their marker on the board. The first player to get three in a row either diagonally, horizontally, or vertically, wins the games. In the event all squares are taken on the board without a winner then it is a tie. The program should set up the game by asking for the names of the players. Player one should be assigned an 'X' as their marker, player two should be assigned the 'O'. After the game has been completed, the program should congratulate the winner by name. The players should then have the option to play again. If they decide to play again, then the program should keep track of the number of times each player has won and display that information at the end of each game. You may not assume that any input the user provides you is initially valid.If the information provided by the user at any stage of the program is invalid, the program should reprompt until valid information is provided.

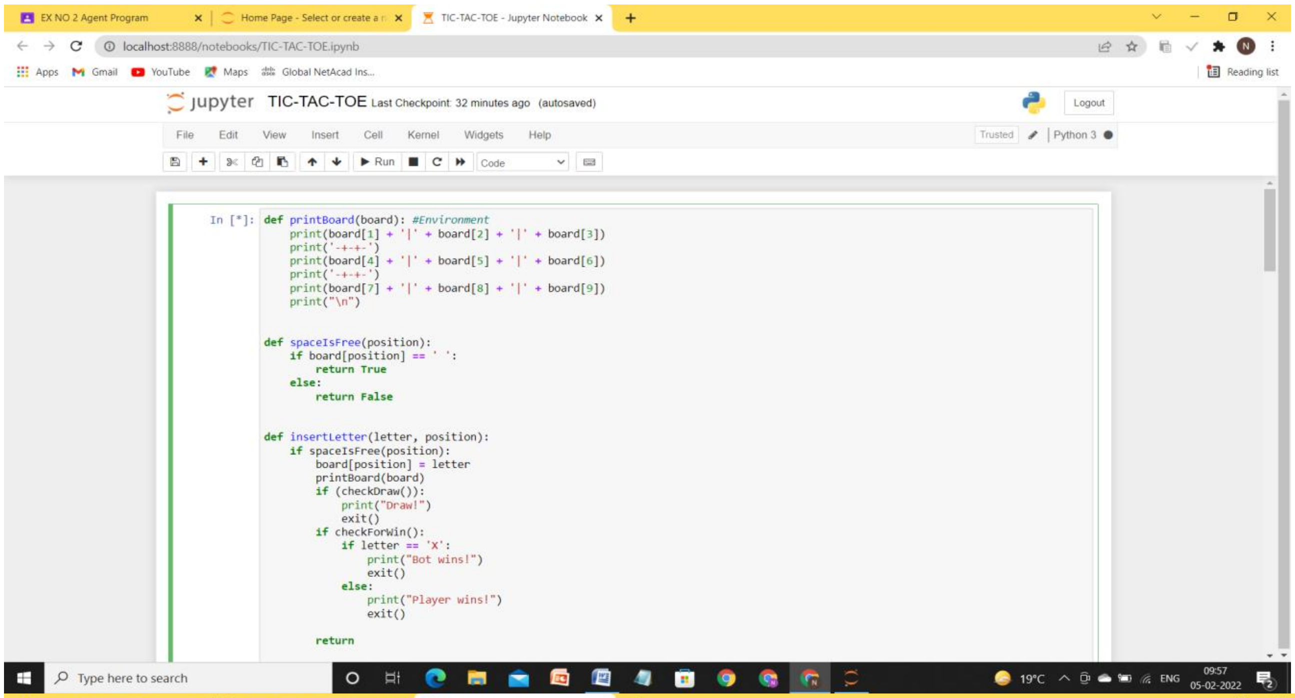
*AGENT USED: Goal-based agent

*Code:

```
def printBoard(board):     #Environment

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

   print('-+-+-')
```

