

**DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING
Faculty of Engineering and Technology
SRM Institute of Science and Technology**

18ECE201J – PYTHON AND SCIENTIFIC PYTHON

MINI PROJECT REPORT

ODD Semester, 2020-2021

Lab code & Sub Name : 18ECE201J – PYTHON & SCIENTIFIC PYTHON

Year & Semester : 3RD YEAR, 5TH SEMESTER

Project Title : **Tic-Tac-Toe Using Python**

Lab Supervisor : **Dr.K. Kalimuthu**
Associate Professor /ECE

Team Members :
1.Yajnish.M (RA1811004010291)
2. Prateeksha.G (RA1811004010319)
3. Ajay.K (RA1811004010315)

Tic-Tac-Toe Using Python

OBJECTIVE: To create the Tic-Tac-Toe game using python language from scratch.

ABSTRACT: Tic-tac-toe is a two-player game that is played on a 3×3 square grid. Each player occupies a cell in turns, with the objective of placing three marks in a horizontal, vertical, or diagonal pattern. One player uses cross 'X' as his marker, while the other uses a naught 'O'. This project shows how to create this game using simple functions in python language step by step.

INTRODUCTION: Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Its applications include web, game development, AI, machine learning, scientific & numeric applications and in operating systems.

SOFTWARE USED:

Anaconda3 (64 bit)

Jupyter Notebook (Anaconda3)

Python 3

```
In [*]: # PYTHON MINI PROJECT
# MEMBERS: Prateeksha.G(RA1811004010319)
#           Yajnish.M(ra1811004010291)
#           Ajay.K(RA1811004010315)
#Function to print Tic Tac Toe
def print_tic_tac_toe(values):
    print("\n")
    print("\t | \t | ")
    print("\t {} | {} | {}".format(values[0], values[1], values[2]))
    print('\t____|____|____')

    print("\t | \t | ")
    print("\t {} | {} | {}".format(values[3], values[4], values[5]))
    print('\t____|____|____')

    print("\t | \t | ")

    print("\t {} | {} | {}".format(values[6], values[7], values[8]))
    print("\t | \t | ")
    print("\n")

# Function to print the score-board
def print_scoreboard(score_board):
    print("\t-----\t")
    print("\t\t\t\t\t SCOREBOARD\t\t\t")
    print("\t-----\t")

    players = list(score_board.keys())
    print("\t ", players[0], "\t ", score_board[players[0]])
    print("\t ", players[1], "\t ", score_board[players[1]])

    print("\t-----\n")

# Function to check if any player has won
def check_win(player_pos, cur_player):
    # All possible winning combinations
    soln = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [1, 4, 7], [2, 5, 8], [3, 6, 9], [1, 5, 9], [3, 5, 7]]

    # Loop to check if any winning combination is satisfied
    for x in soln:
        if all(y in player_pos[cur_player] for y in x):
            # Return True if any winning combination satisfies
            return True
    # Return False if no combination is satisfied
    return False

# Function to check if the game is drawn
def check_draw(player_pos):
    if len(player_pos['X']) + len(player_pos['O']) == 9:
        return True
    return False

# Function for a single game of Tic Tac Toe
def single_game(cur_player):
    # Represents the Tic Tac Toe
    values = [' ' for x in range(9)]

    # Stores the positions occupied by X and O
    player_pos = {'X':[], 'O':[]}

    # Game Loop for a single game of Tic Tac Toe
    while True:
        print_tic_tac_toe(values)
```

```

# Game Loop for a single game of Tic Tac Toe
while True:
    print_tic_tac_toe(values)

    # Try exception block for MOVE input
    try:
        print("Player ", cur_player, " turn. Which box? : ", end="")
        move = int(input())
    except ValueError:
        print("Wrong Input!!! Try Again")
        continue

    # Sanity check for MOVE inout
    if move < 1 or move > 9:
        print("Wrong Input!!! Try Again")
        continue

    # Check if the box is not occupied already
    if values[move-1] != ' ':
        print("Place already filled. Try again!!")
        continue

    # Update game information

    # Updating grid status
    values[move-1] = cur_player

    # Updating player positions
    player_pos[cur_player].append(move)

    # Function call for checking win
    if check_win(player_pos, cur_player):
        print_tic_tac_toe(values)
        print("Player ", cur_player, " has won the game!!")
        print("\n")
        return cur_player

    # Function call for checking draw game
    if check_draw(player_pos):
        print_tic_tac_toe(values)
        print("Game Drawn")
        print("\n")
        return 'D'

    # Switch player moves
    if cur_player == 'X':
        cur_player = 'O'
    else:
        cur_player = 'X'

if __name__ == "__main__":

    print("Player 1")
    player1 = input("Enter the name : ")
    print("\n")

    print("Player 2")
    player2 = input("Enter the name : ")
    print("\n")

    # Stores the player who chooses X and O
    cur_player = player1

    # Stores the choice of players
    player_choice = {'X' : "", 'O' : ""}

    # Stores the options

