DELHI TECHNOLOGICAL UNIVERSITY

TIC TAC TOE IMPLEMENTATION USING SOCKET PROGRAMMING

WT CO7 Project Report

Prof. Chingmuankim Naulak

SUBMITTED BY:

PRAJJWAL CHITTORI(2K18/CO/249)

ABSTRACT:

The Tic-Tac-Toe game is a popular two-player game. We have tried to implement a simulation of this game being played optimally by two players and trying to win. We have also implemented an AI player that employs Alpha Beta Pruning to play intelligently.
We created a tic tac toe game in Python utilising socket programming, where the game creates the server across a local area network. The Client Nodes are connected to the server by utilising the server's IP address.

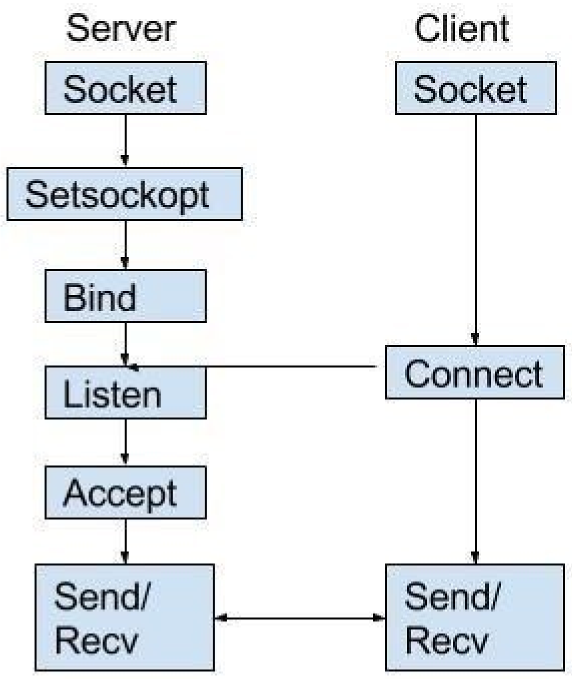
AIM: To implement Tic-Tac-Toe using Socket Programming.

SOCKET PROGRAMMING: Socket programming is a method of joining and operating two nodes in a network that communicate and transfer information. One socket listens on a specific port at an IP address, while another socket establishes a connection with it. While the client connects to the server, the server creates the listener socket.

Module Socket()
To create the server and client for the programme, we used the socket methods listed below.

- connect()
- bind( )
- listen(backlog)
- accept( )

- s.recv()



ACCOUNT REGISTRATION AND LOGIN USING SQLite DATABASE:
SQLite is a C library that contains a relational database management system (RDBMS). SQLite is a client–server database engine, not a client–server database engine. It's built into the final product.

SQLite makes use of a dynamically typed SQL syntax that does not ensure domain integrity.

Highlights of SQLite:

- SQLite doesn't need a different worker interaction or framework to work (serverless).
- SQLite accompanies zero-arrangement, which implies no arrangement or organization required.
- A total SQLite information base is put away in a solitary cross- stage plate record.
- SQLite is little and light weight, under 400KiB completely arranged or under 250KiB with discretionary highlights overlooked.
- SQLite is independent, which implies no outside conditions.
- SQLite exchanges are completely ACID-consistent, permitting safe access from numerous cycles or strings.
- SQLite upholds the majority of the question language highlights found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and gives a straightforward and simple- to-utilize API.
- SQLite is accessible on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

GUI IMPLEMENTATION USING TKINTER MODULE:

Python has numerous choices for creating Graphical User Interface. tkinter is the most ordinarily utilized strategy. It is a standard Python interface to the Tk GUI toolbox dispatched with Python.

Tkinter is the Python port for the Tcl-Tk GUI tool stash created by Fredrik Lundh.

Python
GUI Programming with Tkinter
01
Options for Python GUI Programming
02
Tkinter
03
Widgets