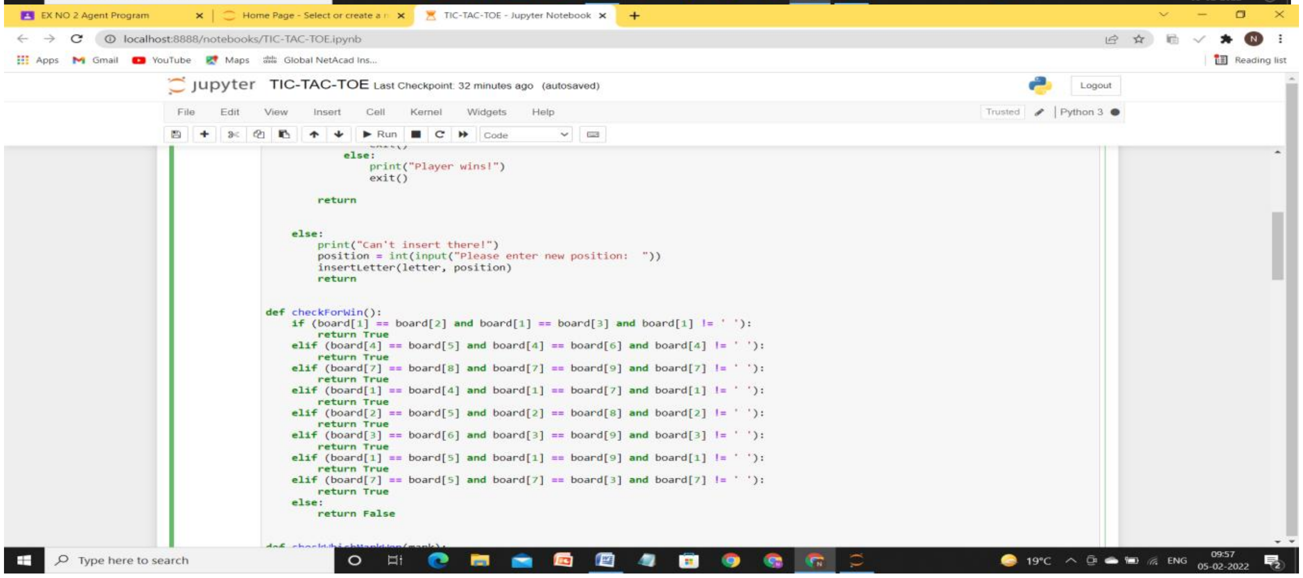
```
    print(board[4] + '|' + board[5] + '|' + board[6])

    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 = minimax(board, 0, False)
            board[key] = ' '
            if (score > bestScore):
                bestScore = score
                bestMove = key

    insertLetter(bot, bestMove)
    return


def minimax(board, depth, isMaximizing):   #agent function  #minmax algorithm
    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 = minimax(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 = minimax(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()
```



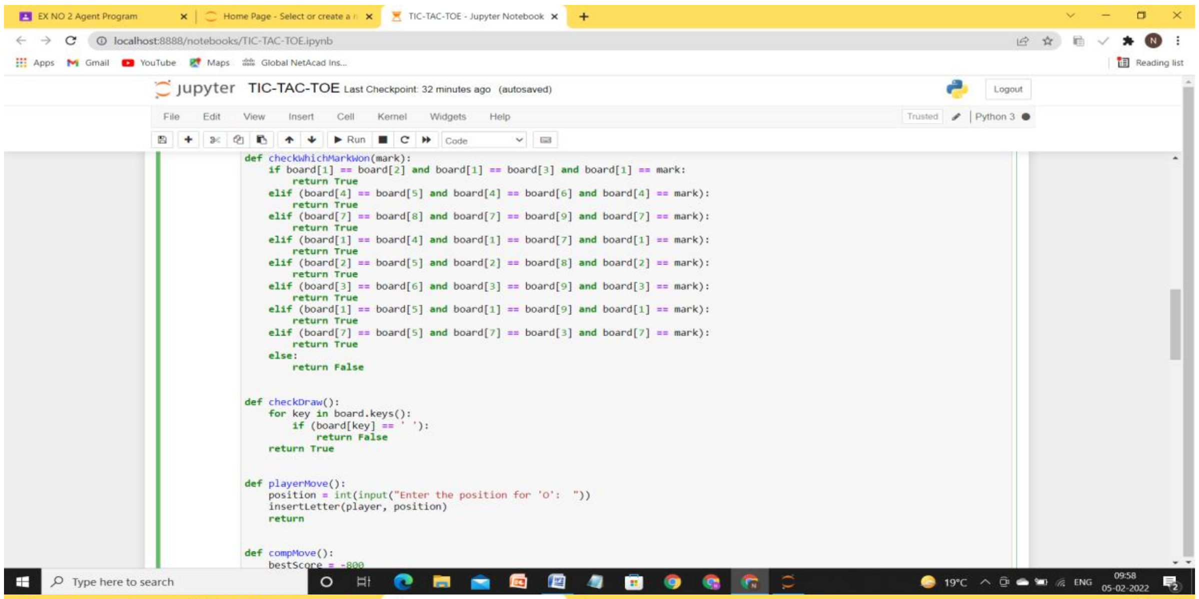
TIC-TAC-TOE Last Checkpoint: 32 minutes ago (autosaved)

```
In [*]: def printBoard(board): #Environment
            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
```

