Jacqueline Hirsch, Matthew Class, & Scout Froney

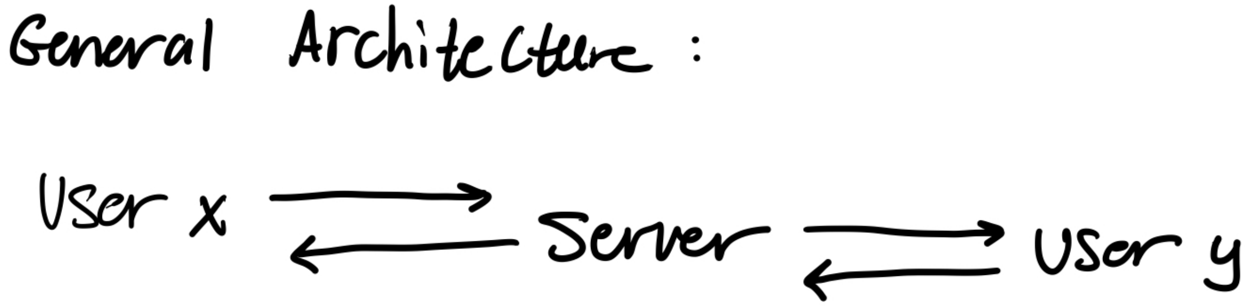
Final Project: Battleship

ECE 470

Description:

The goal of this project is to design and implement a fully functioning two-player game called battleship. The game is multiplayer and supports the two players at the same time of play. This is successful by implementing a multi-user client and server system to support multiple users simultaneously. The client and server architecture will support the two users and allow the users to interact with each other live. In addition to this, the design must allow two users to play battleships with each other from two different devices using a cloud service discussed in class.

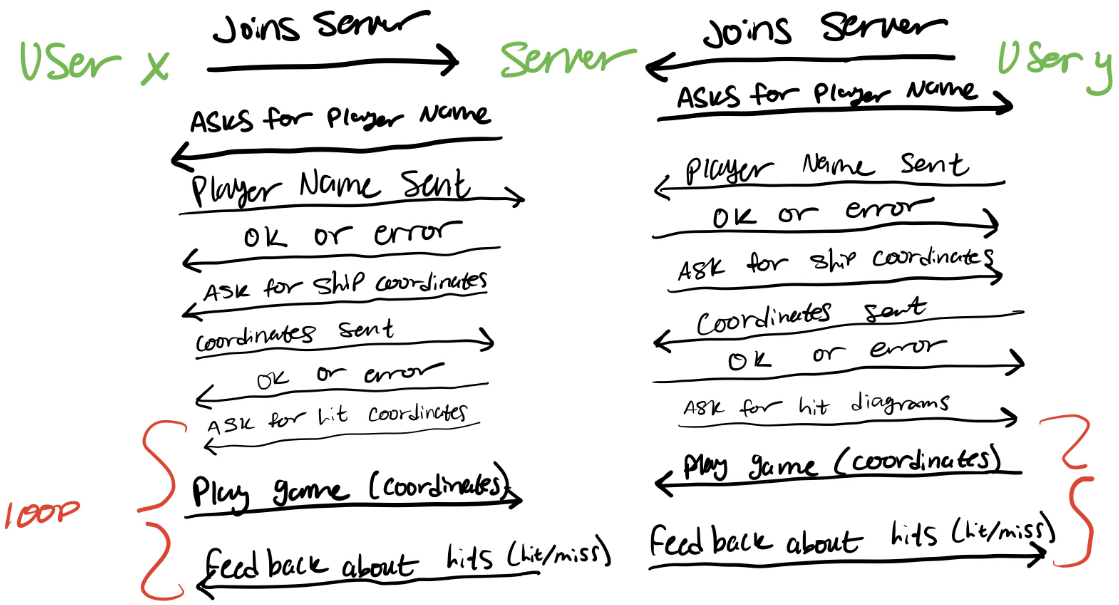
Architecture:



- The architecture of our battleship game supports two players and our communications only allow for one game at a time.
- The architecture of our design is heavy on the server. The server is the part of our design that controls the checks (bad messages/user errors) and keeps track of the game data: hits and misses, ship coordinates.

Storyboard Communication:

Communication Diagram :



- Our communication used TCP. During the game, each user had their own thread on the server. Each client would connect to a thread and communicate with that thread.
- Our communications were powered through Linode using the Ubuntu operating system.

Scenarios:

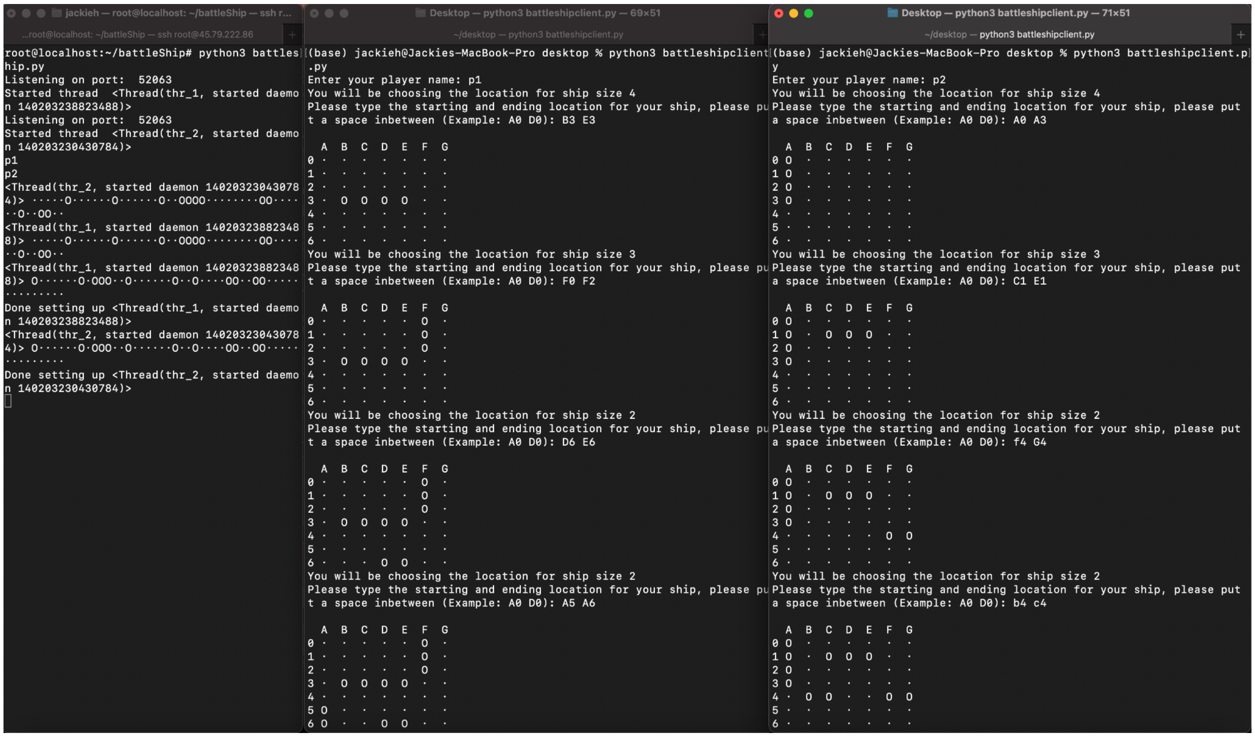
- Join Game:
  - When a player is joining a game there is either one or no players waiting in the game server. If there are no other players in the server when a player joins then the server will wait until another player joins the server to start a battleship game. If there is already one player waiting in the server when another player joins the game then the server will start the game and no other players will be able to connect to the game server.
- Initialize Game:
  - When the game starts the server will ask the two players to give their ship coordinates. As the players give their ship coordinates, the server will check if the ship coordinates are the right sizes for the ships. If they are the right size, the server will continue asking for the next ship coordinates until the board has all of the ship coordinates needed. If the user gives the wrong size with the given coordinates, the server will let the user know that the wrong size was given and to

try again. After the coordinates are given for the ships of both of the users, the game will be initialized and the server will begin the game.

- Hit/Miss:
  - After the game is initialized the server will ask the first player to input coordinates at where they would like to shoot on player two's board. If the coordinate they chose has part of a ship then that coordinate will be considered a hit ('X'). If not, then that coordinate will be considered a miss ('\'). After player one does this, player two will then be asked by the server to give coordinates at where they would like to shoot at. In our code, we had the flaw that the server was not able to successfully check to see if what the user is inputting for a hit/miss coordinate is allowed to be entered. This means users must select an allowed coordinate to not run into error when playing the game.