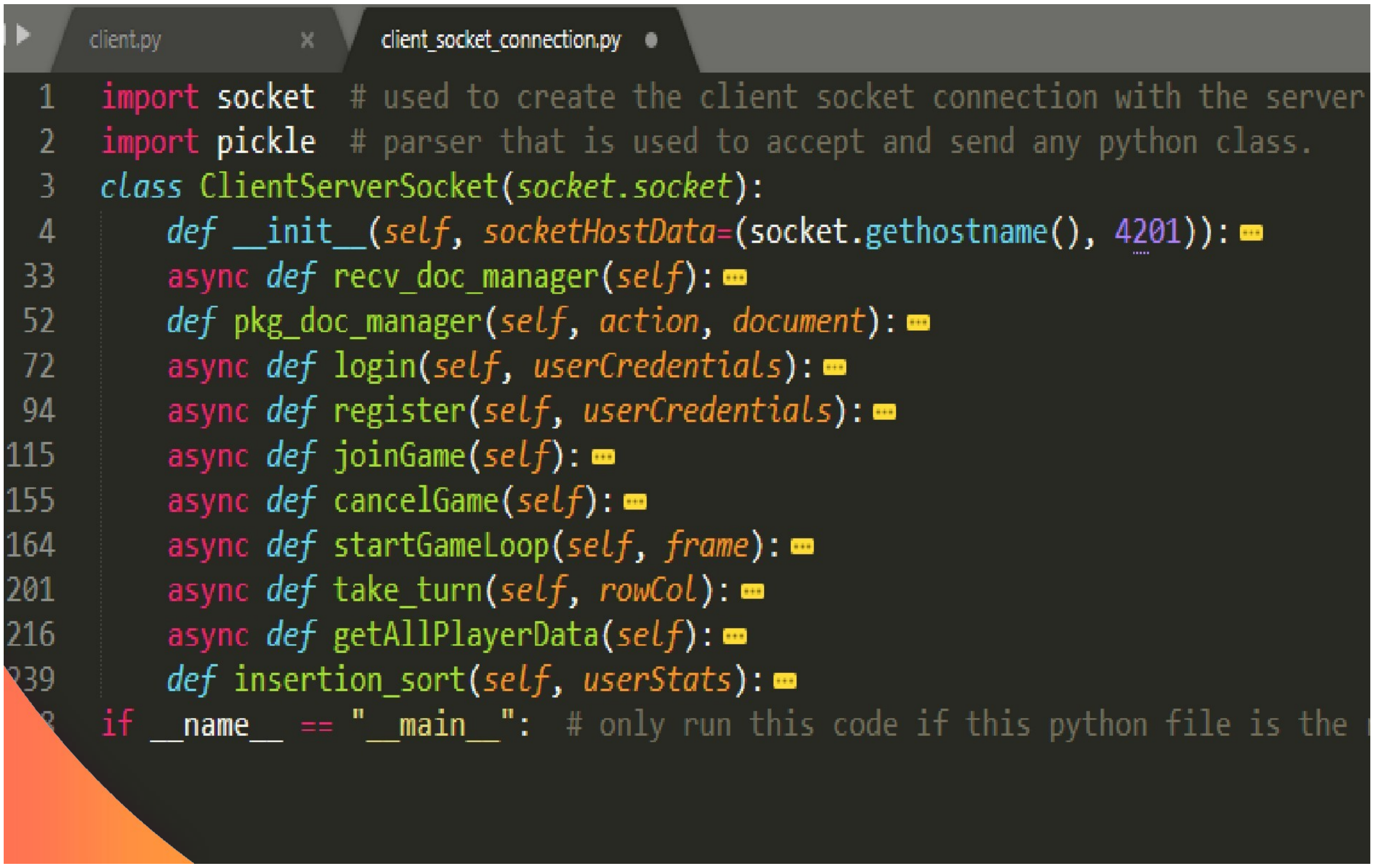
- We have utilized tkinter module to execute the Graphical UI for our application.
- Tkinter gives different controls, like catches, marks and text confines utilized a GUI application. These controls are usually called gadgets

- A portion of the gadgets we have utilized in our application are:
- Button:To add a catch in your application, this gadget is utilized.
- The overall language structure is:
- w=Button(master, option=value)
- Number of choices can be passed as boundaries isolated by commas. Some of them are recorded underneath.
- activebackground: to set the foundation shading when the catch is under the cursor.
- bg: to set the typical foundation tone.
- order: to call a capacity.
- textual style: to set the textual style on the catch mark.
- picture: to set the picture on the catch.
- width: to set the width of the catch.
- tallness: to set the stature of the catch.
- Material: It is utilized to draw pictures and other complex formats like illustrations, text and gadgets. The overall punctuation is: w = Canvas(master, option=value)
- Number of choices can be passed as boundaries isolated by commas. Some of them

are recorded beneath.

