WT Research Project

Tic Tac Toe Using Socket Programming

Made by-

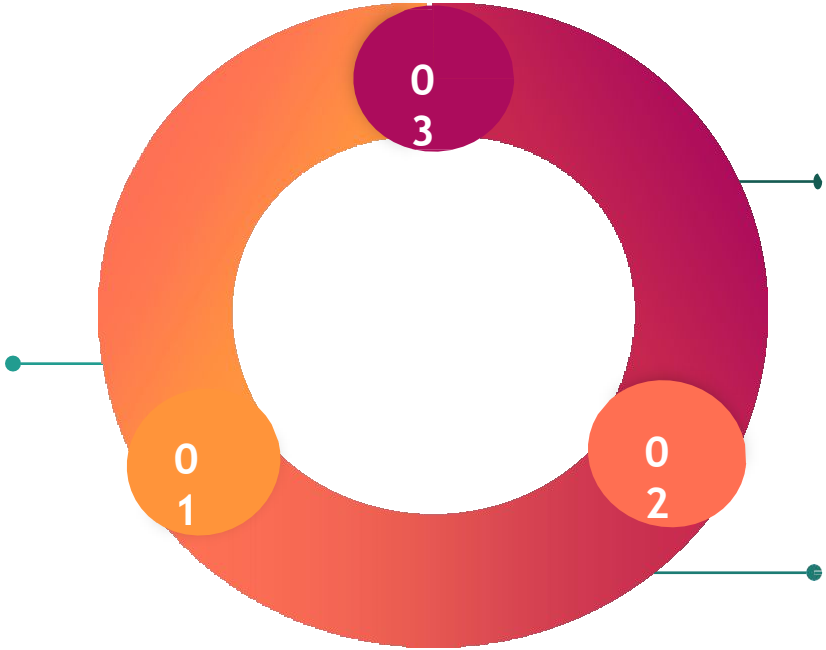
Prajjwal Chittori(2K18/CO/249)

AIM

Building a Tic Tac Toe multiplayer Game using Client Server network Interface of Python Socket.

Also implementing AI playerusing Alpha Beta Pruning algorithm.

