TicTacToe

Authors: Aril Mavinkere, Bonnie Consiglio, Vanessa Ho
Group Number: 23

Overview

This project was done by Aril Mavinkere, Bonnie Consiglio, and Vanessa Ho. The project contains full functionality of Part 1 of the project, and limited functionality for Part 2 of the project. While the server is built to handle multiple games and the help, games, and who functionalities all work properly, there are numerous bugs when running multiple games that we did not have time to handle. There is no functionality to choose who to play against. Due to time constraints, there is also limited error handling.

How To Run/System Documentation

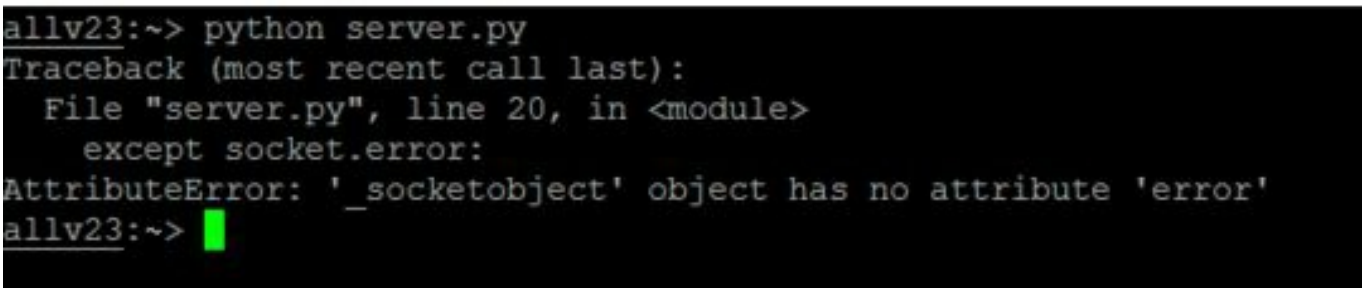
Server Side

To run Tic Tac Toe, upload all files to an allv machine or another Linux server. This can be done with PSFTP or a similar SSH client.
One this is done and you are in the directory of the downloaded files, enter in the command line interface

```
python server.py
```

The server will now be up and running, and it will print “Listening”, as such:

*NOTE- The following error may occur. It can be fixed by simply restarting the SSH client and running the above again.



Client Side

Once the server is up and running, open up another SSH client and navigate to the same directory or a directory where all the required files are present. Upon doing so, enter in the command line interface

```
python client.py hostname
```

*NOTE- hostname is the name of the server in which server.py is running. Entering a host name that is not the same will return an

error.

Classes and Methods

This implementation of TicTacToe consists of five classes.

1. init.py

- This class has no code and does not do anything, but is required for modules to be imported and for the code to compile.

2. server.py

- This class handles all server logic. The main loop of the program can be found at the bottom of the class. It takes client connections, and starts a thread for each new connection with the start_new_thread function, taking the client connection socket and the method that the thread will be running as arguments
- The connect method handles logic for a single client. It handles the user input of each client, and handles game logic, updating player and game info, and sending the appropriate messages back to its client.
- The startGame method is called by the connect class once two clients are matched up. It handles the game loop for the Tic Tac Toe game, and switches between asking each player until the game is finished. It also defines what to send the clients after an end game scenario is reached.

- The checkmove method makes sure a move made by the user is valid before being sent to the game. It returns True when a move is valid, and false when a move is not. If a move is not, the client that input that move is queried for a valid move.

3. client.py

- The client class initiates a connection with the server once the user enters the appropriate command line arguments. It handles sending input from the user to the server, and handling respones from the server to the client.

4. player.py

- The player class represents one player in the game. It contains a constructor for a player, and takes a username and connection socket as arguments. Once a client logs into the server, there is enough information to create a Player object, since that client now has a username and a defined connection between client and server. The player class also has three states; logged in, available, and busy. If a player is logged in, he has connected to the server but is not searching for a game yet. If he is available, he is actively searching for a game. Finally, if he is busy, then he is in game.

5. tttgame.py

- The tttgame.py class contains the constructor for a single instance of a game of Tic Tac Toe. It is constructed every time two clients are looking to play and are matched up. The

TTTGame object takes 3 constructors: player1, player2, and a gameID. player1 and player 2 are Player objects, and gameID is a uniquely generated game id that represents this instance of the game.

- The drawboard method returns a visual representation of the board. It is sent to both players every time they make a move.
- The changeturn method changes whose turn it is. It is called by the startGame method in the server class to switch turns in the game loop.
- The makemove method puts an X or O on the board depending on whose turn it is. It either tells the server after a move is made if the game has ended in a draw, the game has ended in a win, or the move was valid but the game is still in progress. It sends three types of messages, 300 FIN and 300 WIN, and 300 NPT. NPT stands for next player turn.

