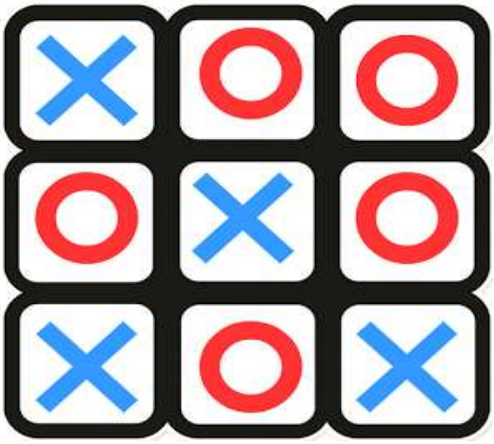
PROJECT

AI BASED GAME

Tic Tac Toe



INT-404

Submitted By :-

Sai Sarthak Mohapatra

Roll no- 49**(Loom Video)**

Sumit kumar

Roll no- 47**(Coding)**

Akhilesh Pandey

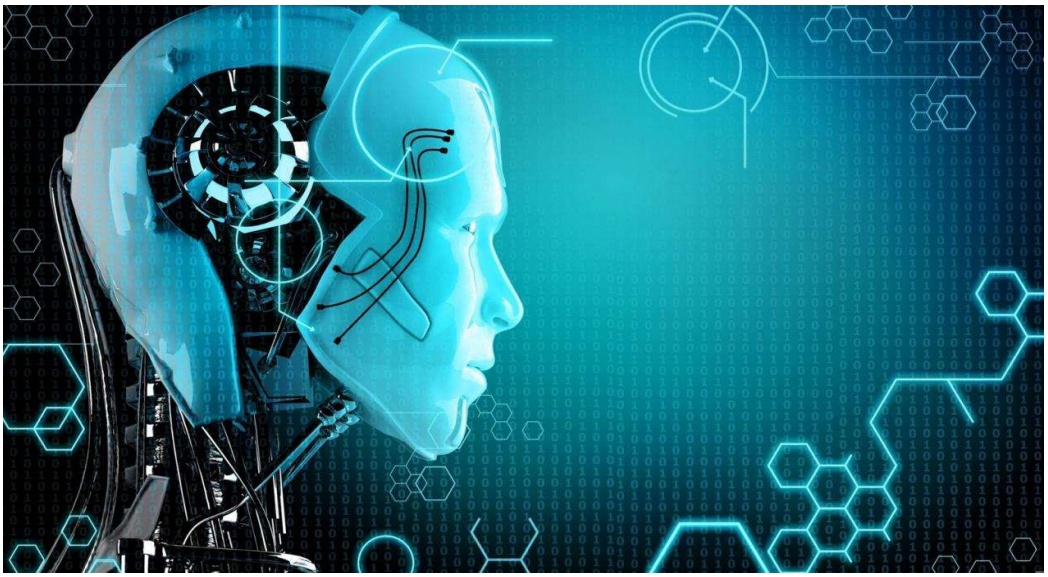
Roll no- 48**(Report)**

Submitted To :-

Ms. Jasleen Kaur

Content

What is AI??



Artificial intelligence (AI) is the ability of a computer program or a machine to think and learn. It is also a field of study which tries to make computers "smart". They work on their own without being encoded with commands.

What is mean AI Based Game?

In video **games**, artificial intelligence (**AI**) is used to generate responsive, adaptive or intelligent behaviors primarily in non-player characters (NPCs) similar to human-like intelligence.Artificial intelligence has been an integral part of video **games** since their inception in the 1950s.

.Tic Tac Toe Game

Tic-tac-toe is a simple, two-player game that, if played optimally by both players, will always result in a tie. The game is also called noughts and crosses or Xs and Os.

Tic-tac-toe is a game that is traditionally played by being drawn on paper, and it can be played on a computer or on a variety of media. Other games, such as Connect 4, are based on this classic.

History Of Tic Tac Toe Game

An early variation of the game was played in the **Roman Empire**, around the 1st century B.C. It was called **"terni lapilli,**" which means "three pebbles at a time." The game's grid markings have been found chalked all over Roman ruins. Evidence of the game was also found in ancient Egyptian ruins.

