

**SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS**

**Winter Semester 2023-2024**

**PMCA602P: PYTHON**

**Faculty Name: Prof. Shynu PG**

**PYTHON PROJECT**

**TIC TAC TOE**

**Name**: **Rishabh Jaiswal**

**Reg. No**: **23MCA0254**

**Tic Tac Toe**

**Objective:**

The objective of this project is to implement and document a Python-based Tic-Tac-Toe game. The game involves two players competing against each other to mark a row, column, or diagonal with their designated symbol (X or O) before their opponent does. The project aims to demonstrate proficiency in Python programming while providing an engaging gaming experience.

**Report Requirements:**

**Introduction:**

The Tic-Tac-Toe game is a classic two-player game that challenges players' strategic thinking and decision-making skills. The objective of the game is simple: to be the first player to form a horizontal, vertical, or diagonal line with their symbol on a 3x3 grid. This project aims to implement a digital version of the Tic-Tac-Toe game using Python, allowing players to compete against each other in a virtual environment.

**Methodology:**

**Design:**

The Tic-Tac-Toe game will be designed using object-oriented programming principles in Python. The game will consist of a 3x3 grid represented as a list of lists, where each cell can contain either 'X', 'O', or an empty space. Players will take turns entering their moves until one player wins or the game ends in a draw.

**Implementation Details:**

The implementation will involve creating a Python class to represent the game. This project will include methods for initializing the game, displaying the game board, processing player moves, checking for a win or draw condition, and determining the winner.

* **Tkinter:** Used for creating the graphical user interface.

Here is the code snippet for each function:

1. **start\_game ():**

def start\_game(self):

if self.player1\_name.get() == "" or self.player2\_name.get() == "":

messagebox.showerror("Tic Tac Toe", "Please enter player names.")

else:

self.player\_name\_frame.destroy()

self.create\_game\_frame()

messagebox.showinfo("Tic Tac Toe", f"Let's start the game between {self.player1\_name.get()} and {self.player2\_name.get()}!")

* This function is called when the "Start Game" button is clicked.
* It checks if both player names are entered. If not, it displays an error message.
* If both player names are entered, it destroys the player name input frame and creates the game frame.
* It displays an information message indicating the start of the game with the names of the players.

1. **button\_clicked ():**

def button\_clicked(self, button):

if button["text"] == " " and self.clicked:

button["text"] = "X"

self.clicked = False

self.count += 1

elif button["text"] == " " and not self.clicked:

button["text"] = "O"

self.clicked = True

self.count += 1

else:

messagebox.showerror("Tic Tac Toe", "Please select another box.")

return

self.check\_winner()

self.check\_draw()

* This function is called when a button on the game grid is clicked.
* It checks if the clicked button is empty and if it's the current player's turn.
* If the conditions are met, it places "X" or "O" symbol on the button based on the current player's turn.
* If the clicked button is not empty, it displays an error message.
* After placing the symbol, it checks for a win or draw condition.

1. **restart\_game ():**

def restart\_game(self):

self.game\_frame.forget()

self.play\_again()

self.create\_player\_name\_frame()

* This function is called to restart the game.
* It clears the game frame by forgetting it.
* It resets game variables and clears the game grid using the `play\_again` method.
* It recreates the player name input frame to allow players to enter their names again.

**Results:**

The functionality of the Tic-Tac-Toe game is illustrated through screenshots or code outputs, demonstrating:

* The game interface displaying the grid with clickable buttons for each space.
* Player names entered before starting the game.
* Player moves being registered and updated on the game board.
* Win/lose conditions being detected and appropriate messages displayed to the players.

**Challenges and Learnings:**

The development of the Tic-Tac-Toe game presented several challenges, including:

* Implementing the game logic to handle player moves and win/lose conditions efficiently.
* Designing a user-friendly GUI interface that enhances the gaming experience.
* Handling edge cases and error conditions to ensure smooth gameplay.

Through overcoming these challenges, valuable lessons were learned in Python programming, GUI design, and problem-solving techniques. The project provided an opportunity to apply theoretical knowledge to real-world scenarios and gain practical experience in software development.

**Conclusion:**

In conclusion, the Tic-Tac-Toe game project demonstrates the interactive implementation a digital version of the classic game, players can engage in strategic gameplay while honing their problem-solving abilities. Through this project, valuable lessons in software development, game design, and Python programming have been acquired, contributing to personal growth and skill enhancement. Overall, the Tic-Tac-Toe game project represents a successful endeavor in leveraging Python for creating entertaining and educational gaming applications.