Types of Messages Sent and Received

1. WAIT Messages

- WAIT messages are sent from the server to the client and indicate that the server is to send more information. It stops the client
  from taking further input momentarily.

2. Normal Messages

- Normal messages are messages sent from the server to the client that after the client receives the message, they are able

to or are expected to take user input and send it back. They comprise the majority of messages sent between client and server.

3. 400 ERR Messages

- 400 ERR messages indicate improper input from the client, and result in the server asking the client to re-enter valid input.

4. DISC Messages

- DISC messages indicate a that the client is requesting to disconnect from the server.

User Documentation

Once a client is running, the following commands are available to you.

1. login (username)

- login allows you to log into an existing account, or create a new one if you don’t have one. It takes one argument,a username.

2. exit

- exit takes no arguments, and allows you to exit the server.

3. play

- play searches for an opponent, and waits until an opponent is found. Upon finding an opponent, it initiates a game with them.

Note that play can only be called once logged in.

4. place (num)

- once in a game and it is indicated that it is your turn, you can play your move by entering place followed by a number between 0 and 8 inclusive.

5. games

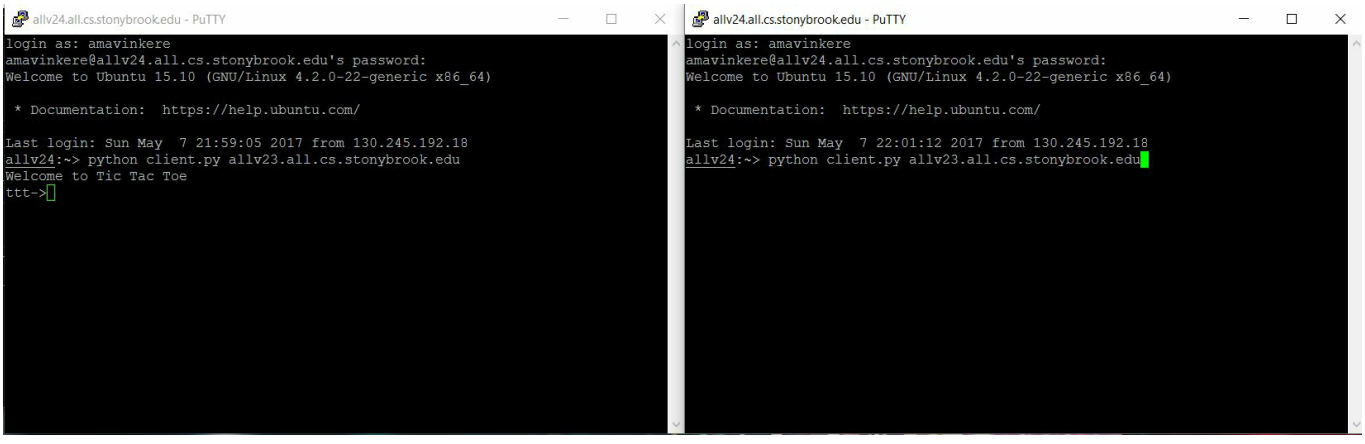
- lists all current games being played

6. who

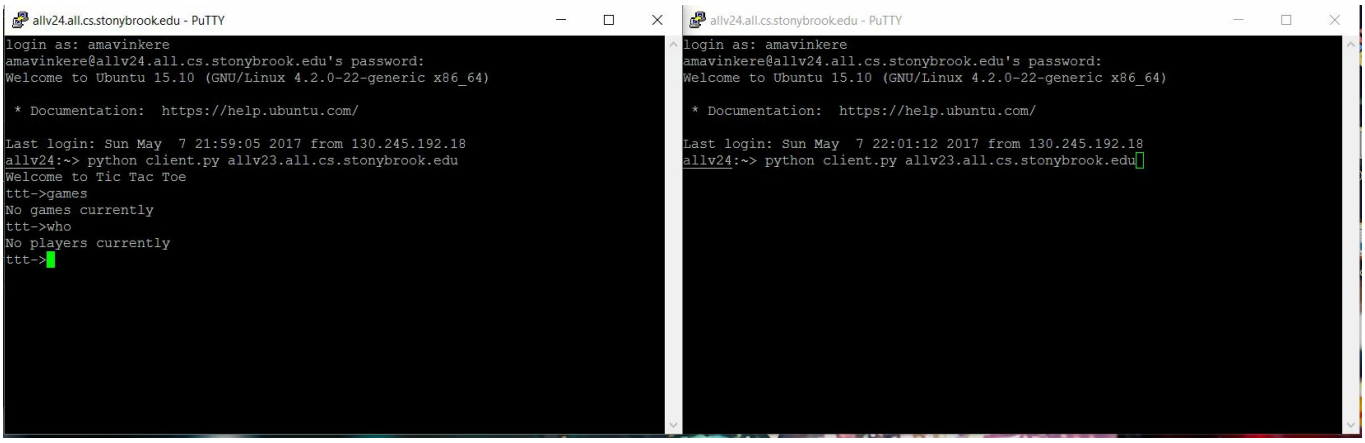
- lists all players currently logged in

Test Documentation

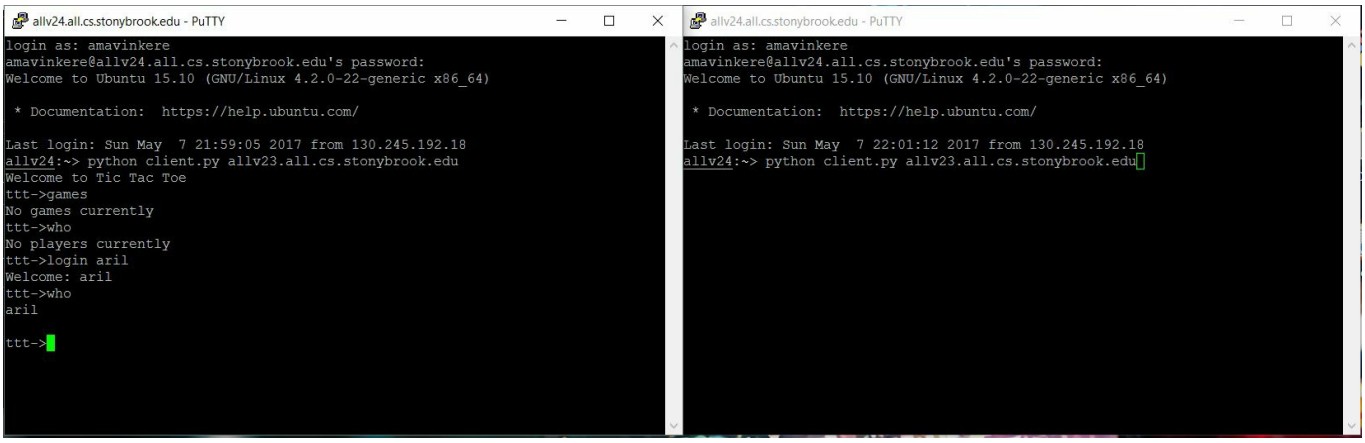
Welcome message when one user connects to the server



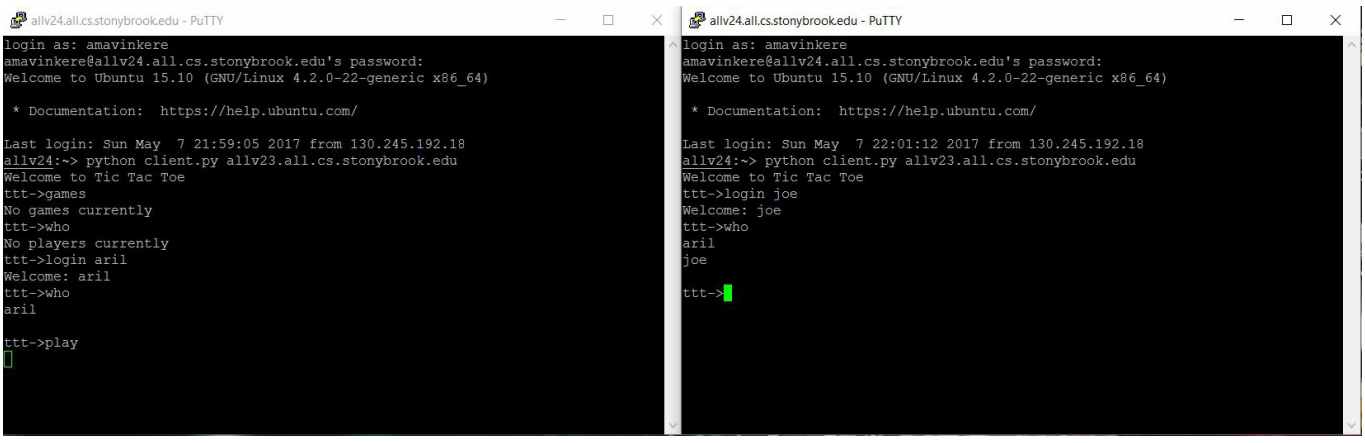
Initial tests of “who” and “games” commands when only one client is connected to the server, but has not logged in yet