Application’s Architechture-



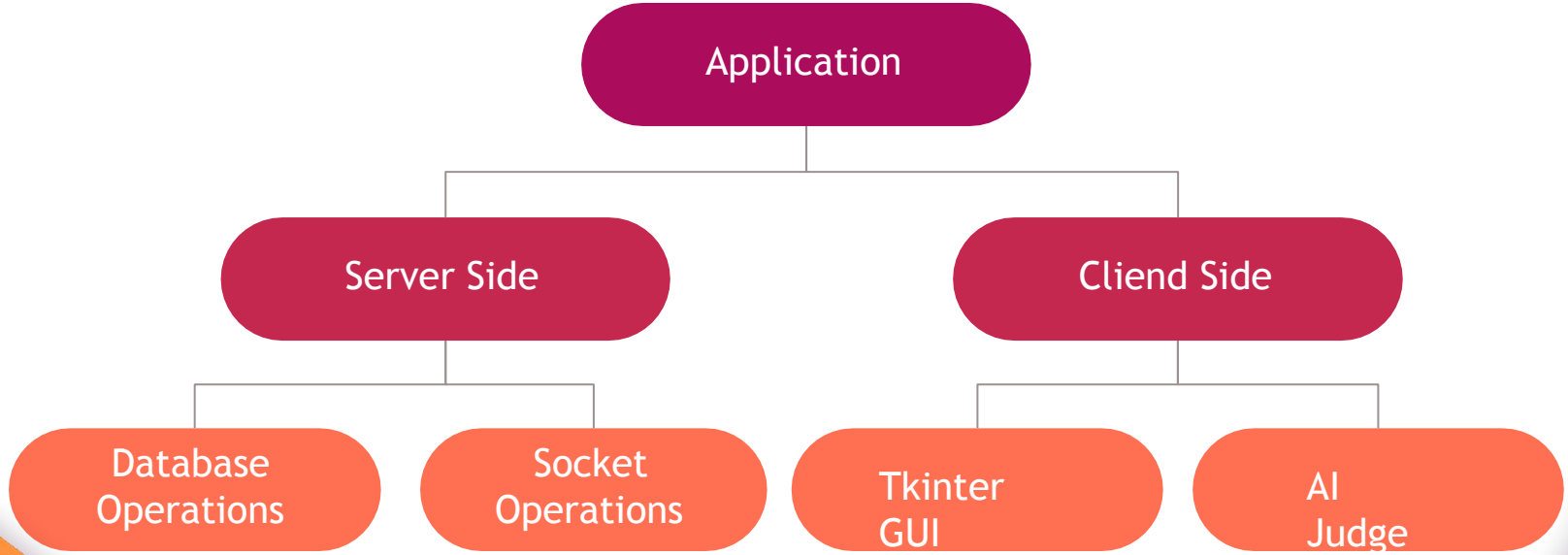
GUI

Tkinter Module

AI
Judge
Alpha Beta Pruning.

