SHENKAR שנקר

ENGINEERING. DESIGN. ART. הנדסה. עיצוב. אמנות.

פרויקט סיום – סדנא ברשתות תקשורת

Name :Morad Zubidat - [redacted]

Report:

Overview:

This Python script implements a simple server-client game using socket programming. The game is a variation of the classic Connect Four game, where two players take turns to drop pieces into a grid and attempt to connect four of their pieces in a row, either horizontally, vertically, or diagonally.

Server Side

Classes:

1. Game Server:

- This class represents the server that manages client connections and facilitates communication between clients.

Attributes:

- clients: A dictionary that maps client names to their socket and address.

- reverse_lookup: A dictionary that maps sockets to client names.

- direct_communications: A dictionary that maps client names to their direct communication partner.

- host: The IP address on which the server listens for connections (default is '127.0.0.1').

- por: The port on which the server listens for connections (default is 12345).

- sock: The server socket object.

Methods:

- __init__(self, host='127.0.0.1', port=12345): Initializes the server with the specified host and port.

- broadcast_clients(self, exclude=None): Sends a list of currently connected clients to all clients, excluding the specified client.

- handle_client(self, client, address): Handles communication with a client.

- run(self): Starts the server and listens for client connections.

פרויקט סיום – סדנא ברשתות תקשורת

Functions:

1. initialize_game():

   - Initializes the game board as a 7x6 grid with all slots set to `None`.

2. **print_game(game):**

   - Prints the current state of the game board to the console.

3. **make_move(game, stack, player):**

   - Allows a player to make a move by dropping a piece into a specified stack.
   - Returns **True** if the move is valid and **False** if the stack is full.

4. **check_win(game):**

   - Checks the game board for a winning condition (four pieces in a row).
   - Returns the winning player's number (**0** or **1**) if there is a winner, otherwise returns `None`.

Execution:

The __main__ block creates an instance of the **ChatServer** class and starts the server by calling the **run()** method.

Game Flow:

- Clients connect to the server and choose whether to play with the server or another client.

- If playing with the server, clients take turns making moves by selecting a stack to drop their piece into.

- If playing with another client, the server facilitates direct communication between the two clients, allowing them to play the game in real-time.

Note

- The server handles multiple client connections concurrently using threading.

- The game logic for player moves, win conditions, and turn switching is implemented within the handle_client() method.

- Clients communicate with the server using simple text-based messages over TCP/IP sockets.

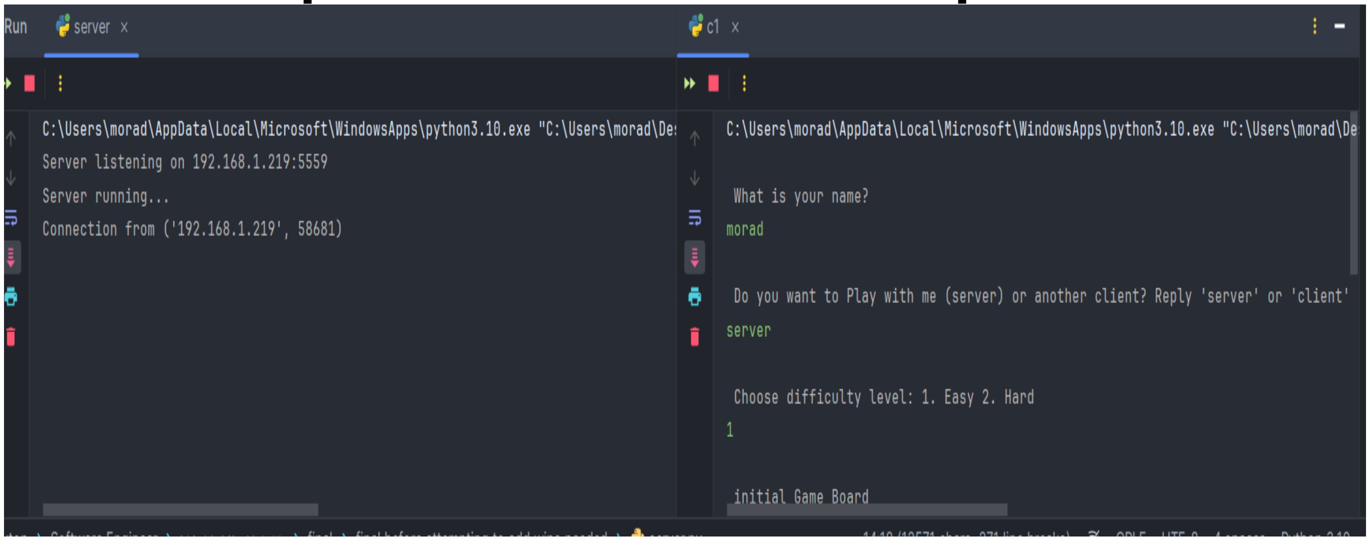
Client Side:

