

Algorithm to check if move is valid

Variables -

Algorithm -

- Step-1 Initialize 3x3 matrix (board)
- Step-2 Check if any winning player exists or check if the board is filled completely.
And print draw or that the player has won depending on the condition.
Convert matrix into a string.
- Step-3 Let the player play and input the choice in a row & column of their choice.
- ~~Step-4 Convert the matrix to a string of 24 characters~~
- Step-4 Check if the ~~player~~ computer can win or if the player can win using the win state.
- Win state can be 8 different conditions based on the rules where the algorithm can check if the win state is achieved.
- Step-5 If win state has not been achieved then find if there is a draw state where board is filled and none of the states match.

0	1	0
1	1	1
1	1	1

if (i) equal to 0 -> 0
["X", "0"] = invalid
"0" = equal - invalid

if (i) equal to 1 -> 1

if (i) equal to 2 -> 2

if (i) equal to 3 -> 3

if (i) equal to 4 -> 4

if (i) equal to 5 -> 5

if (i) equal to 6 -> 6

if (i) equal to 7 -> 7

if (i) equal to 8 -> 8

if (i) equal to 9 -> 9

if (i) equal to 10 -> 10

if (i) equal to 11 -> 11

if (i) equal to 12 -> 12

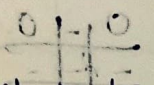
if (i) equal to 13 -> 13

if (i) equal to 14 -> 14

Step - 6 If none of the states match then repeat the working taking moves after analysing the string to get closer to winning.

Step - 7 Go back to step 2 and repeat the entire process.

Code for the following algorithm -



import random

tic = ["O", "X"] * 3 for _ in range(3)

players = ["O", "X"]

current_player = "O"

def place(i, j, player):

tic[i][j] = player

def listify():

res = []

for i in range(3):

res.append(" ".join(tic[i]))

for i in range(3):

temp = ""

for j in range(3):

temp += tic[i][j]

res.append(temp)

res.append(tic[0][2] + tic[1][1] + tic[2][0])

return res

def check(list):

win = 0

for i in range(3):

for i in li:

count = max(count, i.count("o"))

if i.count("o") == 3:

return 0

if i.count("X") == 3

return 1

if count == -1:

return 1

return 4.

def display():

for i in range(3):

for j in range(3):

print(tic[i][j], end=" ")

print("\n")

print("{} tip - (":(s-o) rounded too with rest") before, tip) part = j

: " - " == (j[i] set

(length - 1, i, i) set

"X" = length - 1

def computer-move():

for i in range(3):

for j in range(3):

if tic[i][j] == "-":

place(i, j, "X")

if check(history()) == 1:

return

place(i, j, "-")

def main():

global current-player

while True:

display ()

li = listify ()

flag = check (li)

if flag == 0:

print ("Player 0 wins!")

break

elif flag == 1:

print ("Player X (computer) wins!")

break

elif flag == -1:

print ("It's a draw!")

break

if current_player == "O":

print ("Player O's turn")

i, j = map(int, input("Enter row and column (0-2): ").split())

if li[i][j] == "-":

place(i, j, current_player)

current_player = "X"

else:

print ("Invalid move, try again")

else:

print ("Computer's turn")

computer_move()

current_player = "O"

OUTPUT -

2-0A1

Player O's turn

Enter row and column (0-2): 0 2

-- 0
--
--

Computer's turn

X - 0

--
--

1-442

Player O's turn

Enter row and column (0-2): 2 0

X - 0

-- 0

--

Computer's turn

X - 0

-- 0

-- X

(0, 2) - this is

(2, 0) - this is

(1, 1) - this is

(0, 0) - this is

Player O's turn

Enter row and column : 1 0

X - 0

0 - X

-- X

Computer's turn

X - 0

(("state" X 0 move) turn) to = 0

(("state" 0 X move) turn) to = 0

(1, 1) - this is

Computer wins!

Shree

2-0A1

computer wins!

til - move in i ref

(("state" == state, 1) f

(check)