Verification:

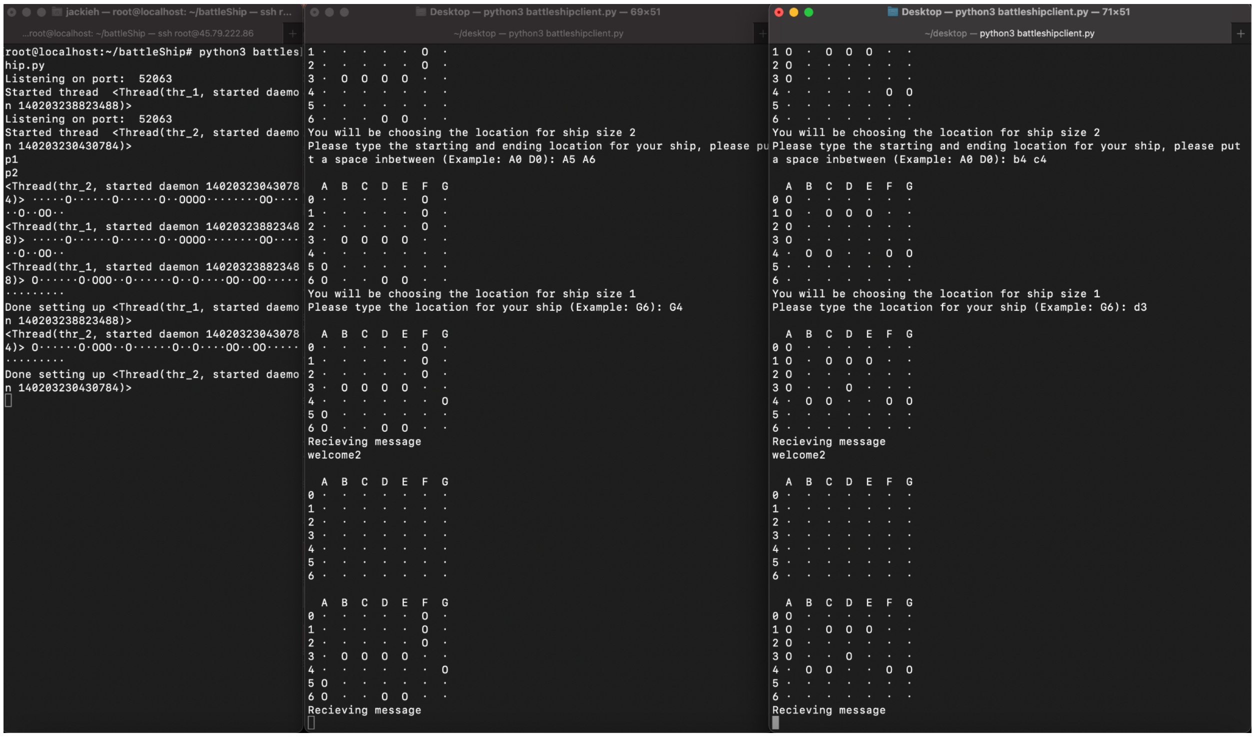
Initializing game:


jackieh — root@localhost: ~/battleShip — ssh r...
...root@localhost: ~/battleShip — ssh root@45.79.222.86
Desktop — python3 battleshipclient.py — 69×51
~/desktop — python3 battleshipclient.py
Desktop — python3 battleshipclient.py — 71×51
~/desktop — python3 battleshipclient.py

```
root@localhost:~/battleShip# python3 battles
hip.py
Listening on port:  52063
Started thread  <Thread(thr_1, started daemo
n 140203238823488)>
Listening on port:  52063
Started thread  <Thread(thr_2, started daemo
n 140203230430784)>
p1
p2
<Thread(thr_2, started daemon 14020323043078
4)> ·····O······O······O··OOOO········OO····
··O··OO··
<Thread(thr_1, started daemon 14020323882348
8)> ·····O······O······O··OOOO········OO····
··O··OO··
<Thread(thr_1, started daemon 14020323882348
8)> O······O·OOO··O······O··O····OO··OO·····
·········
Done setting up <Thread(thr_1, started daemo
n 140203238823488)>
<Thread(thr_2, started daemon 14020323043078
4)> O······O·OOO··O······O··O····OO··OO·····
·········
Done setting up <Thread(thr_2, started daemo
n 140203230430784)>
```

```
(base) jackieh@Jackies-MacBook-Pro desktop % python3 battleshipclient
.py
Enter your player name: p1
You will be choosing the location for ship size 4
Please type the starting and ending location for your ship, please pu
t a space inbetween (Example: A0 D0): B3 E3

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 3
Please type the starting and ending location for your ship, please pu
t a space inbetween (Example: A0 D0): F0 F2

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please pu
t a space inbetween (Example: A0 D0): D6 E6

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  O  O  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please pu
t a space inbetween (Example: A0 D0): A5 A6

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
```

```
(base) jackieh@Jackies-MacBook-Pro desktop % python3 battleshipclient.p
y
Enter your player name: p2
You will be choosing the location for ship size 4
Please type the starting and ending location for your ship, please put
a space inbetween (Example: A0 D0): A0 A3

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  ·  ·  ·  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 3
Please type the starting and ending location for your ship, please put
a space inbetween (Example: A0 D0): C1 E1

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please put
a space inbetween (Example: A0 D0): f4 G4

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please put
a space inbetween (Example: A0 D0): b4 c4

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
```

