1

Tic-tac-toe ~~-P~~Program.

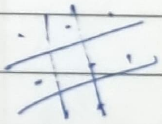
1. Algorithm

→ Create board of 3x3 grid where Emtty Spaces are marked as ~~0~~-1
→ Take input 1 (or) 0
→ Randomly Select one of the Empty Positions and mark it 1 (or) 0
if the Position is available, Place the Player's mark (1) on that Spot

2. To checkk for win.

→ We need to Evaluate whether the Player (or) Combuter marks (1 (or) 0) forms a Combination in any row (or) column (or) diagonal.

~~row-win(board, Player):~~
rows (0, 1, 2)(3, 4, 5)(6, 7, 8)
Columns (0, 3, 6) (1, 4, 7) (2, 5, 8)
diagonals (0, 4, 8) (2, 4, 6)
if any of these conditions are Satisfied that Player will Win.
(Vertical, Horizontal, diagonal) - 8 winning cond chancy



Combuter move
~~To~~ combuter select ~~r~~andom Embty Spot and Place it mark. (0)

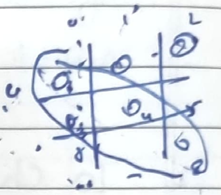
→ Check for tie
a tie Occurs if board is Completly filled, and no one has ~~a~~ won. if there are no empty Spaces and no winning Combination, we can

declare tie.

→ End game
  if there is a winner, we announce who won
  If the board is full & no one wins,
     announce tie.

winning condition.

0- user i/p



```
if all values of row is 0
     return 1
if all Values of col is 0
     return 1
 if [0][0]. [1][1] & [2][2]
     return 1

→diag   if [0][2] [1][1] & [2][0]
          return 1
     else return 0
```

row[0][j]

row[0][i] ~~[~~

```
if the condition is true (1)
     Print user wins

generate random()
row/col = random choice( empty cell present)

generate random row mod 3
generate random col mod 3
  if mat [i][j] = -1
  return 1
else
generate random ( )
```

```
row , Col = random .choice (Embty cell )
 when   Combwing Print comb won
   if  all cell = billed
          Declare tie
```

```
Imbort random
def initialize_board() :
return [[' ' for_ in range (3)] for_in range(3)]
def display_board (board):
    for  row in board :
      Print  ('|'.join(row))
      Print ('-' * 5)
def  check_winner (board);
     for  row in  board :
     if  row[0] == row[1]== row[2]!= ' ';
          return row[0]
  for col   in range (3 ):
     if board [0][col] == board [1][col] ==
  ~~for~~ to  board [2][col] != ' ';
         return board [0][col]
    if board [0][0] == board[1][1] == board[2][2
          != ' ';
      return board [0][0]
   if  board  [0][0] == board[1][1] == board [2][2]!=
         return   board [0][0]
    if board [0][2] == board  [1][1] == board
               [2][0] != ' ';
        return   board [0][2]
      return None
  def  available _moves (board);
       return [(i, j) for i  in range (3) for
    j in range (3) if board [i][j]==' ']
  def Check_two_in_a_row (board, Player):
    for row in range(3):
  if board [row] Count[Player)==2 and
        board[row]. count(' ')==1:
      return row, board [row].index(' ')

 # check diagonals
  if [board [i][i] for i in range (3)]
  Count (Player )==2:
    Empty_index = [i for i in rang(3) if
        board [i][i]==' ' ]
      if Empty_index:
    return Empty_index[0], Empty_index[0]

   def make_move (board, Player, move)
   board [move[0]][move[1] ]=Player

   def      Computer_move (board):

  move = Check_two_in_a_row (board, 0)
   if move:
      make_move (board):
      return

  def User_move(board)

   While True:
     try :
      row = int(input ("Entry row (0-2):"))
        Col = Int(input ("Enter column (0-2)):")
        if     board[row][col]==' ':
```

```
make_move (board , 'x' , (row, col))
    return
else :
  Print ("That Spot is already taken
            Try again )

def Play_game ( ):

  board = initialize -board ( )
    Players = ['x' ; o' ]
    current Player = 0
  for _ in range (9 ):
    display -board (board)
   if current ~~board~~ Player = = 0:
        user_move (board)
    else :
        Computer = check _winner (board)
     if winner:
        display _board (board )
       Print (f "Player {winner } wins! ")
      return
     Current_Player          = 1 - Current-Player

   display _board ( board )
    Print (" it's a draw" )
    Play_game ( )
```

Output   Enter row   (0-2): 0
          Enter col   (0-2): 0

```
 X |   |
---+---+---
   |   |
---+---+---
   |   |
Enter row (0-2) = 0
enter col (0-2) : 1

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

Enter row (0-2) = 0
Enter col (0-2) = 2

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

Enter row (0-2): 1
Enter col (0-2): 10

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

Enter row (0-2) : 1
Enter col (0-2): 1

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

Enter row (0-2 ): 1
Enter col (0-2 ): 2

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

Enter row (0-2): 2
Ent col (0-2): 0
```

```
X | O | X
O | X | O
X |   |

Enter row (0-2): 2
Enter col (0-2): 1

X | O | X
O | X | O
X | O |

Enter row (0-2): 2
Enter col (0-2): 2

X | O | X
O | X | O
X | O | X

It's a draw!
```