```

```

player_choice = {'X' : "", 'O' : ""}

# Stores the options
options = ['X', 'O']

# Stores the scoreboard
score_board = {player1: 0, player2: 0}
print_scoreboard(score_board)

# Game Loop for a series of Tic Tac Toe
# The loop runs until the players quit
while True:

    # Player choice Menu
    print("Turn to choose for", cur_player)
    print("Enter 1 for X")
    print("Enter 2 for O")
    print("Enter 3 to Quit")

    # Try exception for CHOICE input
    try:
        choice = int(input())
    except ValueError:
        print("Wrong Input!!! Try Again\n")
        continue

    # Conditions for player choice
    if choice == 1:
        player_choice['X'] = cur_player
        if cur_player == player1:
            player_choice['O'] = player2
        else:
            player_choice['O'] = player1

    elif choice == 2:
        player_choice['O'] = cur_player
        if cur_player == player1:
            player_choice['X'] = player2
        else:
            player_choice['X'] = player1

    elif choice == 3:
        print("Final Scores")
        print_scoreboard(score_board)
        break

    else:
        print("Wrong Choice!!!! Try Again\n")

    # Stores the winner in a single game of Tic Tac Toe
    winner = single_game(options[choice-1])

    # Edits the scoreboard according to the winner
    if winner != 'D' :
        player_won = player_choice[winner]
        score_board[player_won] = score_board[player_won] + 1

    print_scoreboard(score_board)
    # Switch player who chooses X or O
    if cur_player == player1:
        cur_player = player2
    else:
        cur_player = player1

```

DETAILED EXPLANATION:

#1 Design:

```
: # PYTHON MINI PROJECT
# MEMBERS: Prateeksha.G(RA1811004010319)
#           Yajnish.M(RA1811004010291)
#           Ajay.K(RA1811004010315)
#Function to print Tic Tac Toe
def print_tic_tac_toe(values):
    print("\n")
    print("\t\t | \t\t |")
    print("\t {} | {} | {}".format(values[0], values[1], values[2]))
    print('\t_____|_____|_____')

    print("\t\t | \t\t |")
    print("\t {} | {} | {}".format(values[3], values[4], values[5]))
    print('\t_____|_____|_____')

    print("\t\t | \t\t |")

    print("\t {} | {} | {}".format(values[6], values[7], values[8]))
    print("\t\t | \t\t |")
    print("\n")
```

In the code above, the function creates our tic-tac-toe game according to the values delivered as an argument. Here the argument, **values** is a list containing the status of each cell in the grid.

#2 Storing Information Using Data Structures:

```
# Function for a single game of Tic Tac Toe
def single_game(cur_player):

    # Represents the Tic Tac Toe
    values = [' ' for x in range(9)]

    # Stores the positions occupied by X and O
    player_pos = {'X':[], 'O':[]}
```

At any instant of time, we need two crucial information:

Status of the grid – We must have a data structure that stores each cell's state, that is, whether it is occupied or vacant.