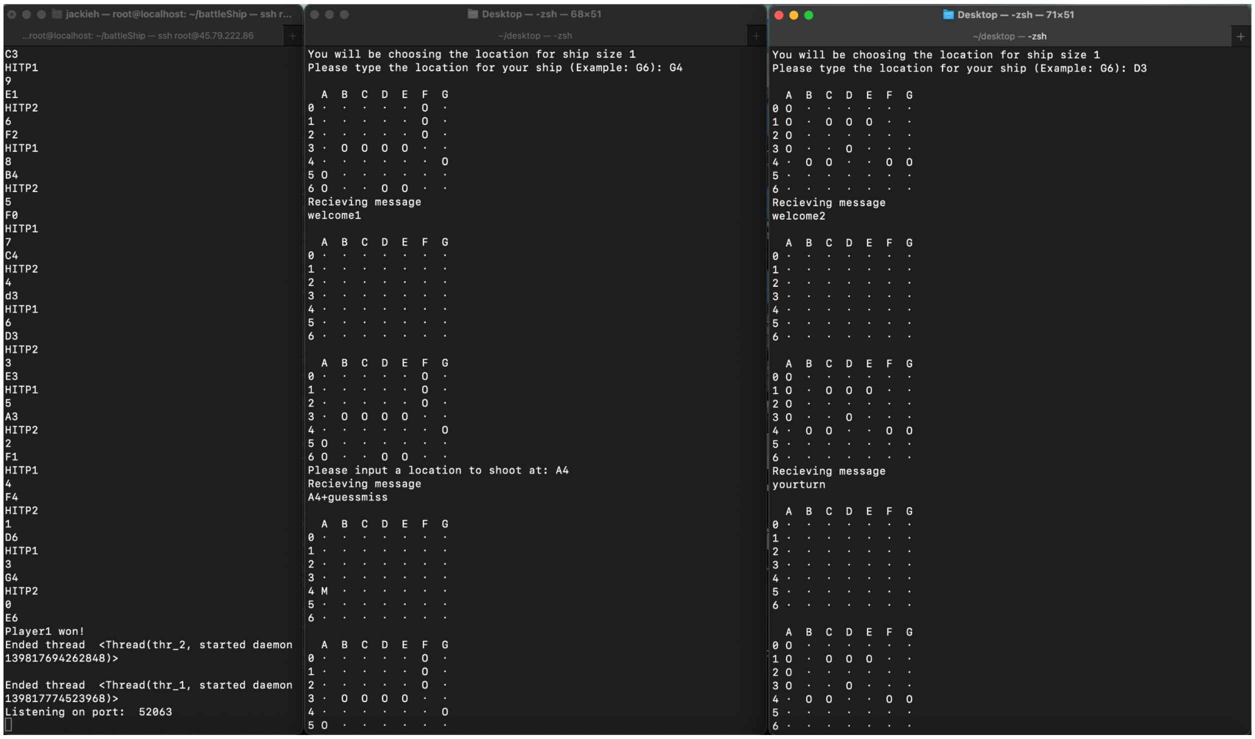

jackieh — root@localhost: ~/battleShip — ssh r...
...root@localhost: ~/battleShip — ssh root@45.79.222.86
Desktop — python3 battleshipclient.py — 69×51
~/desktop — python3 battleshipclient.py
Desktop — python3 battleshipclient.py — 71×51
~/desktop — python3 battleshipclient.py

```
root@localhost:~/battleShip# python3 battles
hip.py
Listening on port:  52063
Started thread  <Thread(thr_1, started daemo
n 140203238823488)>
Listening on port:  52063
Started thread  <Thread(thr_2, started daemo
n 140203230430784)>
p1
p2
<Thread(thr_2, started daemon 14020323043078
4)> ·····O······O······O··OOOO········OO····
··O··OO··
<Thread(thr_1, started daemon 14020323882348
8)> ·····O······O······O··OOOO········OO····
··O··OO··
<Thread(thr_1, started daemon 14020323882348
8)> O······O·OOO··O······O··O····OO··OO·····
·········
Done setting up <Thread(thr_1, started daemo
n 140203238823488)>
<Thread(thr_2, started daemon 14020323043078
4)> O······O·OOO··O······O··O····OO··OO·····
·········
Done setting up <Thread(thr_2, started daemo
n 140203230430784)>
```

```
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  O  O  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please pu
t a space inbetween (Example: A0 D0): A5 A6

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
You will be choosing the location for ship size 1
Please type the location for your ship (Example: G6): G4

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  O
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
Recieving message
welcome2

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  O
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
Recieving message
```

```
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 2
Please type the starting and ending location for your ship, please put
a space inbetween (Example: A0 D0): b4 c4

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  ·  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
You will be choosing the location for ship size 1
Please type the location for your ship (Example: G6): d3

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  O  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
Recieving message
welcome2

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  O  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
Recieving message
```

Glimpse of Game Play:

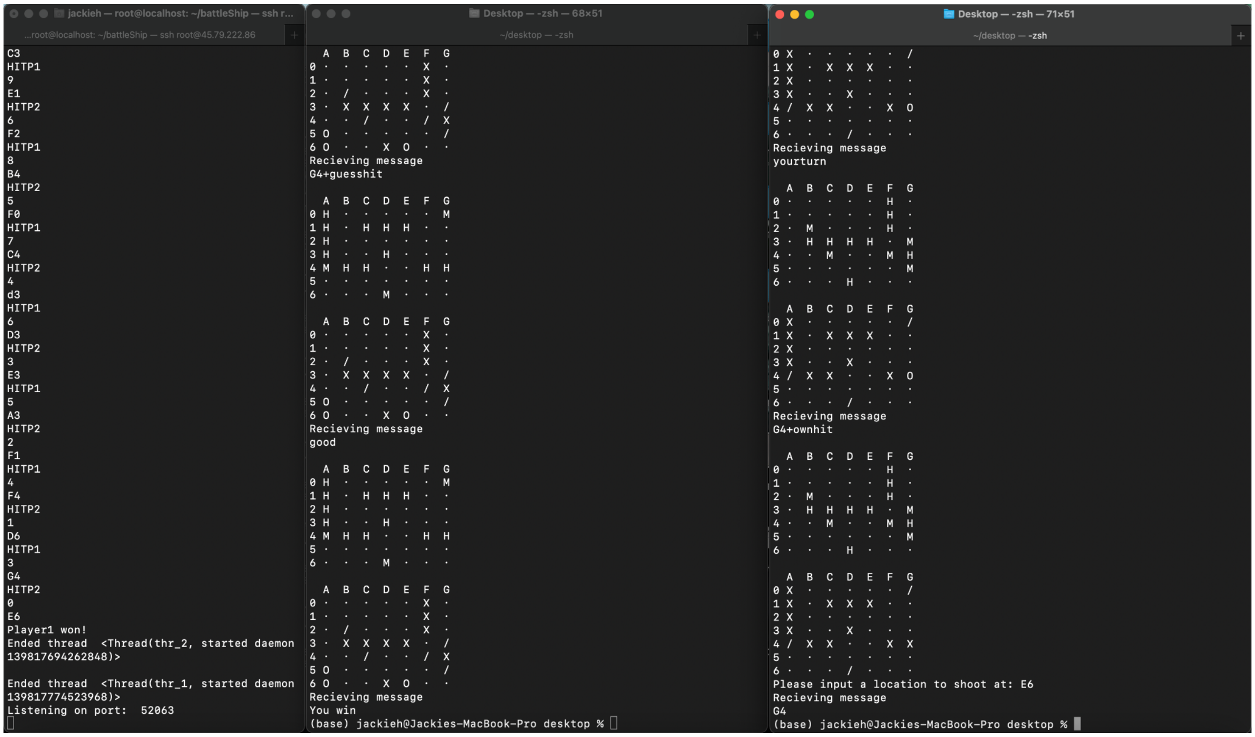

jackieh — root@localhost: ~/battleShip — ssh r...
...root@localhost: ~/battleShip — ssh root@45.79.222.86
Desktop — -zsh — 68×51
~/desktop — -zsh
Desktop — -zsh — 71×51
~/desktop — -zsh

```
C3
HITP1
9
E1
HITP2
6
F2
HITP1
8
B4
HITP2
5
F0
HITP1
7
C4
HITP2
4
d3
HITP1
6
D3
HITP2
3
E3
HITP1
5
A3
HITP2
2
F1
HITP1
4
F4
HITP2
1
D6
HITP1
3
G4
HITP2
0
E6
Player1 won!
Ended thread  <Thread(thr_2, started daemon
139817694262848)>

Ended thread  <Thread(thr_1, started daemon
139817774523968)>
Listening on port:  52063
```

```
You will be choosing the location for ship size 1
Please type the location for your ship (Example: G6): G4

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  O
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
Recieving message
welcome1

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  O
5 O  ·  ·  ·  ·  ·  ·
6 O  ·  ·  O  O  ·  ·
Please input a location to shoot at: A4
Recieving message
A4+guessmiss

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 M  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  O  ·
1 ·  ·  ·  ·  ·  O  ·
2 ·  ·  ·  ·  ·  O  ·
3 ·  O  O  O  O  ·  ·
4 ·  ·  ·  ·  ·  ·  O
5 O  ·  ·  ·  ·  ·  ·
```

```
You will be choosing the location for ship size 1
Please type the location for your ship (Example: G6): D3

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  O  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
Recieving message
welcome2

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  O  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
Recieving message
yourturn

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  ·  ·
1 ·  ·  ·  ·  ·  ·  ·
2 ·  ·  ·  ·  ·  ·  ·
3 ·  ·  ·  ·  ·  ·  ·
4 ·  ·  ·  ·  ·  ·  ·
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·

  A  B  C  D  E  F  G
0 O  ·  ·  ·  ·  ·  ·
1 O  ·  O  O  O  ·  ·
2 O  ·  ·  ·  ·  ·  ·
3 O  ·  ·  O  ·  ·  ·
4 ·  O  O  ·  ·  O  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  ·  ·  ·  ·
```

