**TIC TAC TOE**

TIC TAC TOE is a game which I developed using python language.

The game is automatically played by the program and hence, no user input is needed. Still, developing an automatic game was lots of fun.

NumPy and random Python libraries are used to build this game. Instead of asking the user to put a mark on the board, the code randomly chooses a place on the board and put the mark. It will display the board after each turn unless a player wins. If the game gets drawn, then it returns -1.

**#Functions Used:**

play\_game() is the main function, which performs the following tasks :

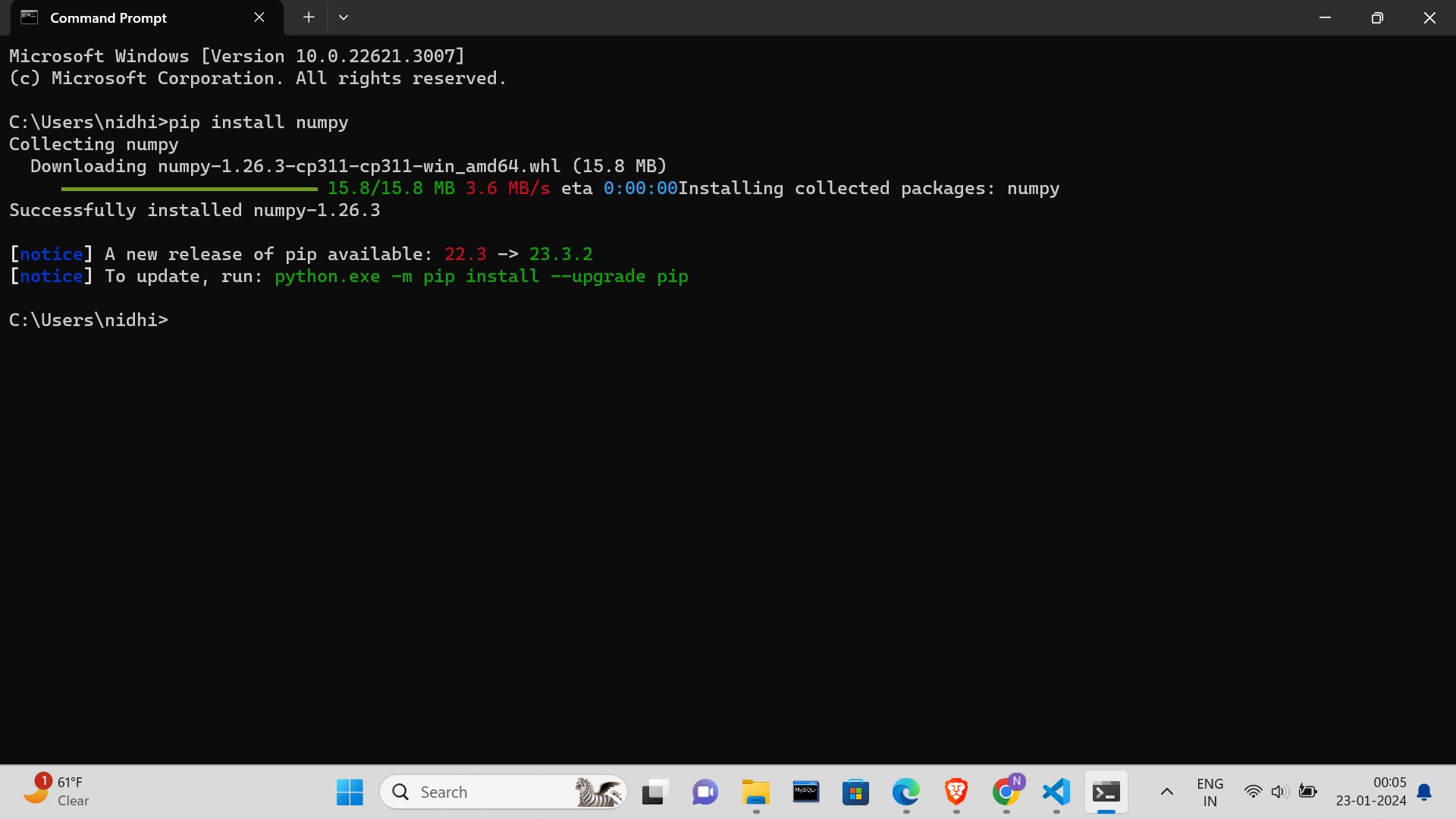
* Calls create\_board() to create a 3×3 board and initializes with 0.
* For each player (1 or 2), calls the random\_place() function to randomly choose a location on board and mark that location with the player number, alternatively.
* Print the board after each move.
* Evaluate the board after each move to check whether a row or column or diagonal has the same player number. If so, displays the winner’s name. If after 9 moves, there is no winner then displays -1

CODE:-

For coding it , we need to import NumPy library

Steps to import NumPy:

1. Open command prompt
2. Type : pip install numpy
3. This will show on the screen and after that you can run your program.



# Tic-Tac-Toe Program using

# random number in Python

# importing all necessary libraries

import numpy as np

import random

from time import sleep

# Creates an empty board

def create\_board():

return(np.array([[0, 0, 0],

[0, 0, 0],

[0, 0, 0]]))

# Check for empty places on board

def possibilities(board):

l = []

for i in range(len(board)):

for j in range(len(board)):

if board[i][j] == 0:

l.append((i, j))

return(l)

# Select a random place for the player

def random\_place(board, player):

selection = possibilities(board)

current\_loc = random.choice(selection)

board[current\_loc] = player

return(board)

# Checks whether the player has three

# of their marks in a horizontal row

def row\_win(board, player):

for x in range(len(board)):

win = True

for y in range(len(board)):

if board[x, y] != player:

win = False

continue

if win == True:

return(win)

return(win)

# Checks whether the player has three

# of their marks in a vertical row

def col\_win(board, player):

for x in range(len(board)):

win = True

for y in range(len(board)):

if board[y][x] != player:

win = False

continue

if win == True:

return(win)

return(win)

# Checks whether the player has three

# of their marks in a diagonal row

def diag\_win(board, player):

win = True

y = 0

for x in range(len(board)):

if board[x, x] != player:

win = False

if win:

return win

win = True

if win:

for x in range(len(board)):

y = len(board) - 1 - x

if board[x, y] != player:

win = False

return win

# Evaluates whether there is

# a winner or a tie

def evaluate(board):

winner = 0

for player in [1, 2]:

if (row\_win(board, player) or

col\_win(board, player) or

diag\_win(board, player)):

winner = player

if np.all(board != 0) and winner == 0:

winner = -1

return winner

# Main function to start the game

def play\_game():

board, winner, counter = create\_board(), 0, 1

print(board)

sleep(2)

while winner == 0:

for player in [1, 2]:

board = random\_place(board, player)

print("Board after " + str(counter) + " move")

print(board)

sleep(2)

counter += 1

winner = evaluate(board)

if winner != 0:

break

return(winner)

# Driver Code

print("Winner is: " + str(play\_game()))

**Output:**