The first print reference to "noughts and crosses," the British name for the game, appeared in **1864**. The first print reference to a game called **"tick-tack-toe"** occurred in **1884** but referred to a children's game played on a slate.

.Advanced Tic Tac Toe

A relatively simple game usually played on a grid of 3-by-3 squares, tic-tac-toe is mainly enjoyed by children. Tic-tac-toe can be made significantly more complex by increasing the size of the board to 4-by-4, 5-by-5, or even up to a 20-by-20 grid.

Game-PLay

The goal of tic-tac-toe is to be the first player to get three in a row on a 3-by-3 grid or four in a row in a 4-by-4 grid.
To start, one player draws a board, creating a grid of squares, usually 3-by-3 or 4-by-4.

In a 3-by-3 grid game, the player who is playing "X" always goes first. Players alternate placing Xs and Os on the board until either player has three in a row, horizontally, vertically, or diagonally or until all squares on the grid are filled. If a player is able to draw three Xs or three Os in a row, then that player wins. If all squares are filled and neither player has made a complete row of Xs or Os, then the game is a draw.

One of the game's best strategies involves creating a "fork," which is placing your mark in such a way that you have the opportunity to win two ways on your next turn. Your opponent can only block one, thereby, you can win after that.

The gameplay is the same if you are playing on a 4-by-4 grid. The "X" player goes first. And, players alternate placing Xs and Os on the board until a row is completed horizontally, vertically, or diagonally, or all 16 squares are filled. If all 16 squares are filled and neither player has four in a row, the game is a draw.

Other Variant

Tic-tac-toe can be also be played on a 5-by-5 grid with each player trying to get five in a row.

The game can also be played on larger grids, such as 10-by-10 or even 20-by-20. For any grid of 6-by-6 or greater, it might be best to make your goal to get five in a row. This turns the basic game of tic-tac-toe into a much more complex game with similarities to the board game **Pente**, meaning **"five**" in Greek. Similarly, the goal of Pente is for a player to score five marks in a row.