*This is the server beginning game play after the players have inputted their ship coordinates


jackieh — root@localhost: ~/battleShip — ssh r...
...root@localhost: ~/battleShip — ssh root@45.79.222.86
Desktop — -zsh — 68×51
~/desktop — -zsh
Desktop — -zsh — 71×51
~/desktop — -zsh

```
C3
HITP1
9
E1
HITP2
6
F2
HITP1
8
B4
HITP2
5
F0
HITP1
7
C4
HITP2
4
d3
HITP1
6
D3
HITP2
3
E3
HITP1
5
A3
HITP2
2
F1
HITP1
4
F4
HITP2
1
D6
HITP1
3
G4
HITP2
0
E6
Player1 won!
Ended thread  <Thread(thr_2, started daemon
139817694262848)>

Ended thread  <Thread(thr_1, started daemon
139817774523968)>
Listening on port:  52063
```

```
  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  X  ·
1 ·  ·  ·  ·  ·  X  ·
2 ·  /  ·  ·  ·  X  ·
3 ·  X  X  X  X  ·  /
4 ·  ·  /  ·  ·  /  X
5 O  ·  ·  ·  ·  ·  /
6 O  ·  ·  X  O  ·  ·
Recieving message
G4+guesshit

  A  B  C  D  E  F  G
0 H  ·  ·  ·  ·  ·  M
1 H  ·  H  H  H  ·  ·
2 H  ·  ·  ·  ·  ·  ·
3 H  ·  ·  H  ·  ·  ·
4 M  H  H  ·  ·  H  H
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  M  ·  ·  ·

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  X  ·
1 ·  ·  ·  ·  ·  X  ·
2 ·  /  ·  ·  ·  X  ·
3 ·  X  X  X  X  ·  /
4 ·  ·  /  ·  ·  /  X
5 O  ·  ·  ·  ·  ·  /
6 O  ·  ·  X  O  ·  ·
Recieving message
good

  A  B  C  D  E  F  G
0 H  ·  ·  ·  ·  ·  M
1 H  ·  H  H  H  ·  ·
2 H  ·  ·  ·  ·  ·  ·
3 H  ·  ·  H  ·  ·  ·
4 M  H  H  ·  ·  H  H
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  M  ·  ·  ·

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  X  ·
1 ·  ·  ·  ·  ·  X  ·
2 ·  /  ·  ·  ·  X  ·
3 ·  X  X  X  X  ·  /
4 ·  ·  /  ·  ·  /  X
5 O  ·  ·  ·  ·  ·  /
6 O  ·  ·  X  O  ·  ·
Recieving message
You win
(base) jackieh@Jackies-MacBook-Pro desktop % 
```

```
0 X  ·  ·  ·  ·  ·  /
1 X  ·  X  X  X  ·  ·
2 X  ·  ·  ·  ·  ·  ·
3 X  ·  ·  X  ·  ·  ·
4 /  X  X  ·  ·  X  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  /  ·  ·  ·
Recieving message
yourturn

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  H  ·
1 ·  ·  ·  ·  ·  H  ·
2 ·  M  ·  ·  ·  H  ·
3 ·  H  H  H  H  ·  M
4 ·  ·  M  ·  ·  M  H
5 ·  ·  ·  ·  ·  ·  M
6 ·  ·  ·  H  ·  ·  ·

  A  B  C  D  E  F  G
0 X  ·  ·  ·  ·  ·  /
1 X  ·  X  X  X  ·  ·
2 X  ·  ·  ·  ·  ·  ·
3 X  ·  ·  X  ·  ·  ·
4 /  X  X  ·  ·  X  O
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  /  ·  ·  ·
Recieving message
G4+ownhit

  A  B  C  D  E  F  G
0 ·  ·  ·  ·  ·  H  ·
1 ·  ·  ·  ·  ·  H  ·
2 ·  M  ·  ·  ·  H  ·
3 ·  H  H  H  H  ·  M
4 ·  ·  M  ·  ·  M  H
5 ·  ·  ·  ·  ·  ·  M
6 ·  ·  ·  H  ·  ·  ·

  A  B  C  D  E  F  G
0 X  ·  ·  ·  ·  ·  /
1 X  ·  X  X  X  ·  ·
2 X  ·  ·  ·  ·  ·  ·
3 X  ·  ·  X  ·  ·  ·
4 /  X  X  ·  ·  X  X
5 ·  ·  ·  ·  ·  ·  ·
6 ·  ·  ·  /  ·  ·  ·
Please input a location to shoot at: E6
Recieving message
G4
(base) jackieh@Jackies-MacBook-Pro desktop % 
```

*This is the end of game play.

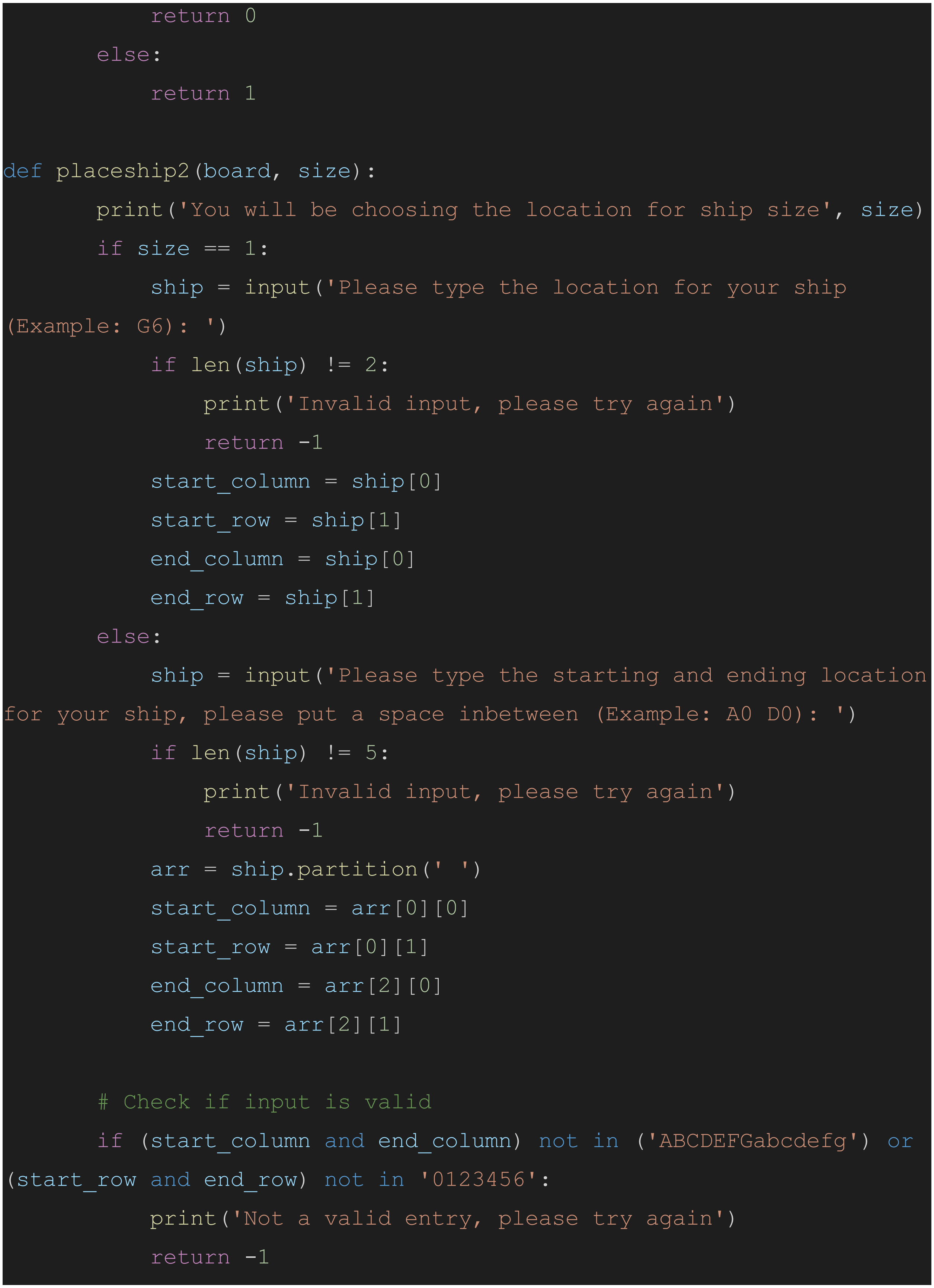
Conclusion:

This project was a great learning experience because it gave us the opportunity to learn about higher level networking. When we first started the project we followed what we did for the first project when we did the digital wallet. We first made our design by hand by making what we wanted to see in an architecture and what scenarios our design would handle. During the design making period we would turn in our designs for feedback on how we could improve. This was helpful because we were able to push ourselves and add more creativity to our project. After creating our architecture and design we then made a simple implementation of a log in and a player search. When we first began the project we thought we would be able to add a login, player search, and a chat feature. When we first started the project we were going to work with Amazon Web Services (AWS). When we were working on the basic implementation stage we explored AWS by watching a LinkedIn Learning course and we created a server. After making the simple implementation it was time to create the game and then ultimately upload it to a cloud service. When we first started the final implementation of our project we first worked on the simple program of making a battleship game. This step was one of the faster steps when working on the final implementation. The hardest part of the final implementation was getting the communications to first work locally with our battleship game. Once we were able to figure out how to get our game to work locally with multiple players we then went back to exploring AWS so we could put our project on a cloud based service to allow for the multiplayer player game to be played on more than one computer. Since we finished the project the morning of the final project due date we decided to explore other cloud services. Since we did this, we were able to find a helpful YouTube tutorial on how to put our code on Linode cloud services. Due to this video being easy to follow we switched from AWS to Linode and that is the cloud service we ended up using with the Ubuntu operating system.