Each player's moves – We must somehow have the knowledge of each player's past and present moves, that is, the positions occupied by 'X' and 'O'.

Status of the grid is managed by a list of characters, which can have three possible values,

' ' – A vacant cell

'X' – A cell occupied by player X

'O' – A cell occupied by player O

Each player's moves are stored as a dictionary of a list of integers. The keys are 'X' and 'O' for the respective player. Their corresponding lists contain the numbers given to the grid cells, they occupy.

The variable **cur_player**, stores the current player making the move, as in 'X' or 'O'.

#3 Game Loop & Handling Player Input:

```
# Game Loop for a single game of Tic Tac Toe
while True:
    print_tic_tac_toe(values)

    # Try exception block for MOVE input
    try:
        print("Player ", cur_player, " turn. Which box? : ", end="")
        move = int(input())
    except ValueError:
        print("Wrong Input!!! Try Again")
        continue

    # Sanity check for MOVE input
    if move < 1 or move > 9:
        print("Wrong Input!!! Try Again")
        continue

    # Check if the box is not occupied already
    if values[move-1] != ' ':
        print("Place already filled. Try again!!")
        continue
```

Every game, has some kind of game loop, which runs until some player wins or the game ends in a draw. In tic-tac-toe, each loop iteration refers to a single move any player makes.

In every game iteration, a player must input his move. We create a **try** block, in case a player enters some unintended value. Such an event must not stop the game, therefore, we handle the exception of **ValueError** and continue with our game.

We need to perform some checks, like value entered is a valid position and if it is a valid position, is it already occupied?

#4 Updating & Checking Win or Draw:

Function Calls:

```
# Update game information

# Updating grid status
values[move-1] = cur_player

# Updating player positions
player_pos[cur_player].append(move)

# Function call for checking win
if check_win(player_pos, cur_player):
    print_tic_tac_toe(values)
    print("Player ", cur_player, " has won the game!!")
    print("\n")
    return cur_player

# Function call for checking draw game
if check_draw(player_pos):
    print_tic_tac_toe(values)
    print("Game Drawn")
    print("\n")
    return 'D'
```

According to the player input, we need to update the information for the smooth functioning of the game. The **values** list updates the cell occupied according to the current player. The player position adds the position just taken by the current player.

After each move, we have to check whether any player won the game or the game has been drawn.

If any player wins, then the **single_game()** function returns the current player, who made the move. In case, the game is drawn, **'D'** is sent back.

check_win() – The function has all the winning combinations. All it does is, it checks whether any of the winning combinations is satisfied by the current player's positions. If it does, it returns **True**. If none of the combinations is satisfied, then the function returns **False**.

check_draw() – The draw condition is fairly simple, as the game is drawn when all 'nine' positions are taken.

#5 Switching & Entering the Player Names:

```
# Switch player moves
if cur_player == 'X':
    cur_player = 'O'
else:
    cur_player = 'X'

if __name__ == "__main__":

    print("Player 1")
    player1 = input("Enter the name : ")
    print("\n")

    print("Player 2")
    player2 = input("Enter the name : ")
    print("\n")
```

Since each player only moves once at a time, therefore after every successful move, we have to swap the current player.

This article also presents a scoreboard system for keeping track, if the players want to play multiple games and it is mandatory for any scoreboard to display each player names.