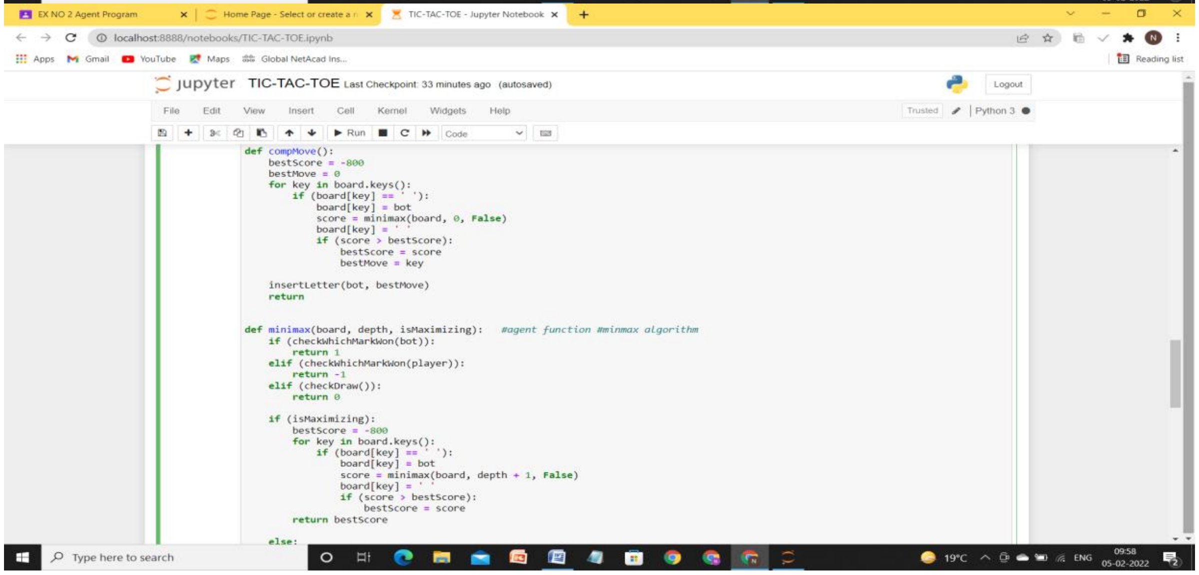


TIC-TAC-TOE Last Checkpoint: 32 minutes ago (autosaved)

