TICTACTOE

1. Code I made:

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Definitions of the Game Board Setup

x\_mark = " X "

O\_mark = " O "

blank = " "

row0 = [blank, blank, blank]

row1 = [blank, blank, blank]

row2 = [blank, blank, blank]

gameBoard = [row0, row1, row2]

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Funtion to Print The TicTacToe Game Board

def printBoard():

rows = 0

while (rows < 3):

print(gameBoard[rows])

rows = rows + 1

print (" ")

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Funtion to Add a Players Move to the Game Board

def addMove(mark, row, col):

gameBoard[row][col] = mark

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Main Program Code is Below

#Winning Combinations

if((gameBoard[0][0] =="x\_mark") and (gameBoard[0][1] =="x\_mark") and (gameBoard[0][2] =="x\_mark"),

(gameBoard[1][0] =="x\_mark") and (gameBoard[1][1] =="x\_mark") and (gameBoard[1][2] =="x\_mark"),

(gameBoard[2][0] =="x\_mark") and (gameBoard[2][1] =="x\_mark") and (gameBoard[2][2] =="x\_mark"),

(gameBoard[0][0] =="x\_mark") and (gameBoard[1][1] =="x\_mark") and (gameBoard[2][2] =="x\_mark"),

(gameBoard[2][0] =="x\_mark") and (gameBoard[1][1] =="x\_mark") and (gameBoard[0][2] =="x\_mark"),

(gameBoard[0][0] =="x\_mark") and (gameBoard[1][0] =="x\_mark") and (gameBoard[2][0] =="x\_mark"),

(gameBoard[0][1] =="x\_mark") and (gameBoard[1][1] =="x\_mark") and (gameBoard[2][1] =="x\_mark"),

(gameBoard[0][2] =="x\_mark") and (gameBoard[1][2] =="x\_mark") and (gameBoard[2][2] =="x\_mark")):

winningCombinations = True

if((gameBoard[0][0] =="O\_mark") and (gameBoard[0][1] =="O\_mark") and (gameBoard[0][2] =="O\_mark"),

(gameBoard[1][0] =="O\_mark") and (gameBoard[1][1] =="O\_mark") and (gameBoard[1][2] =="O\_mark"),

(gameBoard[2][0] =="O\_mark") and (gameBoard[2][1] =="O\_mark") and (gameBoard[2][2] =="O\_mark"),

(gameBoard[0][0] =="O\_mark") and (gameBoard[1][1] =="O\_mark") and (gameBoard[2][2] =="O\_mark"),

(gameBoard[2][0] =="O\_mark") and (gameBoard[1][1] =="O\_mark") and (gameBoard[0][2] =="O\_mark"),

(gameBoard[0][0] =="O\_mark") and (gameBoard[1][0] =="O\_mark") and (gameBoard[2][0] =="O\_mark"),

(gameBoard[0][1] =="O\_mark") and (gameBoard[1][1] =="O\_mark") and (gameBoard[2][1] =="O\_mark"),

(gameBoard[0][2] =="O\_mark") and (gameBoard[1][2] =="O\_mark") and (gameBoard[2][2] =="O\_mark")):

winningCombinations = True

# print the starting board

printBoard()

rowMove = int(input("Player x make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player x make a move: row = "))

colMove = int(input("Player x make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player x make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player O make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player O make a move: row = "))

colMove = int(input("Player O make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player O make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player x make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player x make a move: row = "))

colMove = int(input("Player x make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player x make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player O make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player O make a move: row = "))

colMove = int(input("Player O make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player O make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player x make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player x make a move: row = "))

colMove = int(input("Player x make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player x make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player O make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player O make a move: row = "))

colMove = int(input("Player O make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player O make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player x make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player x make a move: row = "))

colMove = int(input("Player x make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player x make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player O make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player O make a move: row = "))

colMove = int(input("Player O make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player O make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

rowMove = int(input("Player x make a move: row = "))

if (rowMove > 2):

print("Bad row number, Try again...")

rowMove = int(input("Player x make a move: row = "))

colMove = int(input("Player x make a move: col = "))

if (colMove > 2):

print("Bad col number, Try again...")

colMove = int(input("Player x make a move: col = "))

addMove(O\_mark, rowMove, colMove)

printBoard()