#6 Storing Game Information & Designing of the Scoreboard:

```
# Stores the player who chooses X and O
cur_player = player1

# Stores the choice of players
player_choice = {'X' : "", 'O' : ""}

# Stores the options
options = ['X', 'O']

# Stores the scoreboard
score_board = {player1: 0, player2: 0}
print_scoreboard(score_board)

# Function to print the score-board
def print_scoreboard(score_board):
    print("\t-----")
    print("\t\t\t\t\tSCOREBOARD\t\t\t\t\t")
    print("\t-----")

    players = list(score_board.keys())
    print("\t\t", players[0], "\t\t\t\t", score_board[players[0]])
    print("\t\t", players[1], "\t\t\t\t", score_board[players[1]])

    print("\t-----\n")
```

The information like the current player, the choice of players (take cross or naught), the available options (cross and naught), and the scoreboard need to be stored. By default, the current player is the player who entered the name first.

The scoreboard is stored as a dictionary, where keys are the player names and values are their win number.

To display the scoreboard, we need the player names. The keys are extracted using `.keys()` function and then converted to list, so that it can indexed while displaying the scores.

#7 Handling & Assigning Player Choice:

```
# Try exception for CHOICE input
try:
    choice = int(input())
except ValueError:
    print("Wrong Input!!! Try Again\n")
    continue

# Conditions for player choice
if choice == 1:
    player_choice['X'] = cur_player
    if cur_player == player1:
        player_choice['O'] = player2
    else:
        player_choice['O'] = player1

elif choice == 2:
    player_choice['O'] = cur_player
    if cur_player == player1:
        player_choice['X'] = player2
    else:
        player_choice['X'] = player1

elif choice == 3:
    print("Final Scores")
    print_scoreboard(score_board)
    break

else:
    print("Wrong Choice!!!! Try Again\n")
```

Each iteration, we have to handle and store current player's choice. According to the player's choice, the data has been stored. This is important since after each game finishes, it will tell us which player won.