Server-Client
Interface
Python Socket Module

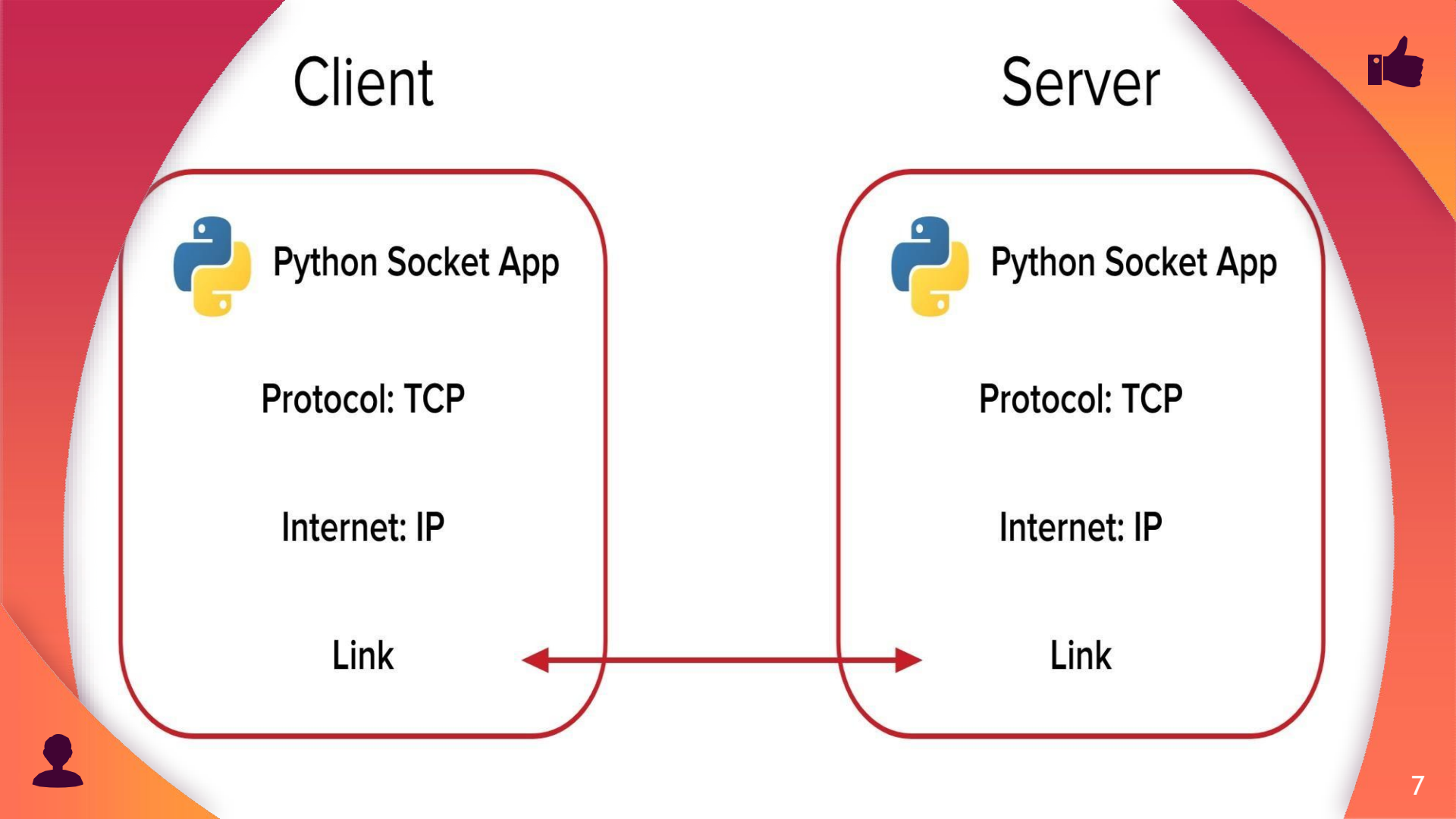
Application Breakdown



Socket Programming & Socket Module

“

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection.


Client
Server
Python Socket App
Protocol: TCP
Internet: IP
Link
Python Socket App
Protocol: TCP
Internet: IP
Link

Socket() Module

- We have used following socket methods to create server and client for the application.
  - **connect()**
  - **bind( )**
  - **listen(backlog)**
  - **accept( )**
  - s.recv()

server.py

```
1   from server_sql_connection import SqlServerConnection
2   import socket  # used to run the socket server
3   import select  # used to manage cuncurent connections to the server so
4   import pickle  # parser that is used when accept and send any python cl
5   import _thread
6   import uuid    # used to generate a random ID using uuid()
7   import hashlib  # used to encrypt the password in the database
8
9
10  class SocketServer(socket.socket):
11      def __init__(self): ...
46      def _action_handler(self): ...
41      def recv_doc_manager(self, client_socket): ...
57      def pkg_doc_manager(self, action, document): ...
81      def registration_manager(self, userCredentials): ...
08      def login_manager(self, userCredentials:  ("username", "password") ): ...
40      def joinGame(self, client, data): ...
81      def cancelGame(self, client, data): ...
93      def takeTurn(self, client, data): ...
47      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
    server = SocketServer()
```

client.py | client_socket_connection.py

```
1   import socket  # used to create the client socket connection with
2   import pickle  # parser that is used to accept and send any python
3   class ClientServerSocket(socket.socket):
4       def __init__(self, socketHostData=(socket.gethostname(), 4201)):...
33      async def recv_doc_manager(self):...
52      def pkg_doc_manager(self, action, document):...
72      async def login(self, userCredentials):...
94      async def register(self, userCredentials):...
115     async def joinGame(self):...
155     async def cancelGame(self):...
164     async def startGameLoop(self, frame):...
201     async def take_turn(self, rowCol):...
216     async def getAllPlayerData(self):...
239     def insertion_sort(self, userStats):...
    if __name__ == "__main__":  # only run this code if this python file is t
```

Implemented Account Registeration and Login

Using SQLite DB*

Skip to slide 23 for SQLite

* For saving and retrieving user data

2

Authentication

Username: prajjwal

Password: *********

Server Info

Host name: DESKTOP-KK9I1C5

Port number: 4201

Login

Register

Account was created successfully.

Tkinter Module

Used to implement the GUI of the application

Tkinter is the standard Python interface to the Tk GUI toolkit shipped with Python.