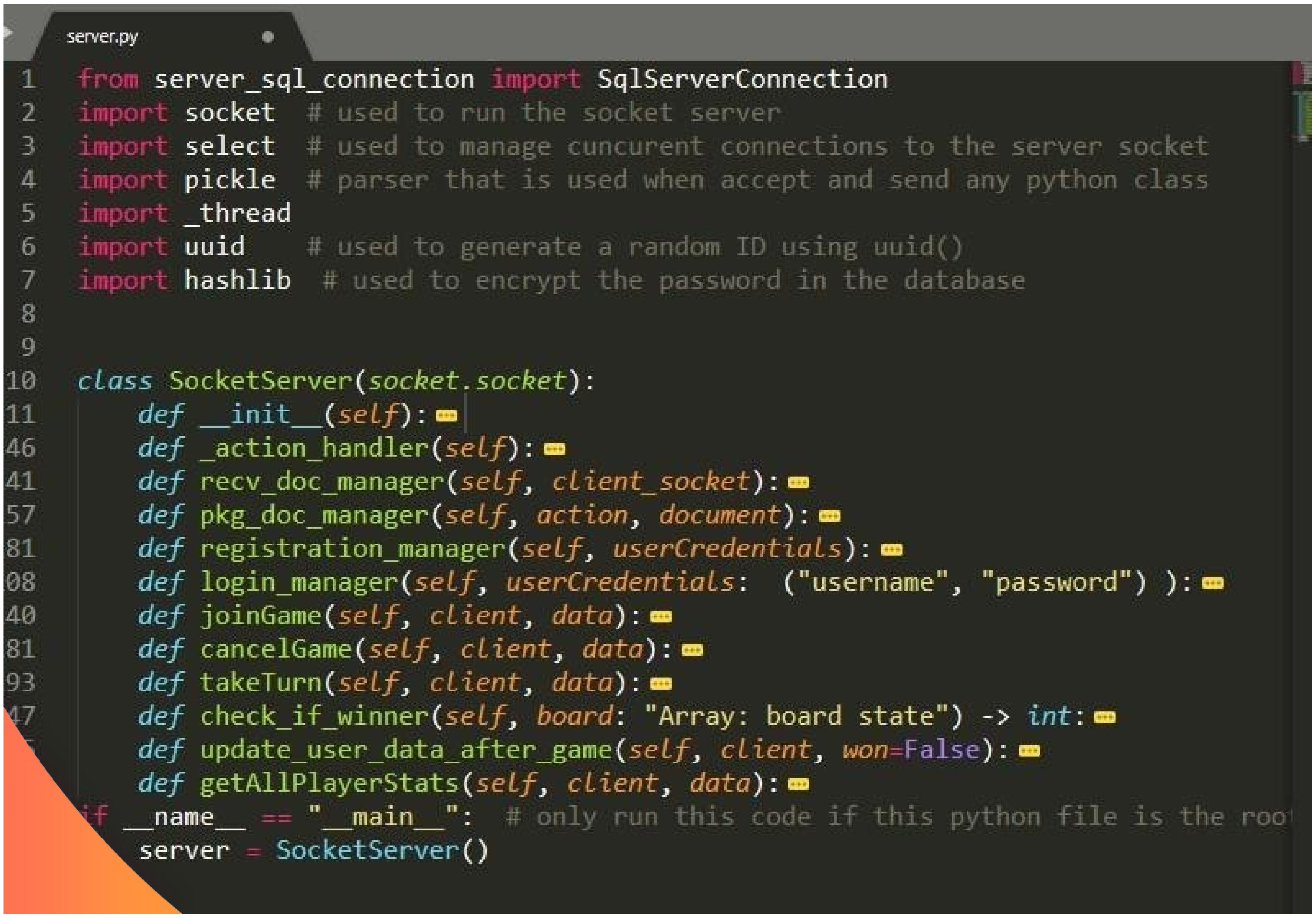
- bg: to set the ordinary foundation tone.
- cursor: to set the cursor utilized in the material.
- highlightColor: to set the shading appeared in the center feature.
- width: to set the width of the gadget.
- stature: to set the tallness of the gadget.