- **Game Initialization and Display:** The game board is initialized as a 7x6 grid (7 columns and 6 rows), where moves are represented by placing a player's marker in one of the columns. The board state is displayed in the console, with '-' indicating an empty space.
- **Making Moves:** Players take turns to make moves by sending the column number (0-based indexing) in which they wish to place their marker. The game checks if the move is valid (i.e., the chosen column is not full) and updates the board accordingly.
- **Winning Condition:** The game checks for a win condition after every move. A win is declared if a player manages to align four of their markers vertically in any column. Currently, the code only checks for vertical wins and does not consider horizontal or diagonal alignments.
- **Network Communication:** The game uses TCP/IP sockets for network communication. It connects to a server (expected to run at 127.0.0.1 on port 12345) and sends messages corresponding to player moves. It also listens for messages from the server in a separate thread, which could represent game state updates, moves made by the opponent, or other commands.
- **Error Handling and Game Flow:** Basic error handling is implemented, catching exceptions that might occur during network communication or input handling. The game loop allows for sending messages (moves) to the server and displays the updated board state after each move. The game ends if the player types 'exit', an error occurs, or the game reaches a draw condition (all positions filled without a winner).

Player vs Server

SHENKAR — שנקר
ENGINEERING. DESIGN. ART. הנדסה. עיצוב. אמנות.

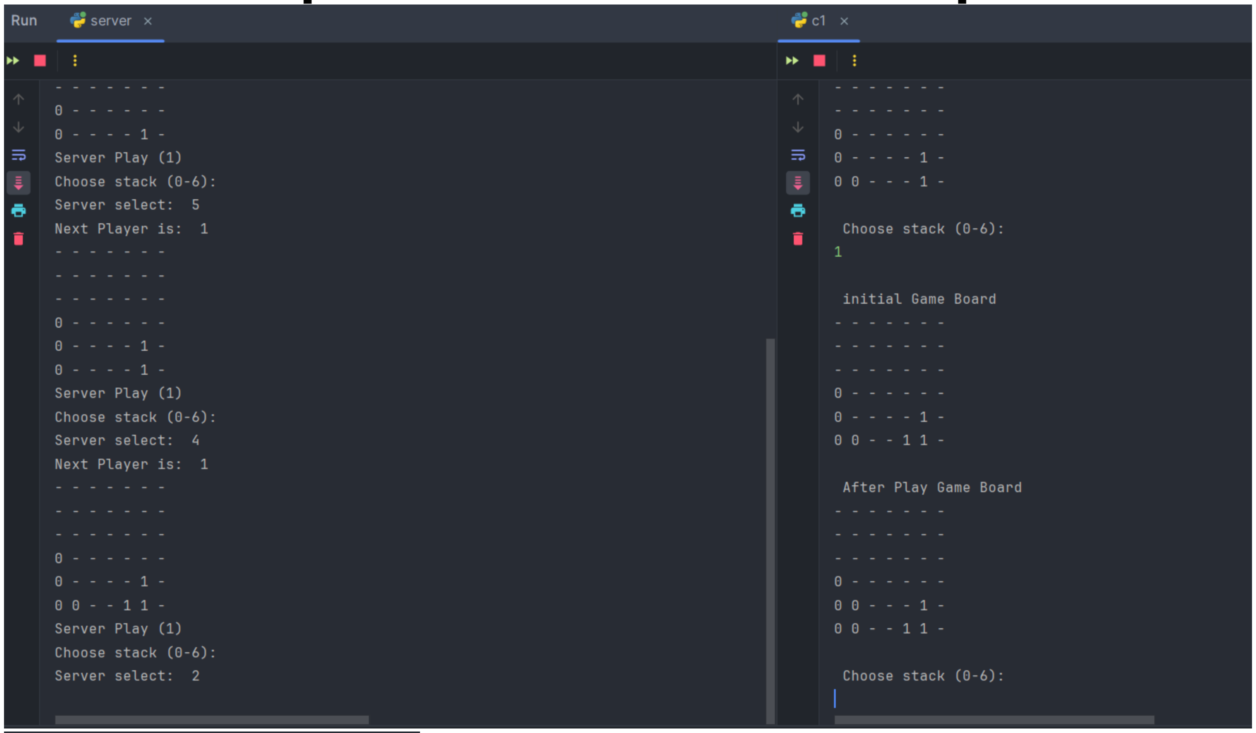
פרויקט סיום – סדנא ברשתות תקשורת



```
C:\Users\morad\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\morad\De
Server listening on 192.168.1.219:5559
Server running...
Connection from ('192.168.1.219', 58681)
```

```
C:\Users\morad\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\morad\De

 What is your name?
morad

 Do you want to Play with me (server) or another client? Reply 'server' or 'client'
server

 Choose difficulty level: 1. Easy 2. Hard
1

 initial Game Board
```

