Tic Tac Toe in CLI

A CLI version of the popular paper-and-pencil game tic-tac-toe.

Alex Cole
Hackbright Academy

March 10, 2023
DevOps Presentation

About Me

Hi There!

Allow me to introduce myself,

My name is Alex. I'm twenty-seven years old and was born, and raised, in a small town in Alabama called Altoona.

Some previous professional experience I have is retail management, electrical contracting, and water operations for my town. Due to these unique opportunities I have fortunately been able to broaden my skills and network with great people.

Why software engineering?

My father had a Debian server when I was young and I can remember watching him run commands and various jobs on it. Of course I was curious and would pester him with questions, and thankfully he was patient enough to explain things to me the best he could! Thanks to him I had a proficiency for computers at a young age.

I have helped countless people with their computer troubles, set up computer applications for my high-school, and even introduced a crontab for a church's chimes. My proficiency paved a way for me in my early teens, but it was only a hobby. I did not realize my eagerness for engineering until this opportunity from Shipt.

»

Project Overview

Tic Tac Toe

Technical Stack:

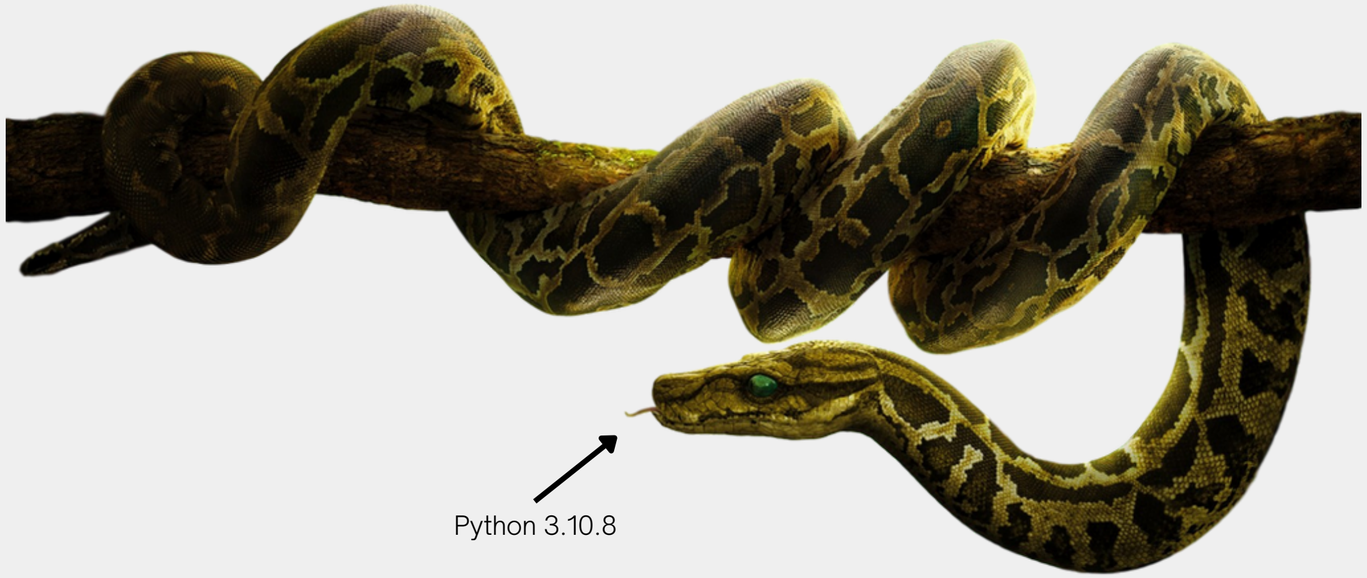
Python
Random Module
PyInstaller (for .exe file)

Why?

The purpose of this project was to showcase my (fairly new) understanding of Python. I accomplished what I set out for in this project and also gave it a sprinkle of that syntactic sugar that we all know and love.

What's Next?

Player vs AI
Timer
Leaderboards



Python 3.10.8

»

Project Overview

The Deets

Fig. 1

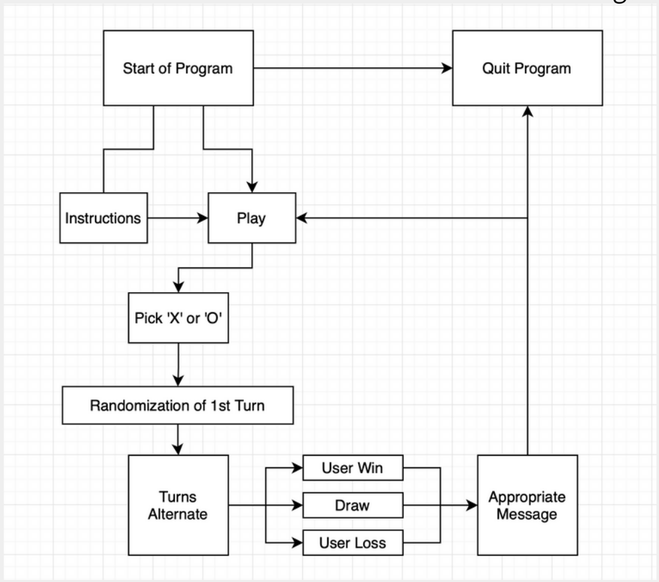


Figure 1 is a flowchart I created for showcasing how the game will work. I used this as a guide while building the program and it allowed me to be effecient with my coding.