# END OF PROGRAM

1. Code I found online :

|  |
| --- |
| import sys |
|  | import random |
|  |  |
|  | person = input('Enter your name: ') |
|  | print('Hello', person) |
|  |  |
|  | moveCounter = 0 |
|  | matchOver = False |
|  |  |
|  | userMoves = [] |
|  | myMoves = [] |
|  |  |
|  | field = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] |
|  |  |
|  | winnerPos = [["1", "2", "3"], ["1", "4", "7"], |
|  | ["1", "5", "9"], ["4", "5", "6"], |
|  | ["7", "8", "9"], ["2", "5", "8"], |
|  | ["3", "6", "9"], ["3", "5", "7"]] |
|  |  |
|  | firstAdvantage = ["5", "1", "3", "7", "9"] |
|  |  |
|  | secondAdvantage = [["5", "1"], ["5", "3"], ["5", "7"], |
|  | ["5", "9"], ["1", "3"], ["3", "9"], |
|  | ["7", "9"], ["1", "7"]] |
|  |  |
|  | thirdAdvantage = [["1", "3", "5"], ["3", "5", "9"], ["7", "5", "9"], |
|  | ["1", "5", "7"], ["1", "3", "9"], ["1", "3", "7"], |
|  | ["7", "9", "3"], ["2", "4", "1"], ["2", "3", "6"], |
|  | ["6", "9", "8"], ["4", "7", "8"]] |
|  |  |
|  | moves = ['1', '2', '3', '4', '5', '6', '7', '8', '9'] |
|  |  |
|  |  |
|  | "If player gets three in a row, player wins" |
|  | def checkWin(pos1, pos2, pos3, player): |
|  | if ((pos1 == player) and (pos2 == player) and (pos3 == player)): |
|  | playerWins = True |
|  | catsGame = False |
|  | else: |
|  | playerWins = False |
|  | return playerWins |
|  |  |
|  |  |
|  | "Check all possible win scenarios for a player." |
|  | def checkWinner(player): |
|  | for i in range(3): |
|  | win = checkWin(field[i][0], field[i][1], field[i][2], player) |
|  | if win: |
|  | if player == 'o': |
|  | print("I win!") |
|  | else: |
|  | print ("You win!") |
|  | catsGame = False |
|  | sys.exit(0) |
|  | for i in range(3): |
|  | win = checkWin(field[0][i], field[1][i], field[2][i], player) |
|  | if win: |
|  | if player == 'o': |
|  | print("I win!") |
|  | else: |
|  | print ("You win!") |
|  | catsGame = False |
|  | sys.exit(0) |
|  | win = checkWin(field[0][0], field[1][1], field[2][2], player) |
|  | if win: |
|  | if player == 'o': |
|  | print("I win!") |
|  | else: |
|  | print ("You win!") |
|  | catsGame = False |
|  | sys.exit(0) |
|  |  |
|  | win = checkWin(field[0][2], field[1][1], field[2][0], player) |
|  | if win: |
|  | if player == 'o': |
|  | print("I win!") |
|  | else: |
|  | print ("You win!") |
|  | catsGame = False |
|  | sys.exit(0) |
|  |  |
|  |  |
|  | "Print the board." |
|  | def printField(): |
|  | counter = 0 |
|  |  |
|  | for i in field[:2]: |
|  | for j in i[:2]: |
|  | print(" ", j, " | ", end="") |
|  | print(" ", field[counter][2]) |
|  | counter += 1 |
|  | print ("------------------") |
|  | print (" ", field[2][0], " | ", field[2][1], " | ", field[2][2]) |
|  |  |
|  |  |
|  | counter = 0 |
|  |  |
|  |  |
|  | "Defines coordinates of each numbered position on board." |
|  | def switch(x): |
|  | return { |
|  | '1': [0, 0], |
|  | '2': [0, 1], |
|  | '3': [0, 2], |
|  | '4': [1, 0], |
|  | '5': [1, 1], |
|  | '6': [1, 2], |
|  | '7': [2, 0], |
|  | '8': [2, 1], |
|  | '9': [2, 2] |
|  | }[x] |
|  |  |
|  |  |
|  | "Fills a position with players move." |
|  | def populate(x, side): |
|  | nums = switch(x) |
|  | field[nums[0]][nums[1]] = side |
|  |  |
|  |  |