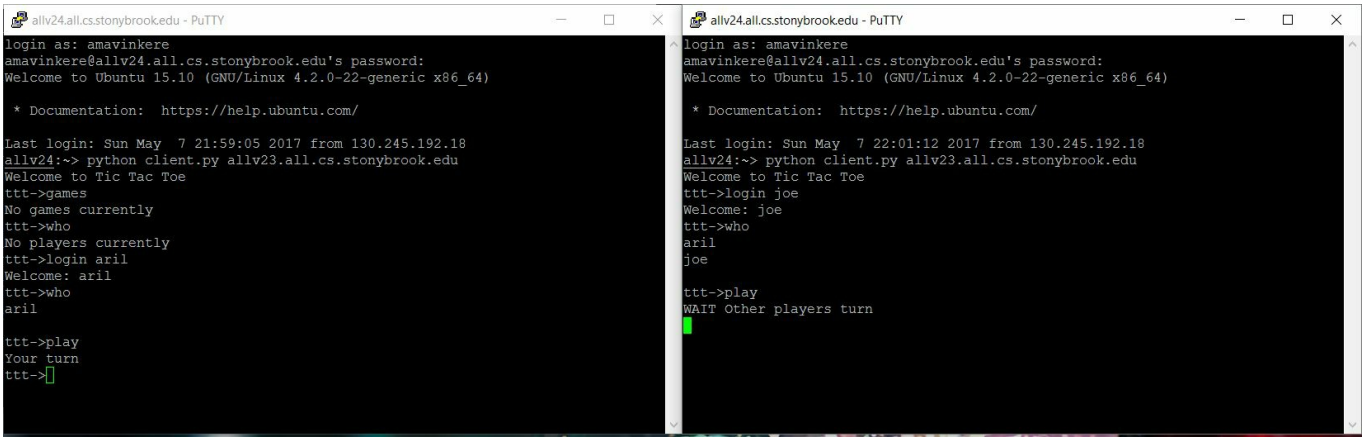
Logging in and testing the “who” command again. This time it returns my username, since I am the only one logged into the server at this time.



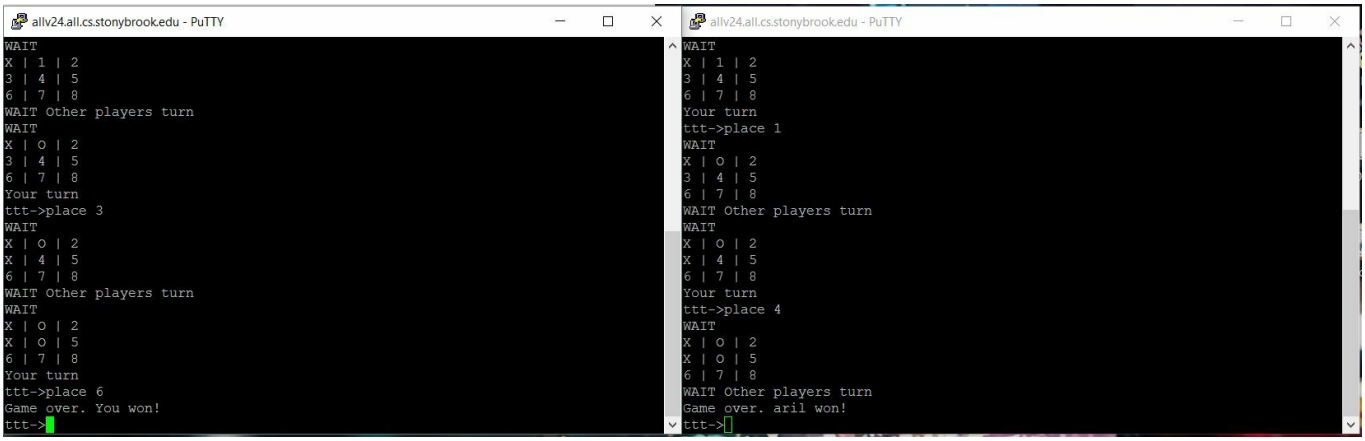
Under a second client (right side of the image), I connect to the server and login under the username “joe”. After running the “who” command now, both aril and joe show up.



Both players initiate a play request, but aril initiates first and joe initiates second. A game is started and aril is allowed to enter a move.



Both players make moves using the place command. After each place command, both players are given an updated version of the board and the player that was previously waiting is allowed to enter a command. Player 1 (on the left) places X's on 0, 3, and 6 and wins the game.



Example of the "games" command with a game underway. The top two command prompts show two users (aril and joe) currently in game. On the bottom left, the server is running and printing information. Once aril and joe enter a game, the server generates a Game ID for that game and prints it on the server side. On the bottom right, a client connects to the server during that game. This client asks to see which games are currently underway with the "games" command, and is

returned the game ID of aril and joe’s game.