Fig. 2

```
def display_board(board):
    # This function is the tic-tac-toe board with the {}s formatted to numbers 1-9.
    print("     " + " {} | {} | {} ".format(board[7], board[8], board[9]))
    print("     " + "-----------")
    print("     " + " {} | {} | {} ".format(board[4], board[5], board[6]))
    print("     " + "-----------")
    print("     " + " {} | {} | {} ".format(board[1], board[2], board[3]))
```

Figure 2 is the **display_board** function which, as you may have guessed, displays the board for the user in the CLI. Another feature of this function is that it formats the whitespace on the board (where X or O would go) to numbers between 1-9. This is ultimately what allows users to select a specific space to place their mark.

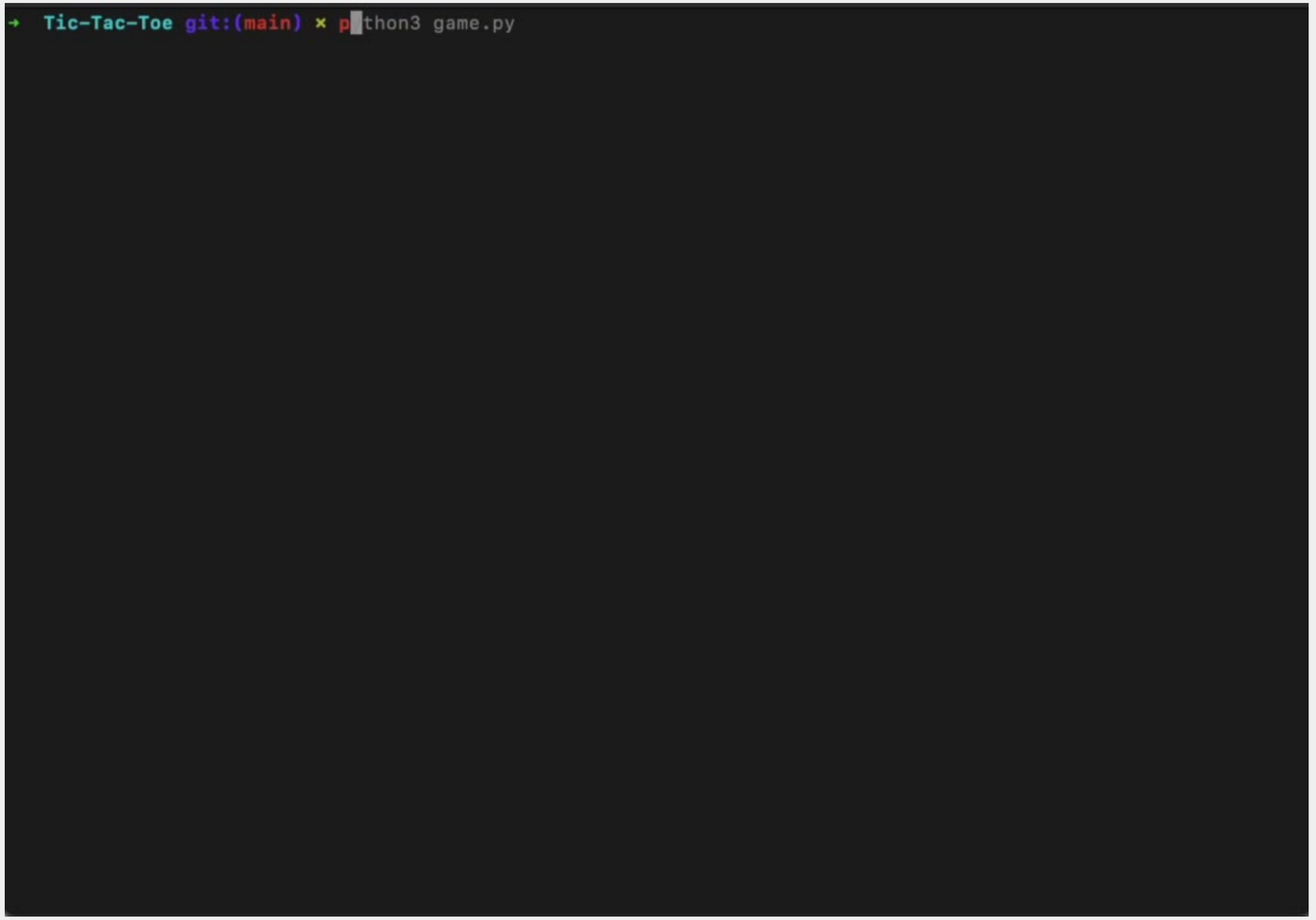
```
def win_check(board, choice):
    # This function checks if there is a pattern of three Xs or Os.

    return (
            # Horizontal patterns:
            (board[1] == choice and board[2] == choice and board[3] == choice)
            or (board[4] == choice and board[5] == choice and board[6] == choice)
            or (board[7] == choice and board[8] == choice and board[9] == choice)
            # Vertical patterns:
            or (board[1] == choice and board[4] == choice and board[7] == choice)
            or (board[2] == choice and board[5] == choice and board[8] == choice)
            or (board[3] == choice and board[6] == choice and board[9] == choice)
            # Diagonal patterns:
            or (board[1] == choice and board[5] == choice and board[9] == choice)
            or (board[3] == choice and board[5] == choice and board[7] == choice))
```

Fig. 3

Figure 3 is the **win_check** function; all it does is determine if you win! It will check the board for a pattern. If there is one either vertically, horizontally, or diagonally then the game is over and the appropriate player will receive the winning message!

»

The Demo



Thank You

Alex Cole
github.com/alexhcole95/
linkedin.com/in/alexhcole
alexhcole95@gmail.com