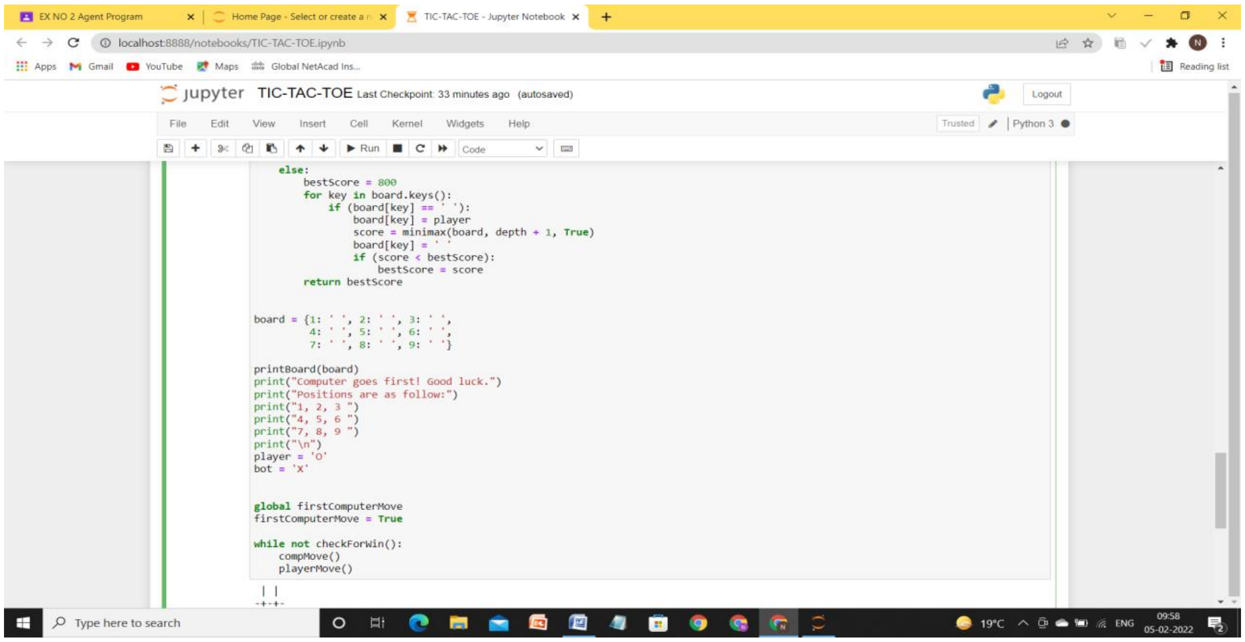
```
                    else:
                        print("Player wins!")
                        exit()

                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
```


jupyter TIC-TAC-TOE Last Checkpoint: 32 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code

```
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
```


jupyter TIC-TAC-TOE Last Checkpoint: 33 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code

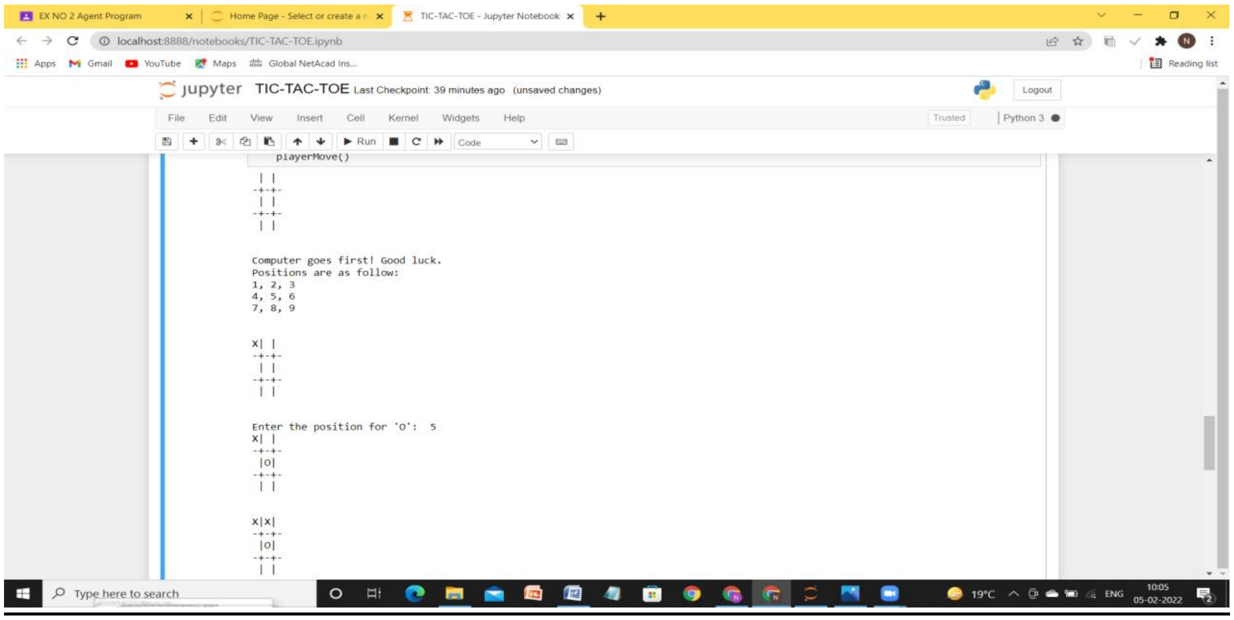
```
def compMove():
    bestScore = -800
    bestMove = 0
    for key in board.keys():
        if (board[key] == ' '):
            board[key] = bot
            score = minimax(board, 0, False)
            board[key] = ' '
            if (score > bestScore):
                bestScore = score
                bestMove = key

    insertLetter(bot, bestMove)
    return


def minimax(board, depth, isMaximizing):   #agent function #minmax algorithm
    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 = minimax(board, depth + 1, False)
                board[key] = ' '
                if (score > bestScore):
                    bestScore = score
        return bestScore

    else:
```



```
    else:
        bestScore = 800
        for key in board.keys():
            if (board[key] == ' '):
                board[key] = player
                score = minimax(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()
```