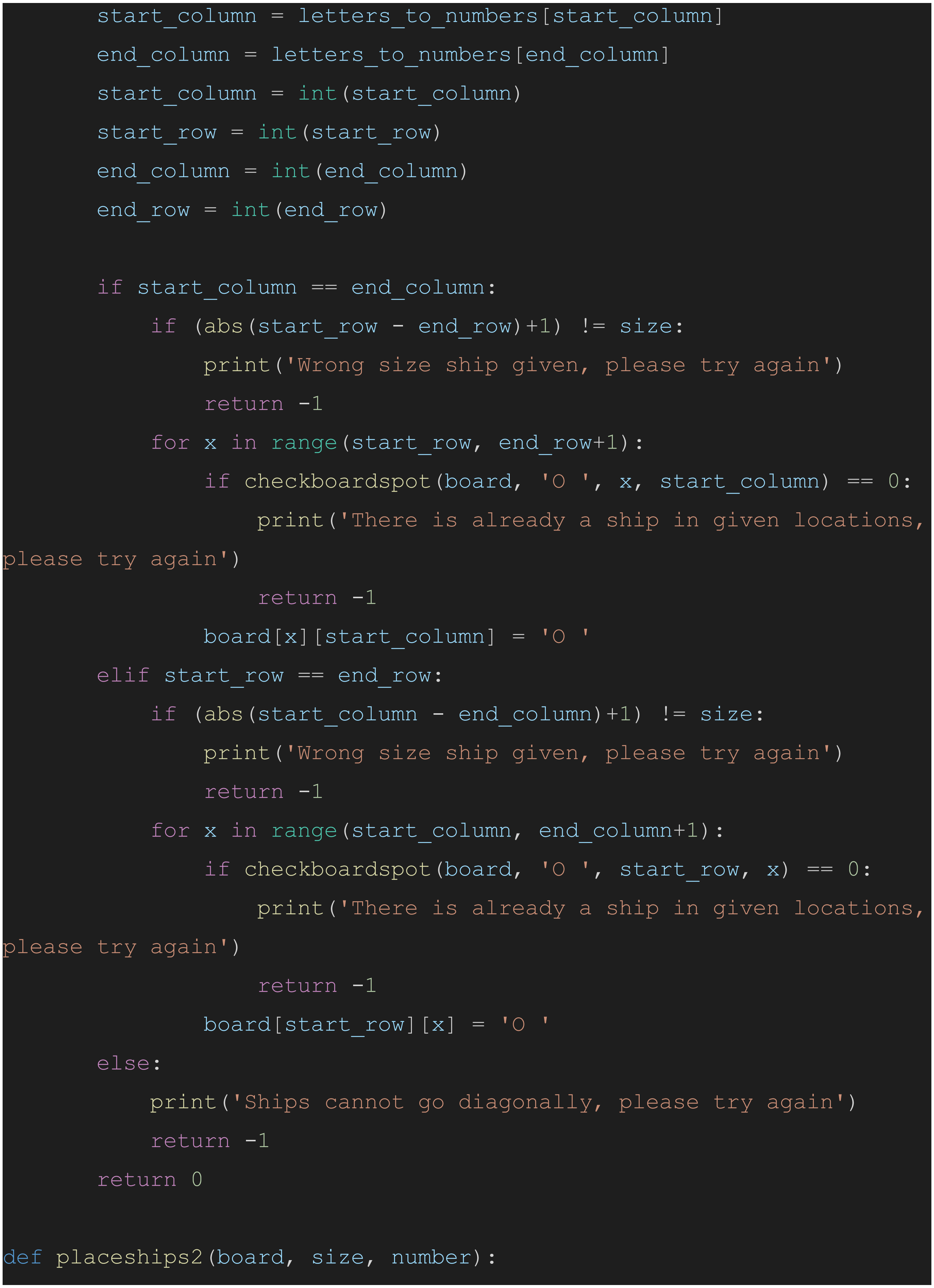
Code:

Client code:

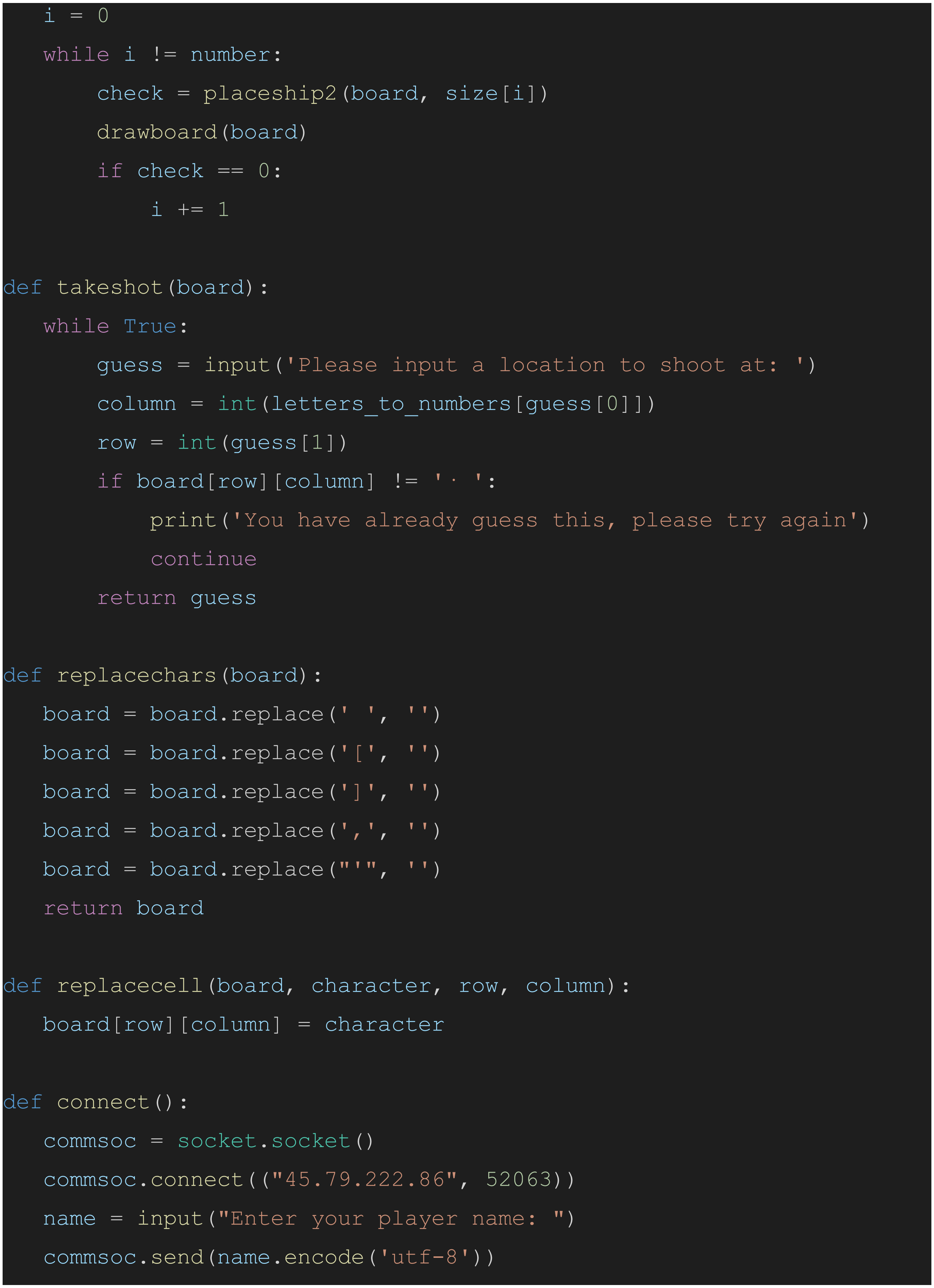
```
import socket
import threading
from time import sleep

from numpy import take

your_turn = False

yourships = {'Ship': 'O', 'Hit': 'X', 'Miss': '/', 'noship': '·'}
enemyships = {'Hit': 'H', 'Miss': 'M', 'noattempt': '·'}
letters_to_numbers = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5,
'G': 6,
                     'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5,
'g': 6}
ship_sizes = [4, 3, 2, 2, 1]
playershipboard = [['· '] * 7 for x in range(7)]
playerguessboard = [['· '] * 7 for x in range(7)]
playerhealth = 12

def drawboard(board):
       for i in range(0,7):
           print()
           if i == 0:
               print('  A  B  C  D  E  F  G ')
           print(i, end= ' ')
           for j in range(0,7):
               print(board[i][j], end=' ')
       print()

def checkboardspot(board, checking, row, column):
       if board[row][column] == checking:
```