פרויקט סיום – סדנא ברשתות תקשורת

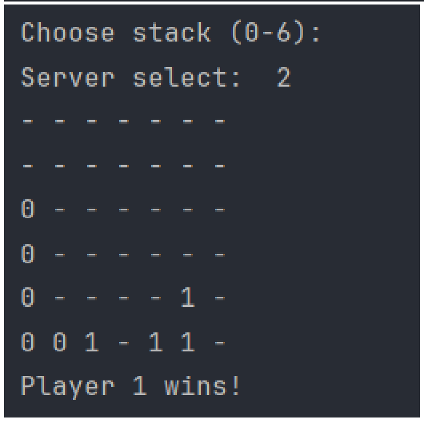


Run server ×

```
- - - - - - -
0 - - - - - -
0 - - - - 1 -
Server Play (1)
Choose stack (0-6):
Server select:  5
Next Player is:  1
- - - - - - -
- - - - - - -
- - - - - - -
0 - - - - - -
0 - - - - 1 -
0 - - - - 1 -
Server Play (1)
Choose stack (0-6):
Server select:  4
Next Player is:  1
- - - - - - -
- - - - - - -
- - - - - - -
0 - - - - - -
0 - - - - 1 -
0 0 - - 1 1 -
Server Play (1)
Choose stack (0-6):
Server select:  2
```

c1 ×

```
- - - - - - -
- - - - - - -
0 - - - - - -
0 - - - - 1 -
0 0 - - - 1 -

 Choose stack (0-6):
1

 initial Game Board
- - - - - - -
- - - - - - -
- - - - - - -
0 - - - - - -
0 - - - - 1 -
0 0 - - 1 1 -

 After Play Game Board
- - - - - - -
- - - - - - -
- - - - - - -
0 - - - - - -
0 0 - - - 1 -
0 0 - - 1 1 -

 Choose stack (0-6):
```