#8 Execution & Updating the Scoreboard:

```
# Stores the winner in a single game of Tic Tac Toe
winner = single_game(options[choice-1])

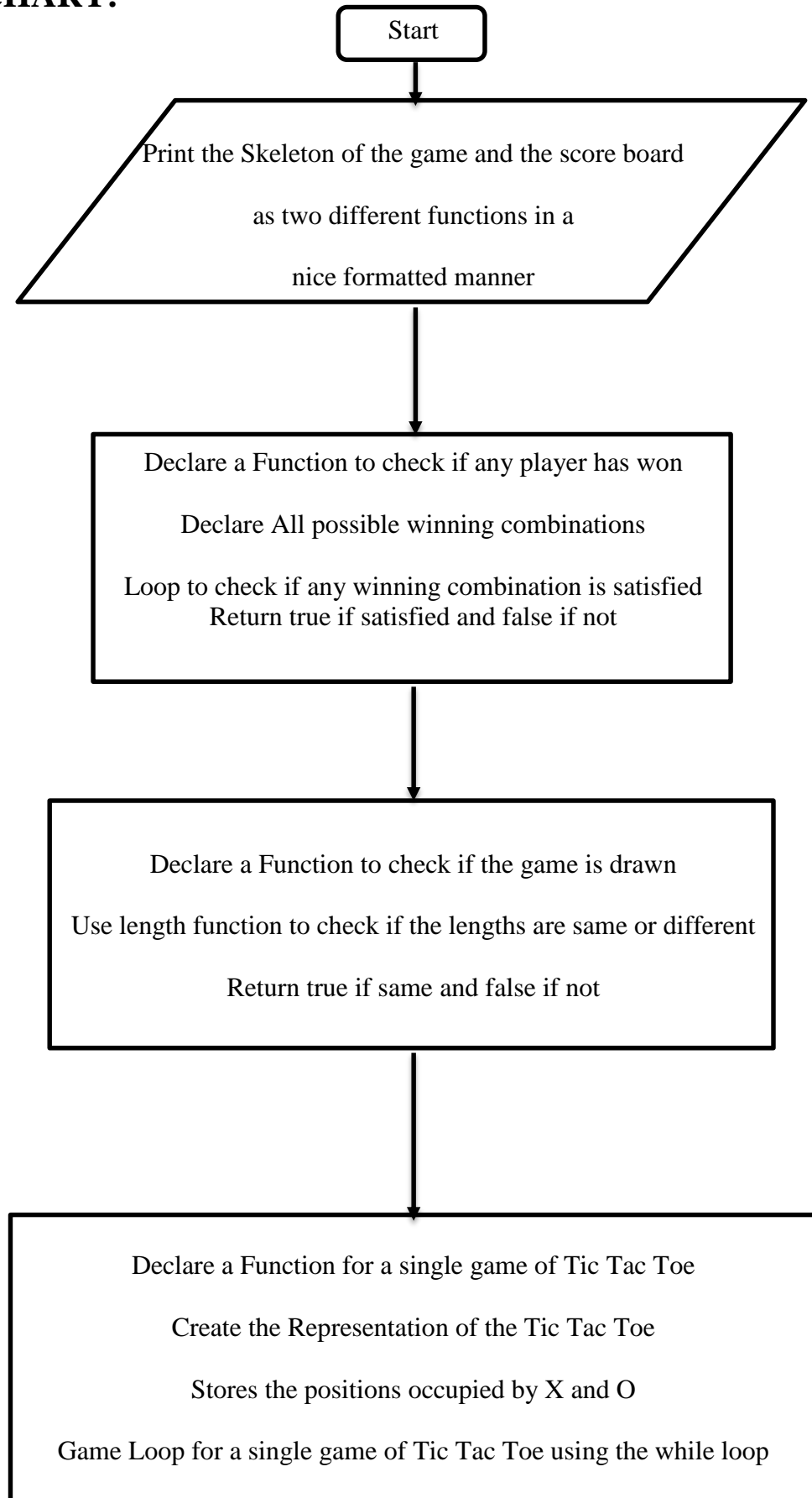
# Updates the scoreboard according to the winner
if winner != 'D' :
    player_won = player_choice[winner]
    score_board[player_won] = score_board[player_won] + 1

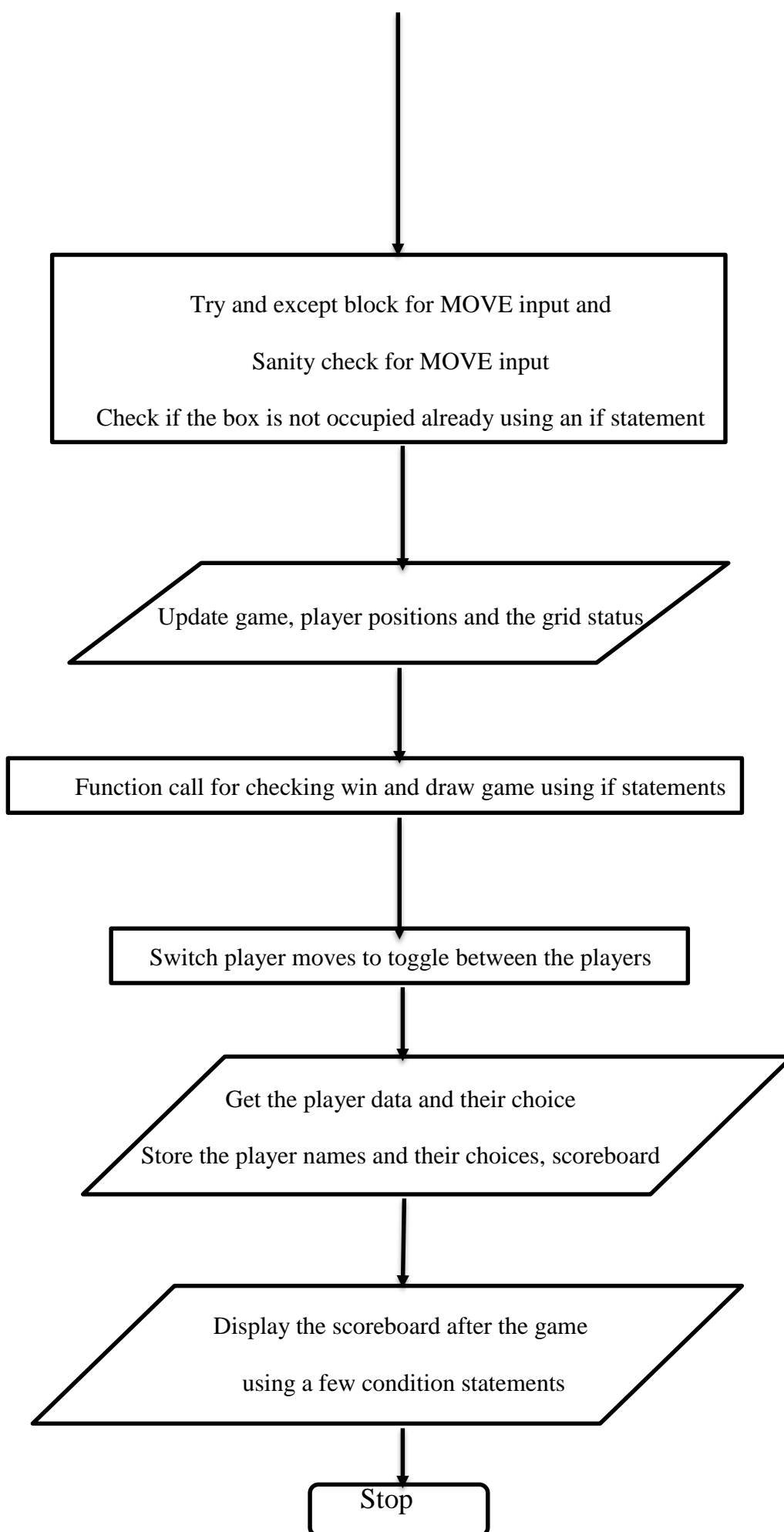
print_scoreboard(score_board)
```