Client-Side Code:



```
import socket  # used to create the client socket connection with the server
import pickle  # parser that is used to accept and send any python class.
class ClientServerSocket(socket.socket):
    def __init__(self, socketHostData=(socket.gethostname(), 4201)):
    async def recv_doc_manager(self):
    def pkg_doc_manager(self, action, document):
    async def login(self, userCredentials):
    async def register(self, userCredentials):
    async def joinGame(self):
    async def cancelGame(self):
    async def startGameLoop(self, frame):
    async def take_turn(self, rowCol):
    async def getAllPlayerData(self):
    def insertion_sort(self, userStats):
if __name__ == "__main__":  # only run this code if this python file is the
```

- **Server-Side Code:**



server.py

```
from server_sql_connection import SqlServerConnection
import socket  # used to run the socket server
import select  # used to manage cuncurent connections to the server socket
import pickle  # parser that is used when accept and send any python class
import _thread
import uuid    # used to generate a random ID using uuid()
import hashlib  # used to encrypt the password in the database


class SocketServer(socket.socket):
    def __init__(self): ...
    def _action_handler(self): ...
    def recv_doc_manager(self, client_socket): ...
    def pkg_doc_manager(self, action, document): ...
    def registration_manager(self, userCredentials): ...
    def login_manager(self, userCredentials:  ("username", "password") ): ...
    def joinGame(self, client, data): ...
    def cancelGame(self, client, data): ...
    def takeTurn(self, client, data): ...
    def check_if_winner(self, board: "Array: board state") -> int: ...
    def update_user_data_after_game(self, client, won=False): ...
    def getAllPlayerStats(self, client, data): ...
if __name__ == "__main__":  # only run this code if this python file is the roo
    server = SocketServer()
```

GAME LOGIC USING ALPHA-BETA PRUNING:

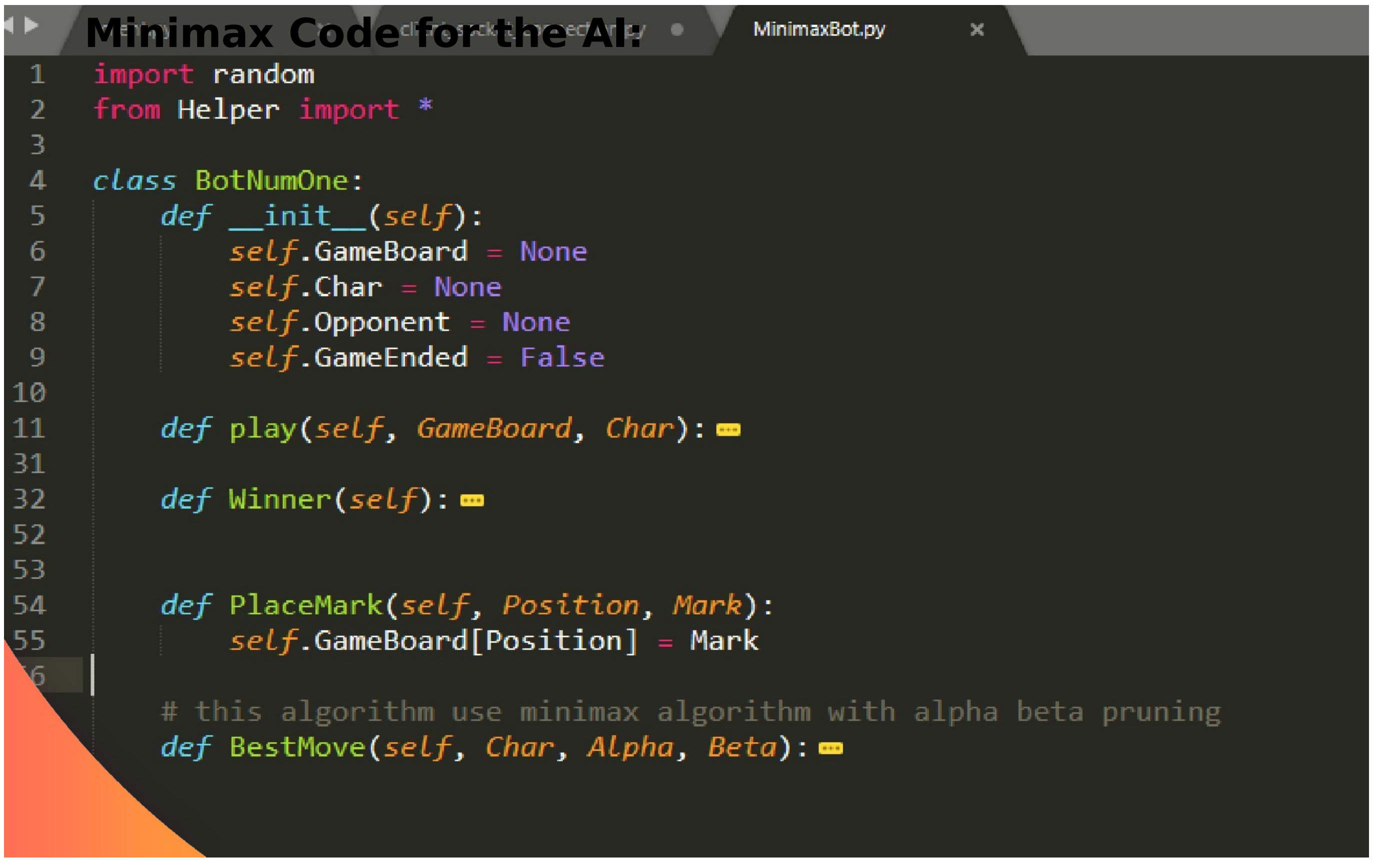
Alpha-beta pruning is a changed form of the minimax calculation. It is an advancement method for the minimax calculation using boundaries Alpha and Beta.