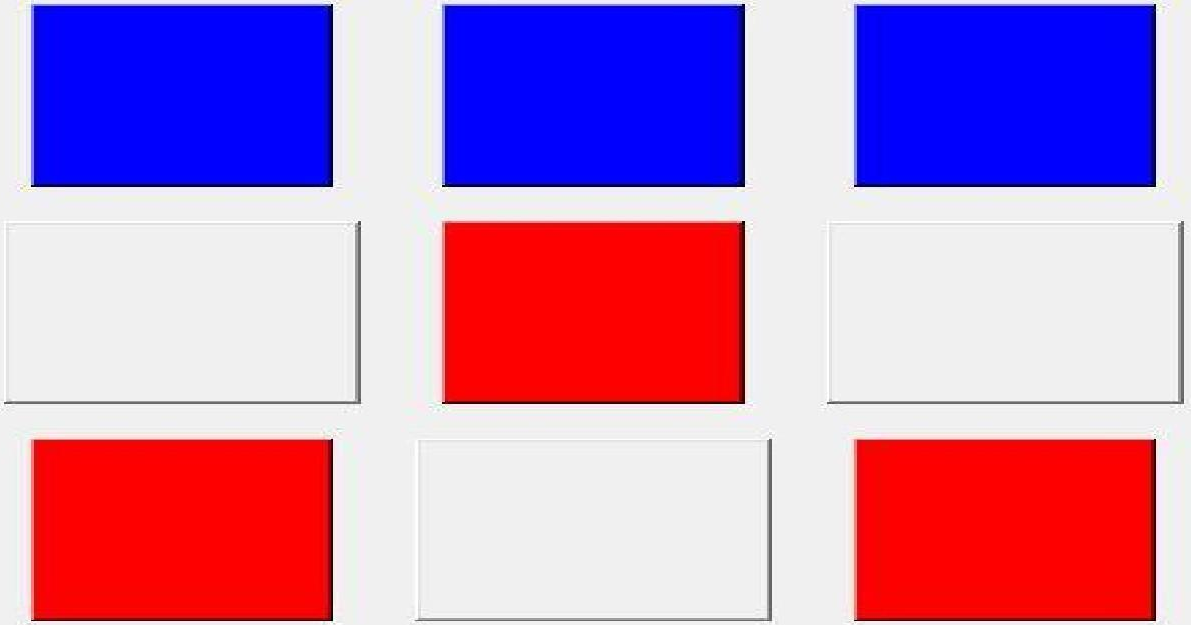
Tkinter offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes

Tkinter Module

- We have used following Tkinter Methods to create GUI-
  - **Tk(screenName=None, baseName=None, className='Tk', useTk=1)**
  - **mainloop()**
  - **pack()**
  - **grid()**
  - place()

client.py

```
1   import tkinter as tk  # Ui builder
2   from tkinter import ttk # leaderboard UI builder
3   import asyncio
4   # the client socket controller that is used by the ui/client, this will
5   from client_socket_connection import ClientServerSocket
6   # used to create multiple threads (in my case allow the client to wait to
7   import _thread
8   class Application(tk.Tk):...
29  class HomePage(tk.Frame):  # inherit from the tk frame...
94  class JoinGamePage(tk.Frame):  # inherit from the tk frame...
27  class GamePage(tk.Frame):  # inherit from the tk frame...
03  class LeaderBoardPage(tk.Frame):  # inherit from the tk frame...
    class AuthenticationPage(tk.Frame):  # inherit from the tk frame...
    if __name__ == "__main__":  # only run this code if this python file is the
```



Player 1: prajjwal
Player 2: sssss
Turn: Player 1

? has won!