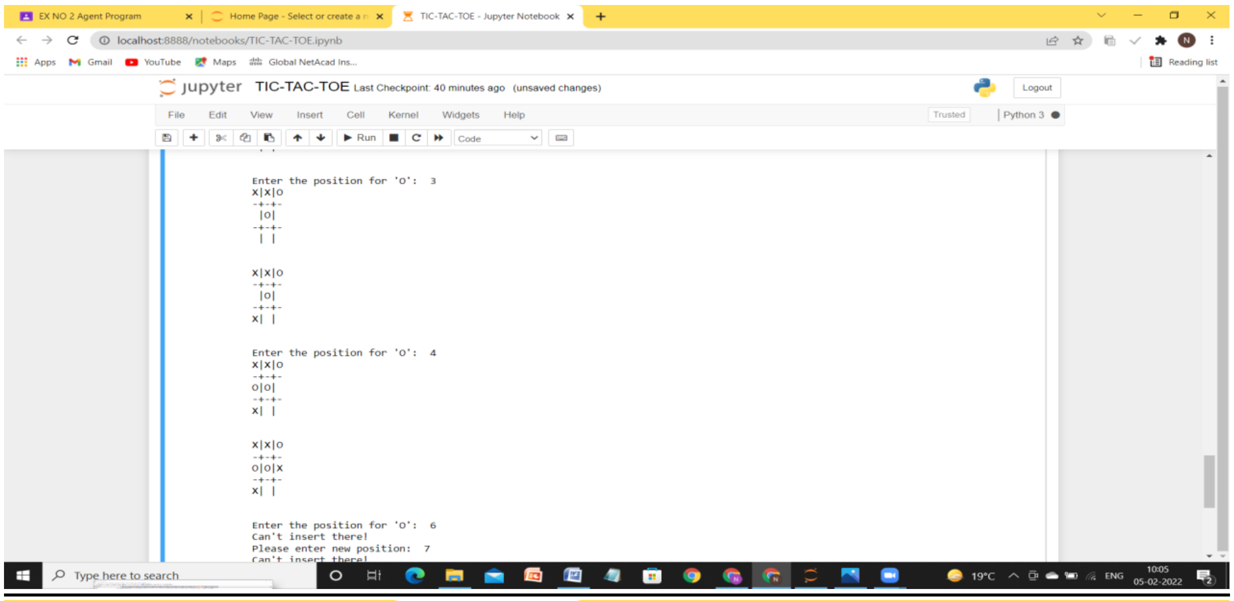
```
 | |
-+-+-
```

*Output:

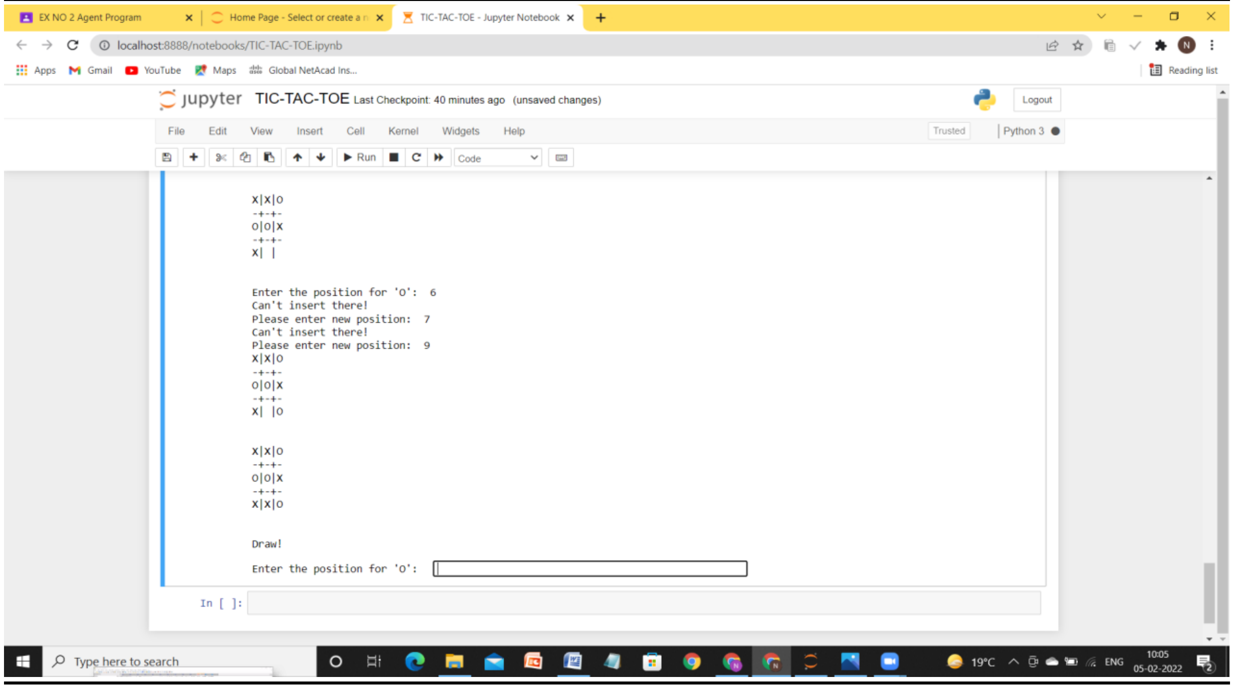


```
    playerMove()
```

```
 | |
-+-+-
 | |
-+-+-
 | |


Computer goes first! Good luck.
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9


X| |
-+-+-
 | |
-+-+-
 | |


Enter the position for 'O':  5
X| |
-+-+-
 |O|
-+-+-
 | |


X|X|
-+-+-
 |O|
-+-+-
 | |
```



```
Enter the position for 'O':  3
X|X|O
-+-+-
 |O|
-+-+-
 | |


X|X|O
-+-+-
 |O|
-+-+-
X| |


Enter the position for 'O':  4
X|X|O
-+-+-
O|O|
-+-+-
X| |


X|X|O
-+-+-
O|O|X
-+-+-
X| |


Enter the position for 'O':  6
Can't insert there!
Please enter new position:  7
Can't insert there!
```



```
X|X|O
-+-+-
O|O|X
-+-+-
X| |


Enter the position for 'O':  6
Can't insert there!
Please enter new position:  7
Can't insert there!
Please enter new position:  9
X|X|O
-+-+-
O|O|X
-+-+-
X| |O


X|X|O
-+-+-
O|O|X
-+-+-
X|X|O


Draw!

Enter the position for 'O':
```

***Result:** The problem statement is satisfied, i.e., the goal is reached.