**Source code:**

from tkinter import \*

from tkinter import messagebox

class TicTacToe:

def \_init\_(self, root):

self.root = root

self.root.title('Tic Tac Toe')

self.player1\_name = StringVar()

self.player2\_name = StringVar()

self.winner\_name = StringVar() # Variable to store the winner's name

self.clicked = True

self.count = 0

self.winner = False

self.create\_player\_name\_frame()

def create\_player\_name\_frame(self):

self.player\_name\_frame = Frame(self.root)

self.player\_name\_frame.pack()

player1\_label = Label(self.player\_name\_frame, text="Player 1 Name:")

player1\_label.grid(row=0, column=0, padx=10, pady=5)

player1\_entry = Entry(self.player\_name\_frame, textvariable=self.player1\_name)

player1\_entry.grid(row=0, column=1, padx=10, pady=5)

player2\_label = Label(self.player\_name\_frame, text="Player 2 Name:")

player2\_label.grid(row=1, column=0, padx=10, pady=5)

player2\_entry = Entry(self.player\_name\_frame, textvariable=self.player2\_name)

player2\_entry.grid(row=1, column=1, padx=10, pady=5)

start\_button = Button(self.player\_name\_frame, text="Start Game", command=self.start\_game,foreground="white", background="red")

start\_button.grid(row=2, columnspan=2, padx=10, pady=5)

def create\_game\_frame(self):

self.game\_frame = Frame(self.root)

self.game\_frame.pack()

self.player1\_label = Label(self.game\_frame, text=f'Player 1: {(self.player1\_name.get())}', foreground="blue", font=("Helvetica", 10))

self.player1\_label.pack(side=LEFT, padx=10, pady=5)

self.player2\_label = Label(self.game\_frame, text=f'Player 2: {self.player2\_name.get()}', foreground="red", font=("Helvetica", 10))

self.player2\_label.pack(side=RIGHT, padx=20, pady=5)

self.winner\_label = Label(self.game\_frame, textvariable=self.winner\_name, foreground="green", font=("Helvetica", 10))

self.winner\_label.pack(padx=10, pady=5)

self.create\_buttons()

self.play\_again\_button = Button(self.game\_frame, text="Play Again", command=self.play\_again, foreground="white", background="green")

self.play\_again\_button.pack(side=LEFT, padx=10, pady=10)

self.restart\_button = Button(self.game\_frame, text="Restart Game", command=self.restart\_game, foreground="white", background="blue")

self.restart\_button.pack(side=LEFT, padx=10)

self.end\_button=Button(self.game\_frame,text="Quit", command=self.quit\_game,foreground="white", background="red")

self.end\_button.pack(side=RIGHT,padx=10,pady=5)

def create\_buttons(self):

self.buttons = []

for i in range(3):

frame = Frame(self.game\_frame)

frame.pack()

for j in range(3):

button = Button(frame, text=" ", font=("Helvetica, 20"), height=3, width=7, bg="White")

button.pack(side=LEFT)

button.config(command=lambda current\_button=button: self.button\_clicked(current\_button))

self.buttons.append(button)

def disable\_buttons(self):

for button in self.buttons:

button.config(state=DISABLED)

def check\_winner(self):

patterns = [[0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 3, 6], [1, 4, 7], [2, 5, 8], [0, 4, 8], [2, 4, 6]]

for pattern in patterns:

if self.buttons[pattern[0]]["text"] == self.buttons[pattern[1]]["text"] == self.buttons[pattern[2]]["text"] != " ":

for i in pattern:

self.buttons[i].config(bg="#80ffaa")

self.winner = True

if self.buttons[pattern[0]]["text"] == "X":

self.winner\_name.set(f"{self.player1\_name.get()} is the Winner!")

else:

self.winner\_name.set(f"{self.player2\_name.get()} is the Winner!")

self.disable\_buttons()

def check\_draw(self):

if self.count == 9 and not self.winner:

self.winner\_name.set("It's a Draw!")

def button\_clicked(self, button):

if button["text"] == " " and self.clicked:

button["text"] = "X"

self.clicked = False

self.count += 1

elif button["text"] == " " and not self.clicked:

button["text"] = "O"

self.clicked = True

self.count += 1

else:

messagebox.showerror("Tic Tac Toe", "Please select another box.")

return

self.check\_winner()

self.check\_draw()

def start\_game(self):

if self.player1\_name.get() == "" or self.player2\_name.get() == "":

messagebox.showerror("Tic Tac Toe", "Please enter player names.")

else:

self.player\_name\_frame.destroy()

self.create\_game\_frame()

messagebox.showinfo("Tic Tac Toe", f"Let's start the game between {self.player1\_name.get()} and {self.player2\_name.get()}!")

def play\_again(self):

for button in self.buttons:

button["text"] = " "

button.config(bg="White", state=NORMAL)

self.winner\_name.set("")

self.count = 0

self.winner = False

self.player1\_name.set("")

self.player2\_name.set("")

def restart\_game(self):

self.game\_frame.forget()

self.play\_again()

# self.player\_name\_frame = Frame(self.root)

# self.player\_name\_frame.pack()

self.create\_player\_name\_frame()

def quit\_game(self):

self.root.destroy()

root = Tk()

game = TicTacToe(root)

root.mainloop()

**Screenshots:**