It lessens the calculation time by a colossal factor. This permits us to look through a lot quicker and surprisingly go into more profound levels in the game tree. It cuts off branches in the game tree which need not be looked at on the grounds that there as of now exists a superior move accessible. It is called Alpha-Beta pruning since it passes 2 additional boundaries in the minimax work, specifically alpha and beta.

Alpha is the best worth that the maximizer as of now can ensure at that level or above. Beta is the best worth that the minimizer as of now can ensure at that level or above.

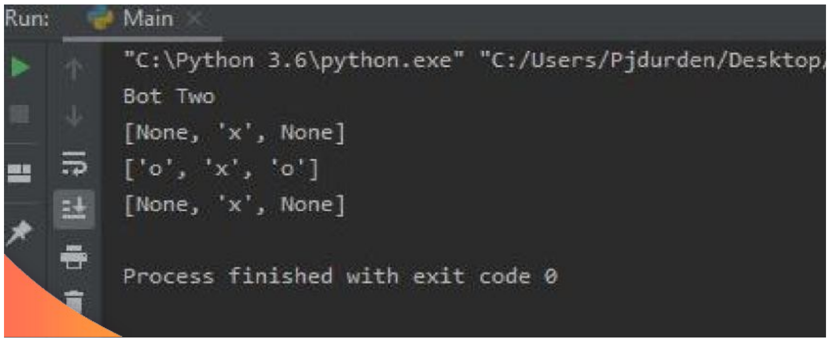
The advantage of alpha–beta pruning lies in the way that parts of the hunt tree can be wiped out. Thusly, the pursuit time can be restricted to the 'more encouraging' subtree, and a more profound inquiry can be acted in a similar time. Like its archetype, it has a place with the branch and bound class of calculations. The enhancement lessens the powerful profundity to somewhat the greater part that of basic minimax if the hubs are assessed in an ideal or close to ideal request (most ideal decision for side on move requested first at every hub).

Minimax Code for the AI:



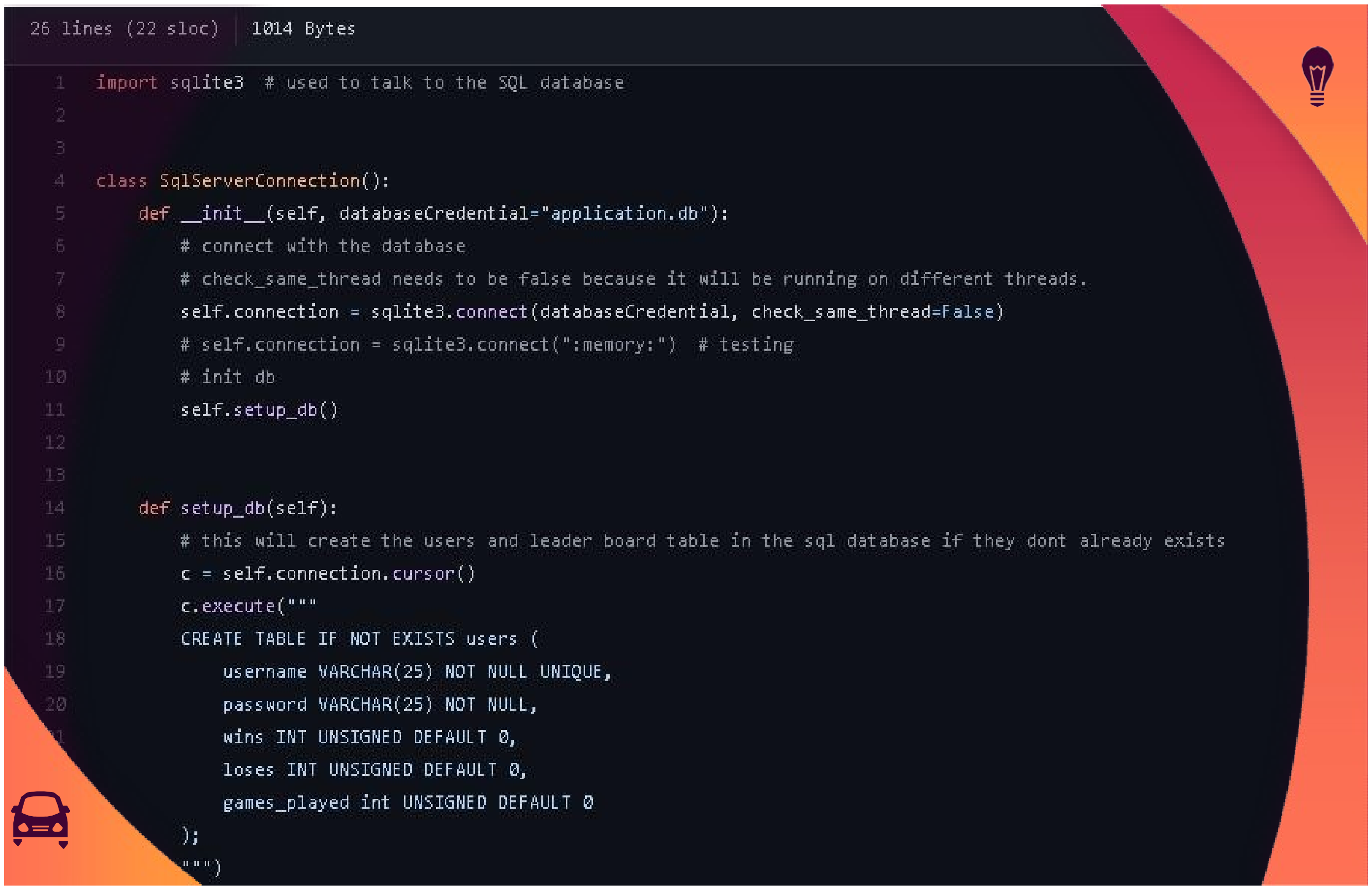
```python
import random
from Helper import *

class BotNumOne:
    def __init__(self):
        self.GameBoard = None
        self.Char = None
        self.Opponent = None
        self.GameEnded = False

    def play(self, GameBoard, Char):

    def Winner(self):


    def PlaceMark(self, Position, Mark):
        self.GameBoard[Position] = Mark

    # this algorithm use minimax algorithm with alpha beta pruning
    def BestMove(self, Char, Alpha, Beta):
```

AI Judge Output:



```
"C:\Python 3.6\python.exe" "C:/Users/Pjdurden/Desktop/
Bot Two
[None, 'x', None]
['o', 'x', 'o']
[None, 'x', None]

Process finished with exit code 0
```

SQLite Implementation:



```
import sqlite3  # used to talk to the SQL database


class SqlServerConnection():
    def __init__(self, databaseCredential="application.db"):
        # connect with the database
        # check_same_thread needs to be false because it will be running on different threads.
        self.connection = sqlite3.connect(databaseCredential, check_same_thread=False)
        # self.connection = sqlite3.connect(":memory:")  # testing
        # init db
        self.setup_db()


    def setup_db(self):
        # this will create the users and leader board table in the sql database if they dont already exists
        c = self.connection.cursor()
        c.execute("""
        CREATE TABLE IF NOT EXISTS users (
            username VARCHAR(25) NOT NULL UNIQUE,
            password VARCHAR(25) NOT NULL,
            wins INT UNSIGNED DEFAULT 0,
            loses INT UNSIGNED DEFAULT 0,
            games_played int UNSIGNED DEFAULT 0
        );
        """)
```

Bibliography:

1. http://en.wikipedia.org/wiki/Berkeley sockets.

2. “Computer Networking: A Top-Down Approach Featuring the Internet,”

by James F. Kurose and Keith W. Ross.

3. “Java Socket Programming” Joseph M. Dibella,

4. (http://www.tutorialspoint.com/java/java networking.htm)

5. (http://www.tutorialspoint.com/java/java networking

6. Rajkumar Buyya , “Lesson: All about Sockets” from the

Java Tutorials. “Socket Programming,”