After storing all the necessary information, it is time to execute an independent match and store the winning mark.

We need to update the scoreboard after each match of Tic-tac-toe. If the game has not ended in a draw, then we update the scoreboard.

FLOWCHART:





OUTPUT:

```
jupyter PYTHON_PROJECT_319_291_315 Last Checkpoint: 4 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
+ < > Run Code

Player 1
Enter the name : PRATEEKSHA

Player 2
Enter the name : AJAY

-----
SCOREBOARD
-----
PRATEEKSHA    0
AJAY          0
-----

Turn to choose for PRATEEKSHA
Enter 1 for X
Enter 2 for O
Enter 3 to Quit
1

  | | 
--|_|
  | | 
  | | 
  | | 

Player X turn. Which box? : 5



  | | 
--|_|
  |X|
  | | 
  | | 

Player O turn. Which box? : 1

  O| | 
--|_|
  |X|
  | | 

Player X turn. Which box? : 2

  O|X| 
--|_|
```


jupyter
PYTHON_PROJECT_319_291_315
Last Checkpoint: 4 minutes ago (autosaved)

Logout

File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3

+
⌂
🔍
📄
⬆️
⬆️
⬆️
Run
⏏️
🔄
⏏️
Code
⌵
📄

Player X turn. Which box? : 2

O	X	
	X	

Player O turn. Which box? : 4

O	X	
O	X	

Player X turn. Which box? : 8

O	X	
O	X	
	X	

Player X has won the game!!

```

-----
SCOREBOARD
-----
PRATEEKSHA      1
AJAY            0
-----

```

Turn to choose for AJAY
Enter 1 for X
Enter 2 for O
Enter 3 to Quit

In []:

CONCLUSION: After successfully building this game, we were able to get a clear idea about dictionaries in python, how to access dictionaries, how to iterate over dictionaries, for loop, if-else conditions and functions in python language.

RESULT: Thus the creation and execution of the Tic-Tac-Toe game was successfully completed in Python programming language.