|  | "Anticipate win or advantage for each player." |
|  | def think(): |
|  | if moveCounter == 0: |
|  | answer = "5" |
|  | elif moveCounter == 1: |
|  | if userMoves[-1] == "5": |
|  | answer = "1" |
|  | else: |
|  | answer = "5" |
|  | else: |
|  | answer = anticipateWin() |
|  | if answer == "0": |
|  | answer = anticipateUserWin() |
|  | if answer == "0": |
|  | answer = anticipateAdvantage() |
|  | if answer == "0": |
|  | answer = anticipateUserAdvantage() |
|  | if answer == "0": |
|  | answer = random.choice(moves) |
|  | myMoves.append(answer) |
|  | return answer |
|  |  |
|  |  |
|  | "Checks win and advantage scenario lists to inform next move." |
|  | def anticipate(posList, whoMoves): |
|  | answer = "0" |
|  | for lis in posList: |
|  | commonEl = set(whoMoves) & set(lis) |
|  | if len(commonEl) > 1: |
|  | for el in lis: |
|  | if el not in commonEl: |
|  | if el in moves: |
|  | answer = el |
|  | break |
|  | break |
|  | return answer |
|  |  |
|  |  |
|  | "Check if next move can win game." |
|  | def anticipateWin(): |
|  | answer = anticipate(winnerPos, myMoves) |
|  | return answer |
|  |  |
|  |  |
|  | "Check if user's next move can win game." |
|  | def anticipateUserWin(): |
|  | answer = anticipate(winnerPos, userMoves) |
|  | return answer |
|  |  |
|  |  |
|  | "Check if user's next move will give user advantage." |
|  | def anticipateUserAdvantage(): |
|  | if len(userMoves) < 2: |
|  | answer = anticipate(secondAdvantage, userMoves) |
|  | else: |
|  | answer = anticipate(thirdAdvantage, userMoves) |
|  | return answer |
|  |  |
|  |  |
|  | "Check if next move can give advantage." |
|  | def anticipateAdvantage(): |
|  | answer = "0" |
|  | if len(myMoves) < 2: |
|  | answer = anticipate(secondAdvantage, myMoves) |
|  | else: |
|  | answer = anticipate(thirdAdvantage, myMoves) |
|  | return answer |
|  |  |
|  |  |
|  | printField() |
|  |  |
|  |  |
|  | "User input decides who moves first." |
|  | moveFirst = input("Who moves first... you or me?: ") |
|  |  |
|  | if (moveFirst == "you"): |
|  | print ("My turn: ") |
|  |  |
|  | if (len(moves) > 0): |
|  | myMove = think() |
|  |  |
|  | if (myMove in moves): |
|  | moves.remove(myMove) |
|  | else: |
|  | print ("not in list") |
|  |  |
|  | populate(myMove, 'o') |
|  |  |
|  | printField() |
|  |  |
|  | moveCounter += 1 |
|  |  |
|  | catsGame = True |
|  |  |
|  | "Main loop of game lasts less than 9 moves." |
|  | while (moveCounter < 9): |
|  |  |
|  | move = input("Choose your move: ") |
|  |  |
|  | userMoves.append(move) |
|  |  |
|  | "Remove move position from list of possible moves." |
|  | if (len(moves) >= 0): |
|  | moves.remove(move) |
|  | else: |
|  | break |
|  |  |
|  | populate(move, 'x') |
|  |  |
|  | printField() |
|  |  |
|  | moveCounter += 1 |
|  |  |
|  | checkWinner('x') |
|  |  |
|  | print ("My turn: ") |
|  |  |
|  | if len(moves) > 0: |
|  | myMove = think() |
|  |  |
|  | if (myMove in moves): |
|  | moves.remove(myMove) |
|  | else: |
|  | print ("not in list") |
|  | break |
|  |  |
|  | populate(myMove, 'o') |
|  |  |
|  | printField() |
|  |  |
|  | moveCounter += 1 |
|  |  |
|  | checkWinner('o') |
|  |  |
|  | if catsGame: |
|  | print ("cats game...") |
|  | else: |
|  | print("...") |