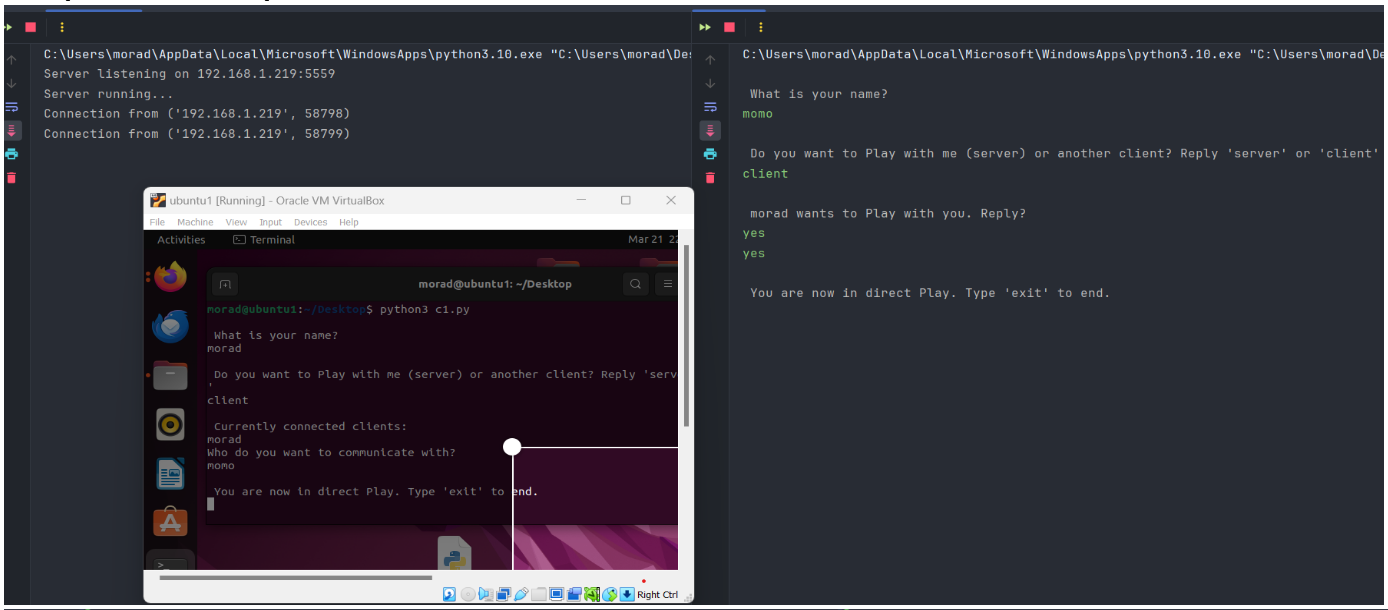
```
Choose stack (0-6):
Server select:  2
- - - - - - -
- - - - - - -
0 - - - - - -
0 - - - - - -
0 - - - - 1 -
0 0 1 - 1 1 -
Player 1 wins!
```

Server Side notify Player 1 Wins

SHENKAR — שנקר
ENGINEERING. DESIGN. ART. הנדסה. עיצוב. אמנות

פרויקט סיום – סדנא ברשתות תקשורת

Player1 vs Player2 :