Go Back to home

AI Player

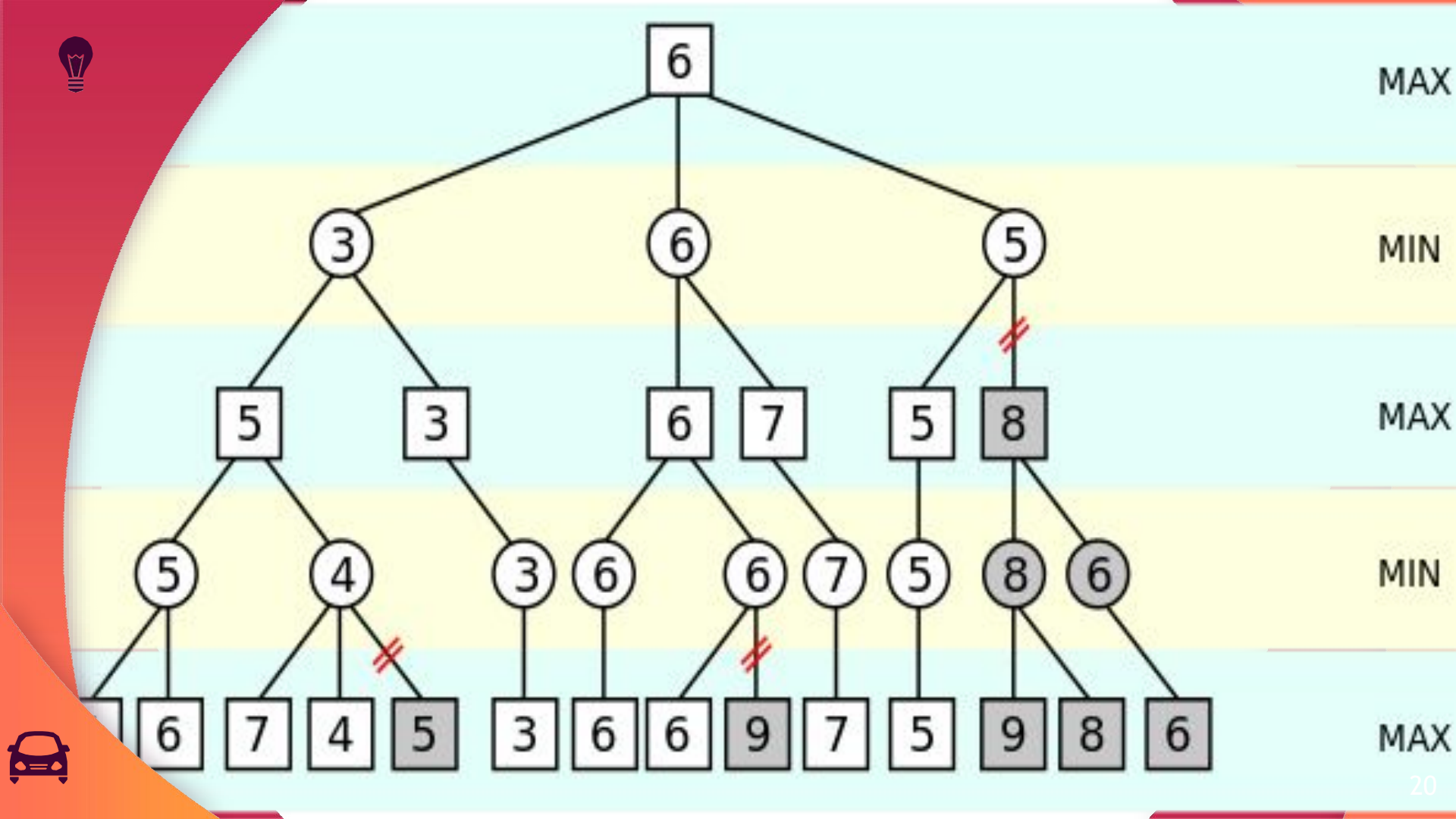
Implemented AI player using Alpha Beta Pruning

“

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm utilizing parameters Alpha and Beta.