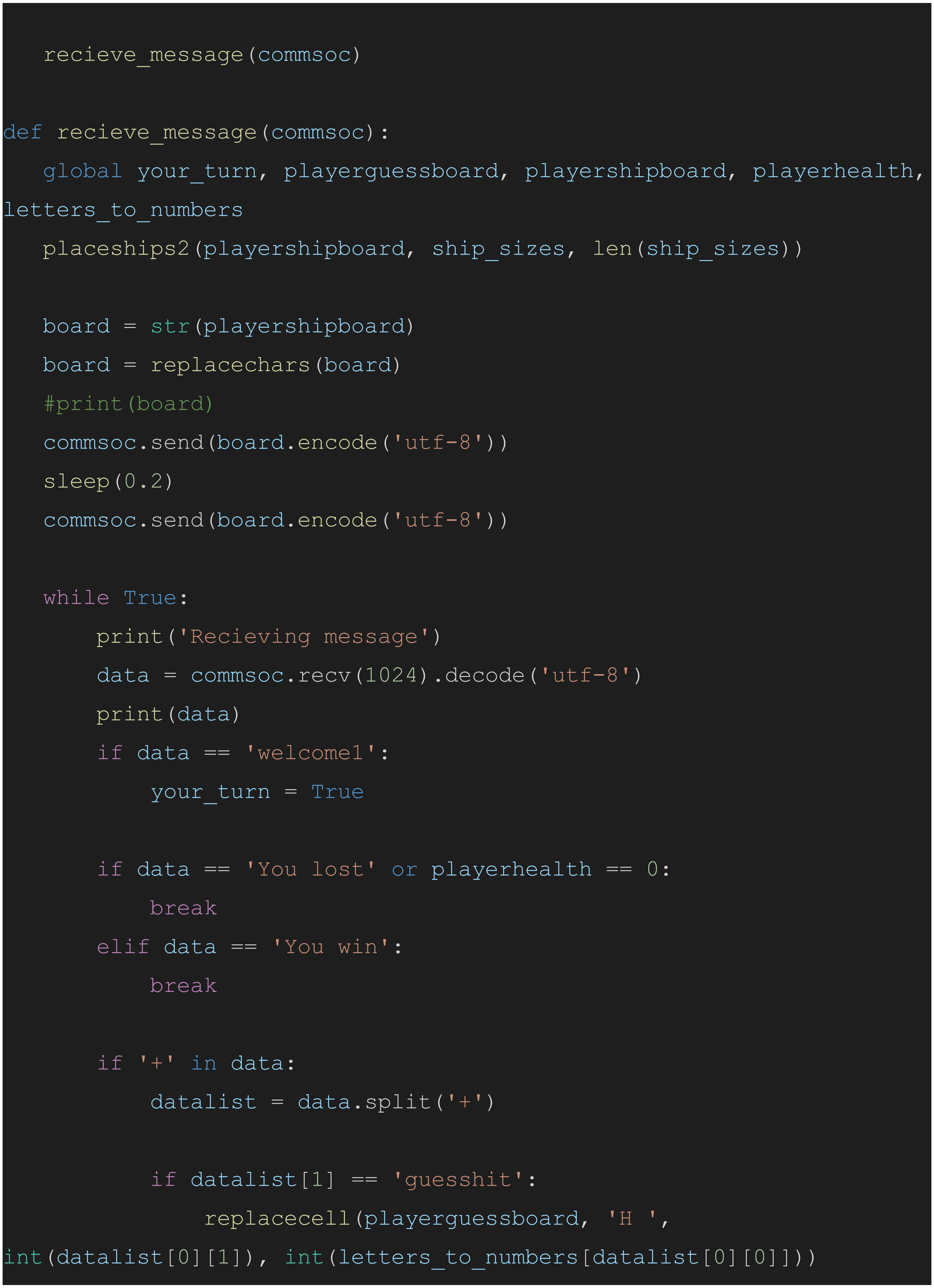
```python
            return 0
        else:
            return 1

def placeship2(board, size):
        print('You will be choosing the location for ship size', size)
        if size == 1:
            ship = input('Please type the location for your ship (Example: G6): ')
            if len(ship) != 2:
                print('Invalid input, please try again')
                return -1
            start_column = ship[0]
            start_row = ship[1]
            end_column = ship[0]
            end_row = ship[1]
        else:
            ship = input('Please type the starting and ending location for your ship, please put a space inbetween (Example: A0 D0): ')
            if len(ship) != 5:
                print('Invalid input, please try again')
                return -1
            arr = ship.partition(' ')
            start_column = arr[0][0]
            start_row = arr[0][1]
            end_column = arr[2][0]
            end_row = arr[2][1]

        # Check if input is valid
        if (start_column and end_column) not in ('ABCDEFGabcdefg') or (start_row and end_row) not in '0123456':
            print('Not a valid entry, please try again')
            return -1
```



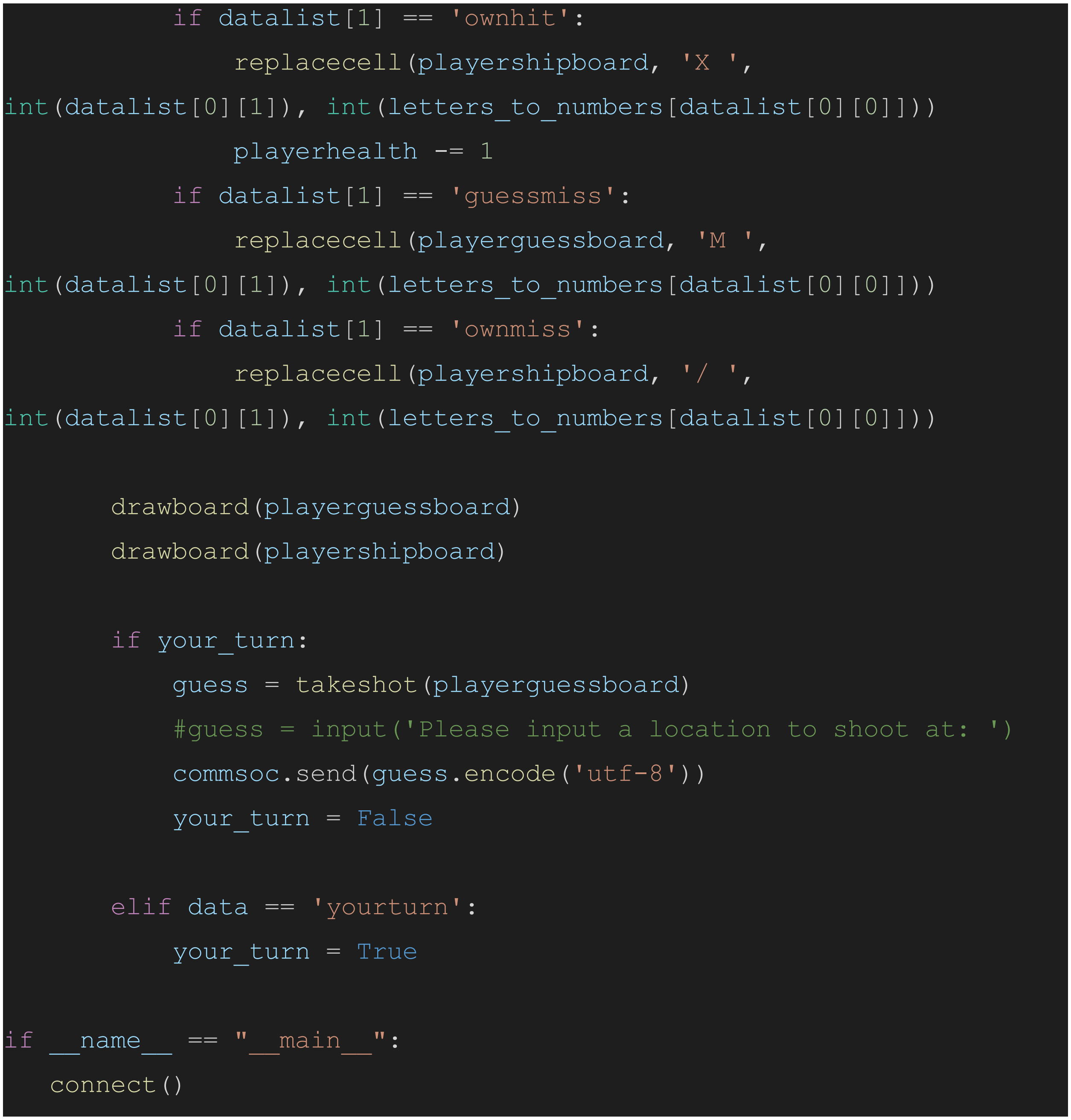
```
        start_column = letters_to_numbers[start_column]
        end_column = letters_to_numbers[end_column]
        start_column = int(start_column)
        start_row = int(start_row)
        end_column = int(end_column)
        end_row = int(end_row)

        if start_column == end_column:
            if (abs(start_row - end_row)+1) != size:
                print('Wrong size ship given, please try again')
                return -1
            for x in range(start_row, end_row+1):
                if checkboardspot(board, 'O ', x, start_column) == 0:
                    print('There is already a ship in given locations,
please try again')
                    return -1
                board[x][start_column] = 'O '
        elif start_row == end_row:
            if (abs(start_column - end_column)+1) != size:
                print('Wrong size ship given, please try again')
                return -1
            for x in range(start_column, end_column+1):
                if checkboardspot(board, 'O ', start_row, x) == 0:
                    print('There is already a ship in given locations,
please try again')
                    return -1
                board[start_row][x] = 'O '
        else:
            print('Ships cannot go diagonally, please try again')
            return -1
        return 0

def placeships2(board, size, number):
```



```
    i = 0
    while i != number:
        check = placeship2(board, size[i])
        drawboard(board)
        if check == 0:
            i += 1

def takeshot(board):
    while True:
        guess = input('Please input a location to shoot at: ')
        column = int(letters_to_numbers[guess[0]])
        row = int(guess[1])
        if board[row][column] != '· ':
            print('You have already guess this, please try again')
            continue
        return guess

def replacechars(board):
    board = board.replace(' ', '')
    board = board.replace('[', '')
    board = board.replace(']', '')
    board = board.replace(',', '')
    board = board.replace("'", '')
    return board

def replacecell(board, character, row, column):
    board[row][column] = character

def connect():
    commsoc = socket.socket()
    commsoc.connect(("45.79.222.86", 52063))
    name = input("Enter your player name: ")
    commsoc.send(name.encode('utf-8'))
```



