27/09/24 Lab-1

Algorithm

1) At first import a random to generate a random number

2) And then define to print board function using '–' to generate 3×3 matrix

3) Now check the rows, columns and diagonals

4) And check the empty spots to the next move

5) To check the winner we will check the rows, colums, diagonals. And it is range from 3

```
if board[i][0] == board[i][1] == board[i][2] != " ":
        return[i][0]
if board[0][i] == board[1][i] == board[2][i] != " ":
        return[0][i]
if board[0][0] == board[1][1] == board[2][2] != " ":
        return[0][0]
if board[0][2] == board[1][1] == board[2][0] != " ":
        return[0][2]
```

6) We have check the board is full or not, if it full it will show board is full there is no space

7) And computer will see for empty cell it will randomly choose a cell for next move

8) Now define function tic tac toe. it will range from 3 and current player is 'x' and a computer player is 'o'.

9) If the current player is equal to the 'x' then give input for rows and columns or else for computer player give input for rows & columns.

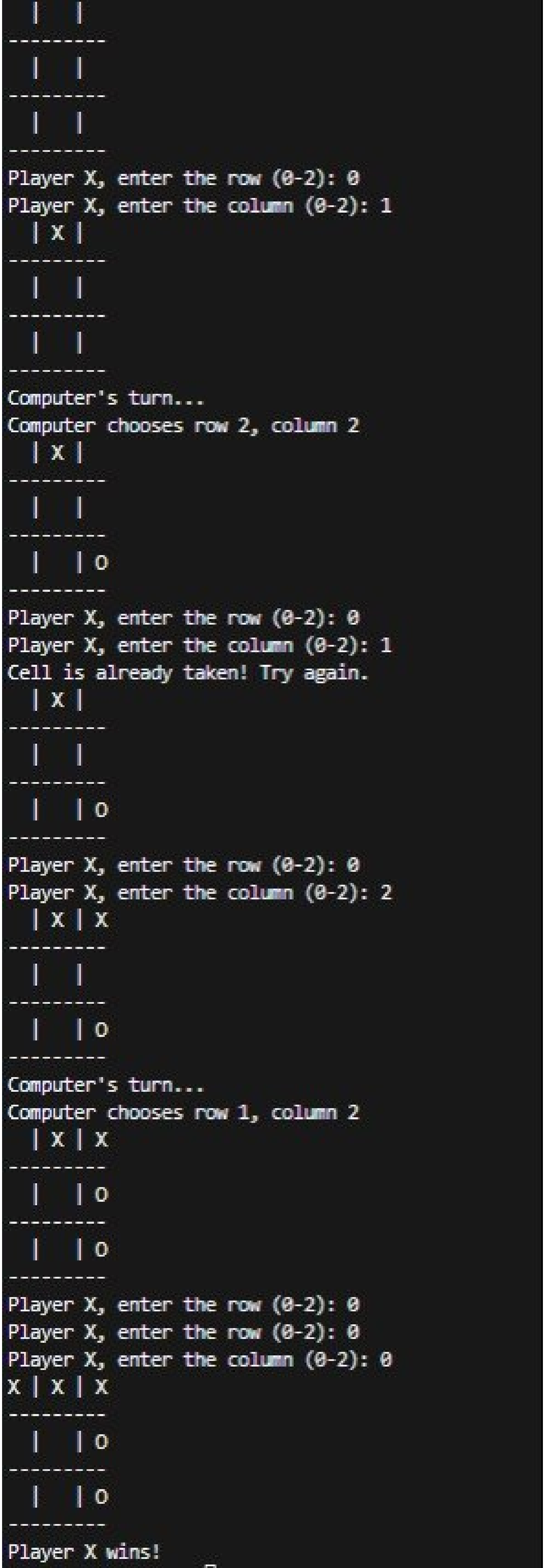
10) If they give a same row & column again it will show cell is already taken.

11) Later it will shows the who is the winner.

Implement a tic tac toe game.

```
import random.
def print_board (board):
    for row in board:
        print ("1" . join(row))
        print("—" *9)
def check-winner(board):
    for i in range(3):
        if board[i][0] == board[i][i] == board[i][2] != " ":
            return board[i][0]
        if board [0][i] == board[i][i] == board [2][i] != " ":
            return board [0][i]
    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 is_full (board):
    return all(cell != " " for row in board for cell in
                        row)
def get-computer-move (board):
    empty-cells= [(i, j) for i in range(3) for j in
                    range(3) if board[i][j] == " "]
    return random.choice (empty-cells)
def tic_tac_toe():
    board = [[" " for _ in range(3)] for _ in range(3)]
    current_player = "x"
    computer_player= "o"
    while True:
        print_board (board)
    if current_player== " X":
        row= int(input(" Player X, enter the row (0-2): "))
        col= int (input(" Player X, enter the column(0-2): "))
    else:
        print (" computer's turn...")
        row, col = get_computer_move(board)
        print(f "computer chooses row {row}, column {col}")
    if board [row][col]== " ":
        board [row][col]= current_player
    else:
        print(cell is already taken! Try again.")
        continue.
    winner= check_winner(board)
    if winner:
        print_board (board)
        print(f "player {winner} wins!"}
        break.
    if is_full (board):
        print_board (board)
        print ("It's a tie!")
        break
    current_player= computer_player if current_player==
                        "X" else "X"
if _name_ == "_main_":
    tic_tac_toe()
```

24/9



```
  |   |
---------
  |   |
---------
  |   |
---------
Player X, enter the row (0-2): 0
Player X, enter the column (0-2): 1
  | X |
---------
  |   |
---------
  |   |
---------
Computer's turn...
Computer chooses row 2, column 2
  | X |
---------
  |   |
---------
  |   | O
---------
Player X, enter the row (0-2): 0
Player X, enter the column (0-2): 1
Cell is already taken! Try again.
  | X |
---------
  |   |
---------
  |   | O
---------
Player X, enter the row (0-2): 0
Player X, enter the column (0-2): 2
  | X | X
---------
  |   |
---------
  |   | O
---------
Computer's turn...
Computer chooses row 1, column 2
  | X | X
---------
  |   | O
---------
  |   | O
---------
Player X, enter the row (0-2): 0
Player X, enter the row (0-2): 0
Player X, enter the column (0-2): 0
X | X | X
---------
  |   | O
---------
  |   | O
---------
Player X wins!
```