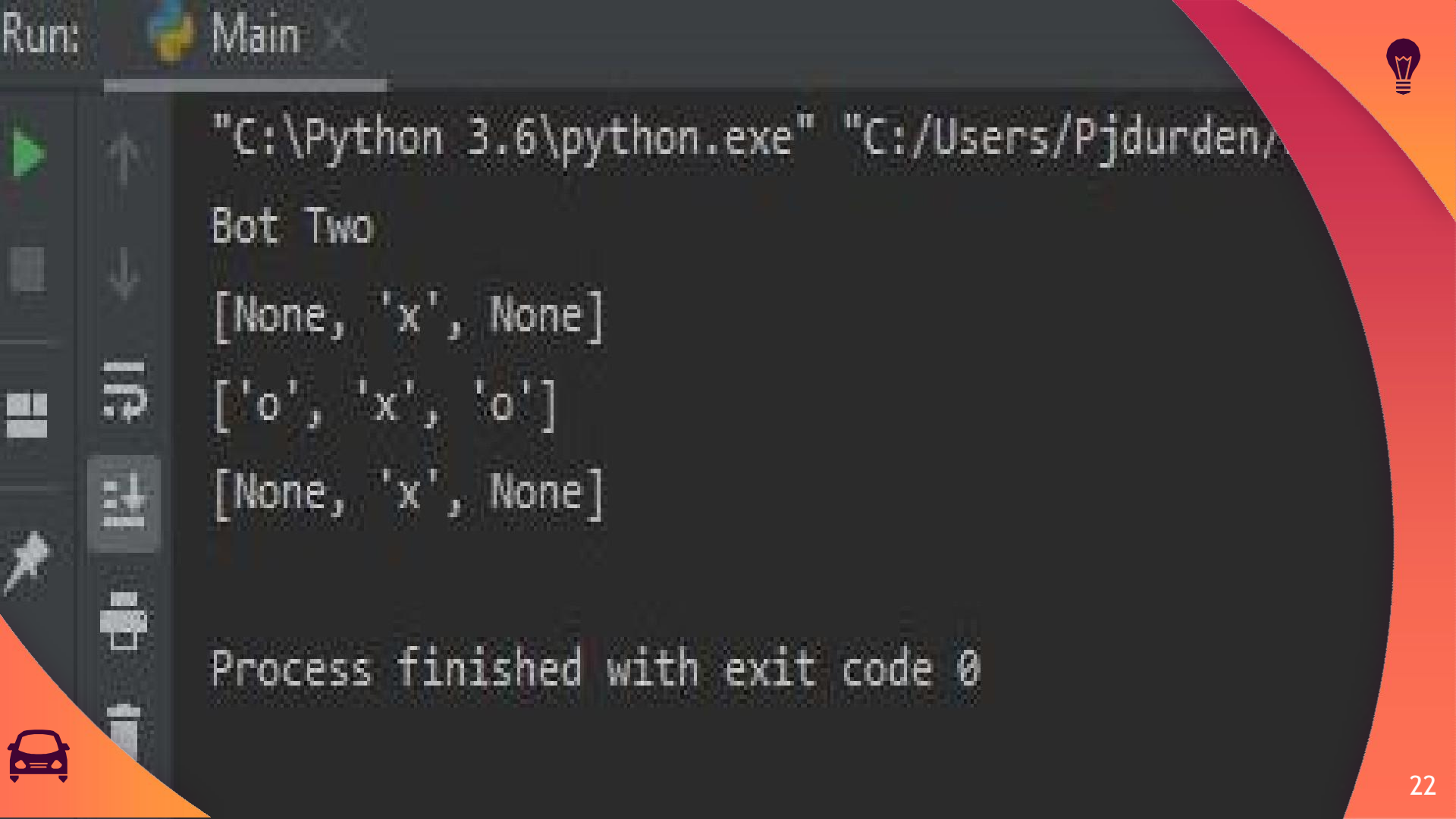
The two parameters are defined as-

- Alpha: The best (highest-value)
- Beta: The best (lowest-value)


6
3
6
5
5
3
6
7
5
8
5
4
3
6
6
7
5
8
6
6
7
4
5
3
6
6
9
7
5
9
8
6
MAX
MIN
MAX
MIN
MAX

client.py | client_socket_connection.py | MinimaxBot.py

```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): ...
```


Run:
Main
"C:\Python 3.6\python.exe" "C:/Users/Pjdurden/
Bot Two
[None, 'x', None]
['o', 'x', 'o']
[None, 'x', None]
Process finished with exit code 0

Implementing DB

Used to save Game Statistics and save User Data

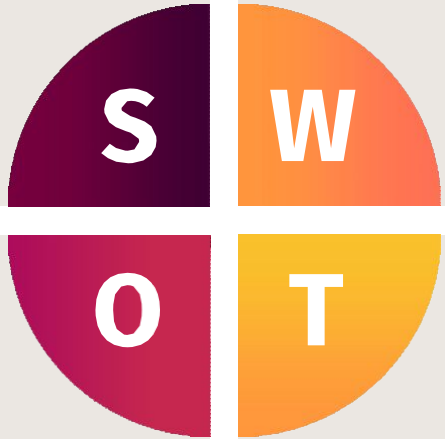
SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

The Python Standard Library includes a module called "sqlite3" intended for working with this database

26 lines (22 sloc) | 1014 Bytes

```python
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
        );
        """)
```

SWOT Analysis



STRENGTHS

The connection Limit of Socket isdependent only on hardware, no software limit.

WEAKNESSES

Tkinter provides primitive GUI features. AI Judge consumes time on slow hardware

Multiuser interactivity of database by using RDBMS

IMPROVEMENTS

Login and User info(hashlib) is saved in DB which can be vulnerable.

THREATS

THANK YOU

END