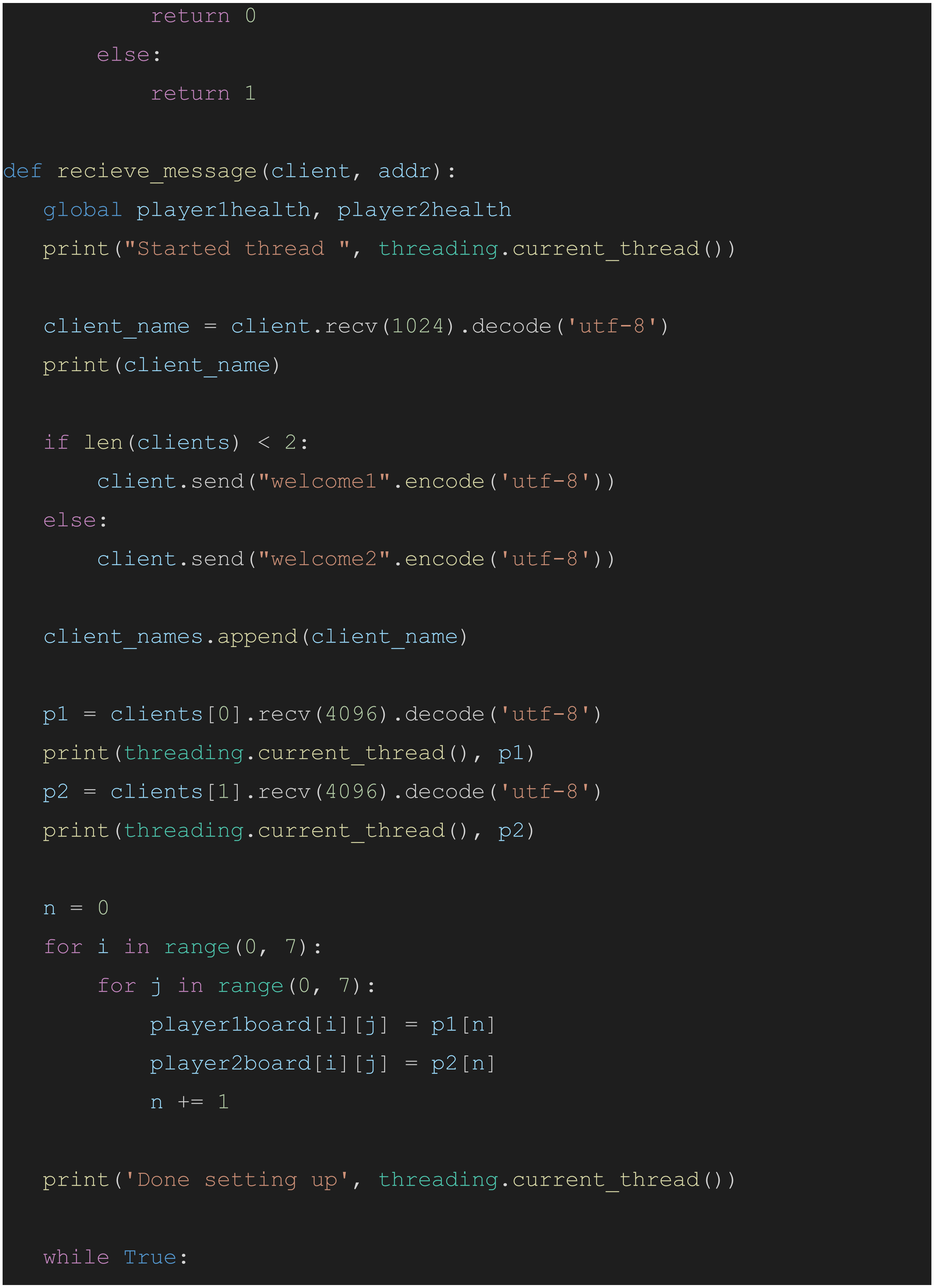
```
   recieve_message(commsoc)

def recieve_message(commsoc):
   global your_turn, playerguessboard, playershipboard, playerhealth,
letters_to_numbers
   placeships2(playershipboard, ship_sizes, len(ship_sizes))

   board = str(playershipboard)
   board = replacechars(board)
   #print(board)
   commsoc.send(board.encode('utf-8'))
   sleep(0.2)
   commsoc.send(board.encode('utf-8'))

   while True:
       print('Recieving message')
       data = commsoc.recv(1024).decode('utf-8')
       print(data)
       if data == 'welcome1':
           your_turn = True

       if data == 'You lost' or playerhealth == 0:
           break
       elif data == 'You win':
           break

       if '+' in data:
           datalist = data.split('+')

           if datalist[1] == 'guesshit':
               replacecell(playerguessboard, 'H ',
int(datalist[0][1]), int(letters_to_numbers[datalist[0][0]]))
```



