**TIC TAC TOE**

import random

def generateSquare(n):

magicSquare = [[0 for x in range(n)] for y in range(n)]

i, j = 0,1

num = 1

while num <= (n \* n):

magicSquare[i][j] = num

next\_i, next\_j = (i - 1) % n, (j - 1) % n

if magicSquare[next\_i][next\_j] != 0:

next\_i, next\_j = (i + 1) % n, j

i, j = next\_i, next\_j

num += 1

return magicSquare

def transposeFirstAndThirdColumn(magicSquare):

for j in range(len(magicSquare)):

magicSquare[j][0], magicSquare[j][2] = magicSquare[j][2], magicSquare[j][0]

return magicSquare

def transposeFirstAndThirdRow(magicSquare):

for j in range(len(magicSquare)):

magicSquare[0][j], magicSquare[2][j] = magicSquare[2][j], magicSquare[0][j]

return magicSquare

def rotateBy180Degrees(magicSquare):

rotatedSquare = [row[::-1] for row in magicSquare[::-1]]

return rotatedSquare

def printMagicSquareWithMoves(magicSquare, player\_moves, computer\_moves, player\_symbol, computer\_symbol):

print("Magic Square:")

for row in magicSquare:

print(' | '.join(

f'{player\_symbol}' if num in player\_moves else

f'{computer\_symbol}' if num in computer\_moves else

f'{num:1}'

for num in row

))

print('-' \* 9)

def checkWin(magicSquare, n, moves):

magic\_constant = n \* (n \* n + 1) // 2

for i in range(n):

if sum(magicSquare[i][j] for j in range(n) if magicSquare[i][j] in moves) == magic\_constant:

return True

if sum(magicSquare[j][i] for j in range(n) if magicSquare[j][i] in moves) == magic\_constant:

return True

if sum(magicSquare[i][i] for i in range(n) if magicSquare[i][i] in moves) == magic\_constant:

return True

if sum(magicSquare[i][n - i - 1] for i in range(n) if magicSquare[i][n - i - 1] in moves) == magic\_constant:

return True

return False

def findWinningMove(magicSquare, n, moves, available\_numbers):

for number in available\_numbers:

temp\_moves = moves + [number]

if checkWin(magicSquare, n, temp\_moves):

return number

return None

def tossForFirstPlayer():

ch = random.choice([0, 1])

print('Toss result: ', ch)

return ch

def playGame(magicSquare, n):

available\_numbers = [num for row in magicSquare for num in row]

player\_moves = []

computer\_moves = []

printMagicSquareWithMoves(magicSquare, player\_moves, computer\_moves, " ", " ")

while True:

human\_choice = input('Select a number "0" or "1" for the toss: ').strip()

if human\_choice in ['0', '1']:

human\_choice = int(human\_choice)

break

else:

print("Invalid choice. Please select either 0 or 1.")

toss\_result = tossForFirstPlayer()

if human\_choice == toss\_result:

print("Human won the toss!!")

player\_symbol = 'X'

computer\_symbol = 'O'

current\_player = 'X'

else:

print("Computer won the toss!!")

player\_symbol = 'O'

computer\_symbol = 'X'

current\_player = 'X'

for turn in range(n \* n):

if current\_player == 'X':

if computer\_symbol == 'X':

winning\_move = findWinningMove(magicSquare, n, computer\_moves, available\_numbers)

if winning\_move is not None:

computer\_move = winning\_move

else:

blocking\_move = findWinningMove(magicSquare, n, player\_moves, available\_numbers)

if blocking\_move is not None:

computer\_move = blocking\_move

else:

computer\_move = random.choice(available\_numbers)

print(f"Computer's choice: {computer\_move}")

computer\_moves.append(computer\_move)

available\_numbers.remove(computer\_move)

else:

while True:

try:

player\_move = int(input(f"Select number from the board: "))

if player\_move not in available\_numbers:

raise ValueError("Invalid choice. Choose from the available numbers.")

break

except ValueError:

print("Invalid choice. Please enter an integer value.")

player\_moves.append(player\_move)

available\_numbers.remove(player\_move)

current\_player = 'O'

else:

if player\_symbol == 'O':

while True:

try:

player\_move = int(input(f"Select number from the board: "))

if player\_move not in available\_numbers:

raise ValueError("Invalid choice. Choose from the available numbers.")

break

except ValueError:

print("Invalid choice. Please enter an integer value.")

player\_moves.append(player\_move)

available\_numbers.remove(player\_move)

else:

winning\_move = findWinningMove(magicSquare, n, computer\_moves, available\_numbers)

if winning\_move is not None:

computer\_move = winning\_move

else:

blocking\_move = findWinningMove(magicSquare, n, player\_moves, available\_numbers)

if blocking\_move is not None:

computer\_move = blocking\_move

else:

computer\_move = random.choice(available\_numbers)

print(f"Computer's choice: {computer\_move}")

computer\_moves.append(computer\_move)

available\_numbers.remove(computer\_move)

current\_player = 'X'

printMagicSquareWithMoves(magicSquare, player\_moves, computer\_moves, player\_symbol, computer\_symbol)

if checkWin(magicSquare, n, computer\_moves):

print("Computer wins!")

return

elif checkWin(magicSquare, n, player\_moves):

print("Player wins!")

return

print("It's a draw!")

def main():

while True:

magicSquare = generateSquare(3)

magicSquare = transposeFirstAndThirdRow(magicSquare)

magicSquare = transposeFirstAndThirdColumn(magicSquare)

playGame(magicSquare, 3)

while True:

play\_again = input("Would you like to play again? (yes/no): ").lower()

if play\_again in ['yes', 'no']:

break

else:

print("Invalid input. Please enter 'yes' or 'no'.")

if play\_again == 'no':

print("Thanks for playing!")

return

main()