Code of Tic Tac Toe game with AI in Python Programming:

```
import random
import sys

print ("Welcome to Tic Tac Toe!")
print ("The board is set up like this: ")
print ("0|1|2")
print ("3|4|5")
print ("6|7|8")
print ("Enter your first move, 0 through 8. (You are O's, the AI is X's)")

# The 'pairs' list is used to check and see if anyone has won the game.
# -------|          Verticals          |          Horizontals          |          Diagonals          |

pairs = ([0, 3, 6], [1, 4, 7], [2, 5, 8], [0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 4, 8], [2, 4, 6])

# 'Corners' is used for the AI to let it pick random corners.

corners = [0, 2, 6, 8]
board = ["", "", "", "", "", "", " ", " ", " ", " "]

# 'Turn' keeps track of whose turn it is. During the game it is either "AI" or "PLAYER".

turn = "PLAYER"

# 'Aiturn' lets the AI know how many turns it has taken.

aiturn = 0
```

```
# Just prints the board and starts the next turn
def printboard(turn, board, aiturn):
    print (board[0] + "|" + board[1] + "|" + board[2])
    print (board[3] + "|" + board[4] + "|" + board[5])
    print (board[6] + "|" + board[7] + "|" + board[8])
    print ("Turn: " + str(turn))
    if turn == 0:
        playermove(turn, board, aiturn)
    if turn == "AI":
        aiturn += 1
        aimove(turn, board, aiturn, corners)
    if turn == "PLAYER":
        playermove(turn, board, aiturn)


# Prompts the player to move
# I added some stuff that makes sure the input is legal.

def playermove(turn, board, aiturn):
    choice = input("Enter a number 0-8: ")
    if choice.isdigit() == False:
        print( "Keep it an integer, buddy.")
        playermove(turn, board, aiturn)
    if int(choice) > 8 or int(choice) < 0:
        print ("Make that a number between 0 and 8.")
        playermove(turn, board, aiturn)
    if board[int(choice)] == 'X' or board[int(choice)] == '0':
        print ("That's already taken! Try again.")
        playermove(turn, board, aiturn)
    else:
        board[int(choice)] = "0"
    turn = "AI"
    checkforwin(turn, board, aiturn)
```

```
# Makes the AI move.


def aimove(turn, board, aiturn, corners):
    alreadymoved = False
    completes = ([0, 3, 6], [1, 4, 7], [2, 5, 8], [0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 4, 8], [2, 4, 6])

    # Pretty self-explanatory, just chooses a random empty corner.

    def cornerchoice(corners, board, alreadymoved):
        goodchoices = []
        if not alreadymoved:
            for i in corners:
                if board[i] == " " or board[i] == "_":
                    goodchoices.append(i)
            board[random.choice(goodchoices)] = "X"

    # If the player didn't take the center on their first move, this makes the AI take the center.

    if aiturn == 1:
        if board[4] != "O":
            board[4] = "X"
            alreadymoved = True
        else:
            cornerchoice(corners, board, alreadymoved)
            alreadymoved = True

    # Checks to see if there are any possible ways for the game to end next turn, and takes proper action.

    else:
        for x in completes:
            # Offensive
```

```
                if board[x[0]] == "X" and board[x[1]] == "X" and board[x[2]] != "O":
                    board[x[2]] = "X"
                    alreadymoved = True
                    break
                if board[x[1]] == "X" and board[x[2]] == "X" and board[x[0]] != "O":
                    board[x[0]] = "X"
                    alreadymoved = True
                    break
                if board[x[0]] == "X" and board[x[2]] == "X" and board[x[1]] != "O":
                    board[x[1]] = "X"
                    alreadymoved = True
                    break
        # start of the list 'pairs' before it would play offensive with items later in 'pairs'.
            for x in completes:

                if alreadymoved == False:
                    # Defensive
                    if board[x[0]] == "O" and board[x[1]] == "O" and board[x[2]] != "X":
                        board[x[2]] = "X"
                        alreadymoved = True
                        break
                    if board[x[1]] == "O" and board[x[2]] == "O" and board[x[0]] != "X":
                        board[x[0]] = "X"
                        alreadymoved = True
                        break
                    if board[x[0]] == "O" and board[x[2]] == "O" and board[x[1]] != "X":
                        board[x[1]] = "X"
                        alreadymoved = True
                        break

        if not alreadymoved:
            # The 'and board[4] == "O"' part was added because of another exploit similar to the last one mentioned
above
            if aiturn == 2 and board[4] == "O":
```

```
                cornerchoice(corners, board, alreadymoved)
            else:
                sides = [1, 3, 5, 7]
                humansides = 0
                for i in sides:
                    if board[i] == "O":
                        humansides += 1
                if humansides >= 1:
                    cornerchoice(corners, board, alreadymoved)
                else:

                    goodchoices = []
                    for i in sides:
                        if board[i] == " " or board[i] == "_":
                            goodchoices.append(i)
                    if goodchoices == []:
                        cornerchoice(corners, board, alreadymoved)
                    else:
                        board[random.choice(goodchoices)] = "X"

        turn = "PLAYER"
        checkforwin(turn, board, aiturn)



def checkforwin(turn, board, aiturn):
    for x in pairs:
        zero = board[x[0]]
        one = board[x[1]]
        two = board[x[2]]
        if zero == one and one == two:
            if zero == "X":
                print ("AI wins.")
                end()
```

```
                if zero == "0":
                    print ("Human wins. Did you cheat?")
                    end()
            else:
                filledspaces = 0
                for i in range(8):
                    if board[i] != " " and board[i] != "_":
                        filledspaces += 1
                    if filledspaces == 8:
                        print ("A draw! You will never win!")
                        end()

    printboard(turn, board, aiturn)

# Displays the final board #

def end():
    print ("Here is the final board.")
    print (board[0] + "|" + board[1] + "|" + board[2])
    print (board[3] + "|" + board[4] + "|" + board[5])
    print (board[6] + "|" + board[7] + "|" + board[8])
    sys.exit(0)


printboard(turn, board, aiturn)
```