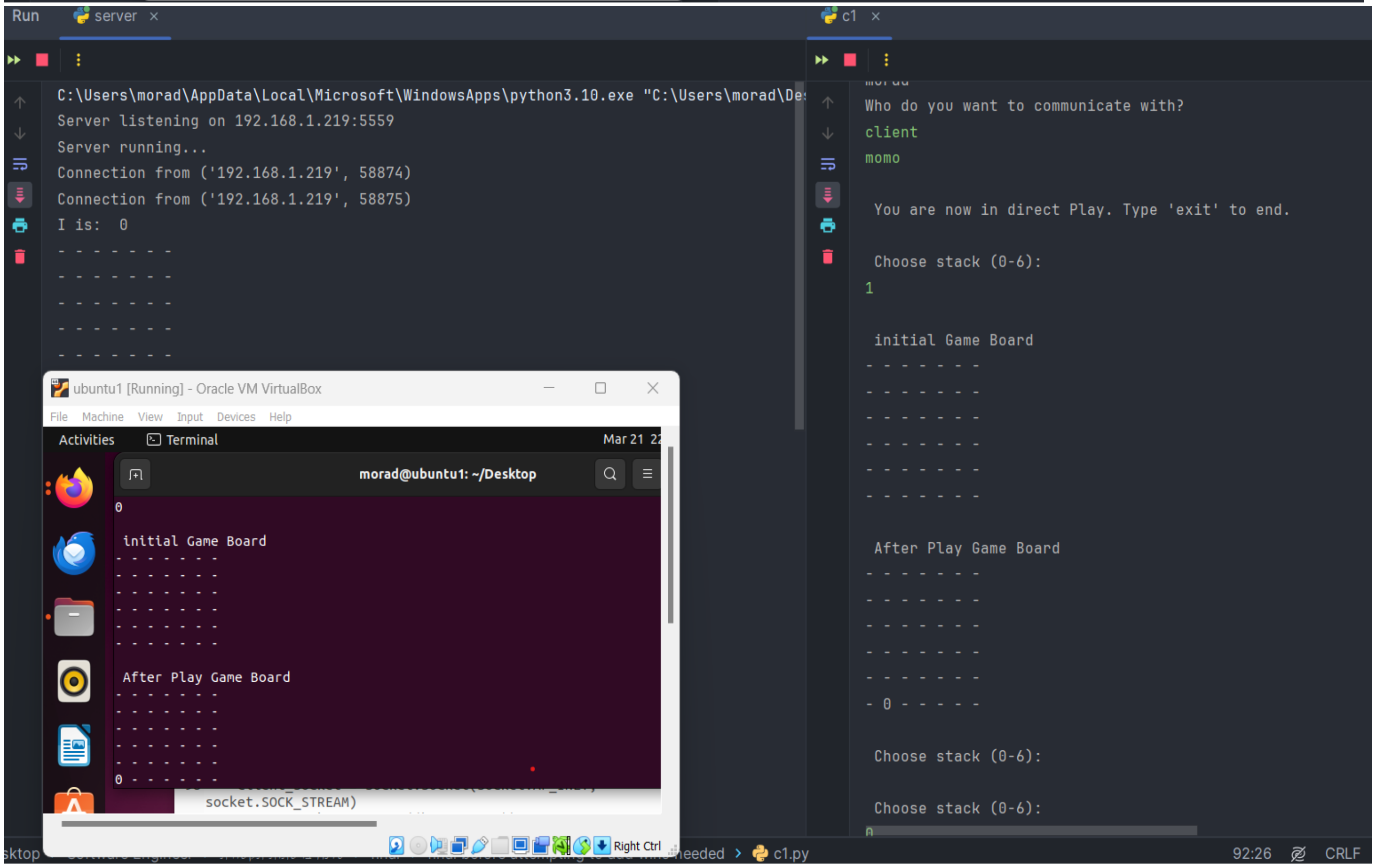


```
C:\Users\morad\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\morad\De
Server listening on 192.168.1.219:5559
Server running...
Connection from ('192.168.1.219', 58798)
Connection from ('192.168.1.219', 58799)
```

ubuntu1 [Running] - Oracle VM VirtualBox

```
morad@ubuntu1:~/Desktop$ python3 c1.py

 What is your name?
morad

 Do you want to Play with me (server) or another client? Reply 'serv
'
client

 Currently connected clients:
morad
Who do you want to communicate with?
momo

 You are now in direct Play. Type 'exit' to end.
```

```
C:\Users\morad\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\morad\De

 What is your name?
momo

 Do you want to Play with me (server) or another client? Reply 'server' or 'client'
client

 morad wants to Play with you. Reply?
yes
yes

 You are now in direct Play. Type 'exit' to end.
```



```
C:\Users\morad\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\morad\De
Server listening on 192.168.1.219:5559
Server running...
Connection from ('192.168.1.219', 58874)
Connection from ('192.168.1.219', 58875)
I is:  0
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
```

ubuntu1 [Running] - Oracle VM VirtualBox

```
0

 initial Game Board
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -

 After Play Game Board
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
0 - - - - - -
```

```
Who do you want to communicate with?
client
momo

 You are now in direct Play. Type 'exit' to end.

 Choose stack (0-6):
1

 initial Game Board
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -

 After Play Game Board
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- 0 - - - - -

 Choose stack (0-6):

 Choose stack (0-6):
0
```

SHENKAR שנקר
ENGINEERING. DESIGN. ART. הנדסה. עיצוב. אמנות.

פרויקט סיום – סדנא ברשתות תקשורת