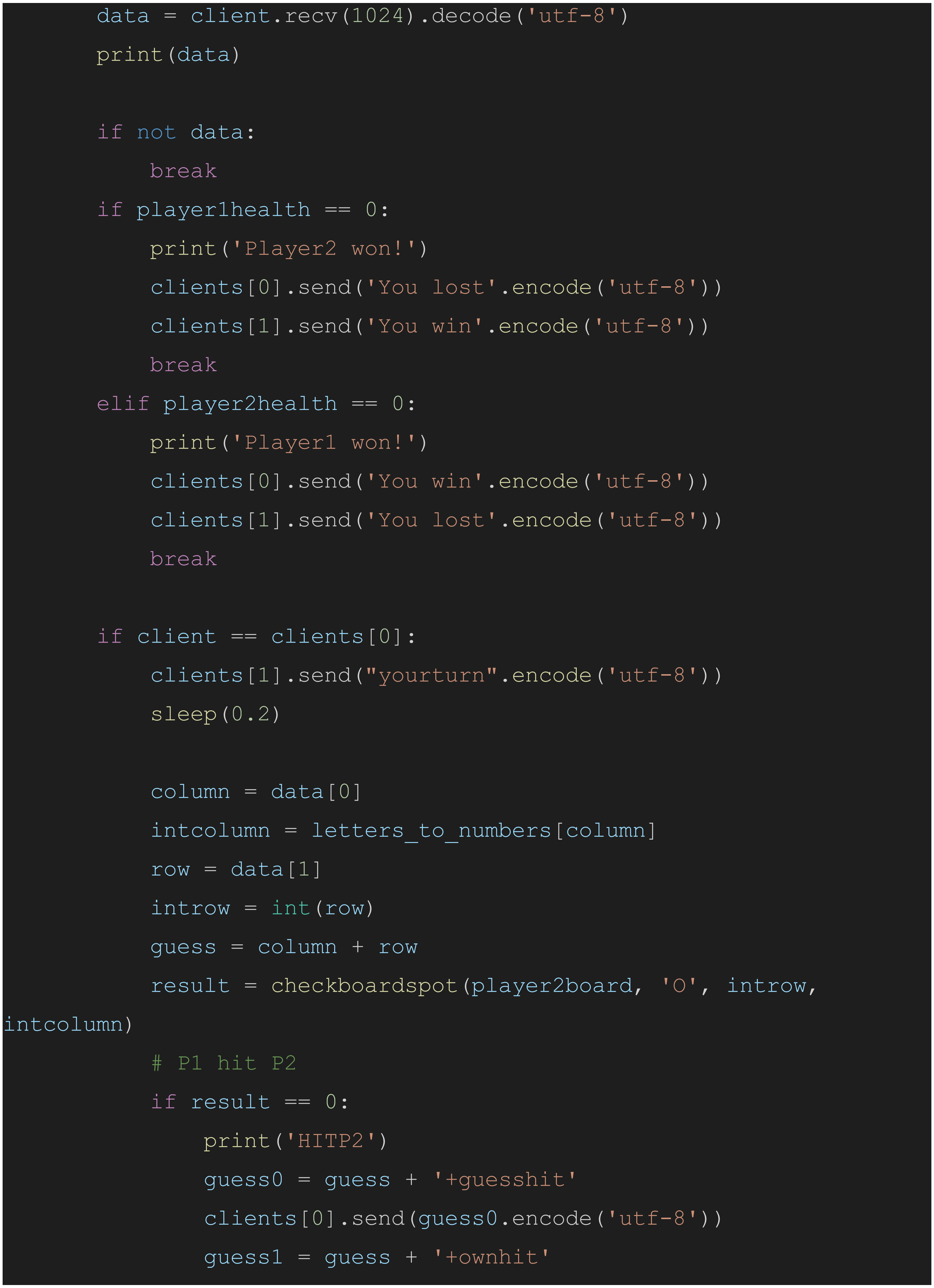
```
            if datalist[1] == 'ownhit':
                replacecell(playershipboard, 'X ',
int(datalist[0][1]), int(letters_to_numbers[datalist[0][0]]))
                playerhealth -= 1
            if datalist[1] == 'guessmiss':
                replacecell(playerguessboard, 'M ',
int(datalist[0][1]), int(letters_to_numbers[datalist[0][0]]))
            if datalist[1] == 'ownmiss':
                replacecell(playershipboard, '/ ',
int(datalist[0][1]), int(letters_to_numbers[datalist[0][0]]))

        drawboard(playerguessboard)
        drawboard(playershipboard)

        if your_turn:
            guess = takeshot(playerguessboard)
            #guess = input('Please input a location to shoot at: ')
            commsoc.send(guess.encode('utf-8'))
            your_turn = False

        elif data == 'yourturn':
            your_turn = True

if __name__ == "__main__":
    connect()
```

Server Code:

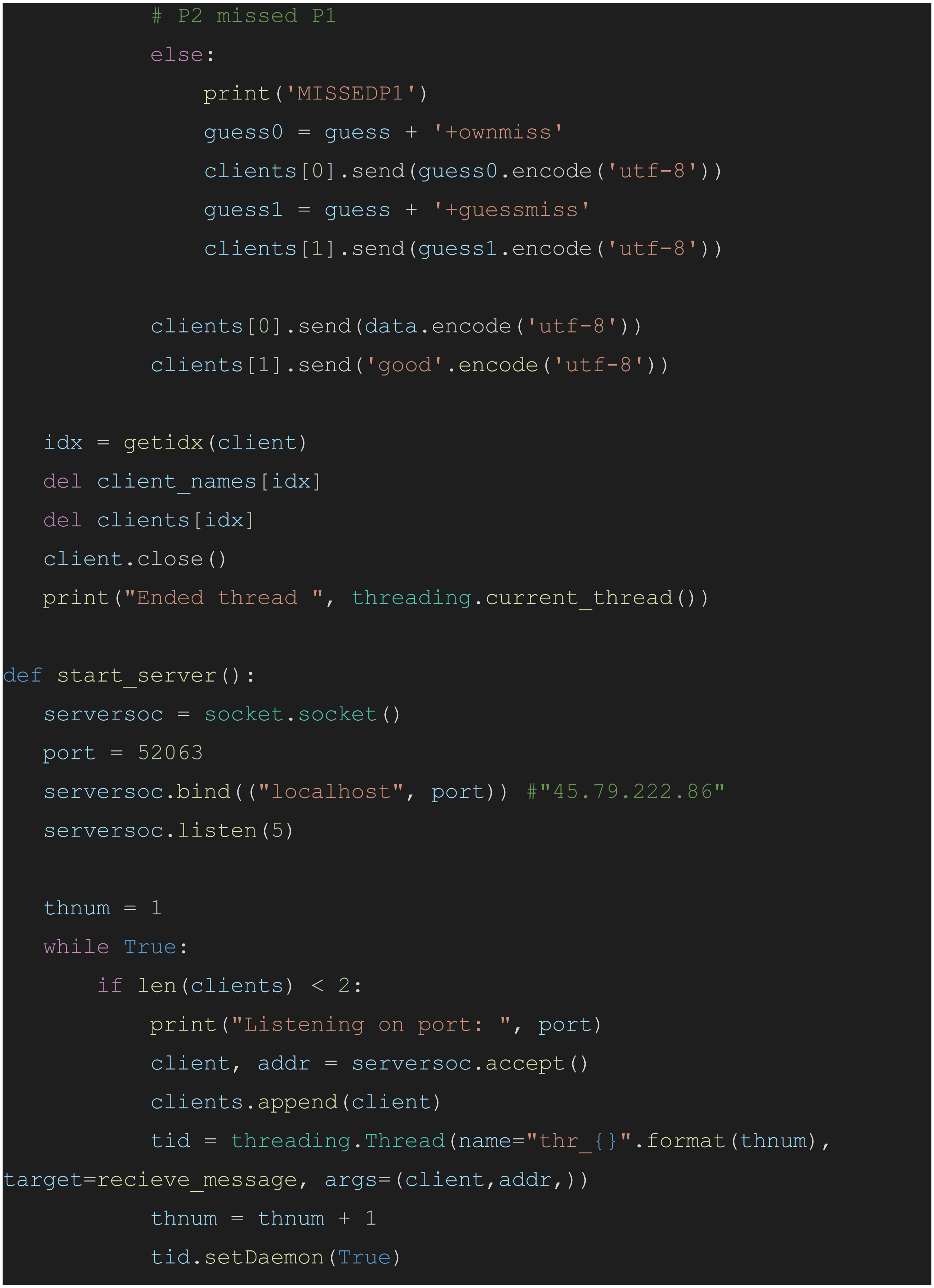
```
import socket
import threading
from time import sleep

clients = []
client_names = []

letters_to_numbers = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5,
'G': 6,
                     'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5,
'g': 6}

player1board = [['· '] * 7 for x in range(7)]
player1health = 12

player2board = [['· '] * 7 for x in range(7)]
player2health = 12

game_start = False

def drawboard(board):
   for i in range(0,7):
       print()
       if i == 0:
           print('  A  B  C  D  E  F  G ')
       print(i, end= ' ')
       for j in range(0,7):
           print(board[i][j], end='  ')
   print()

def checkboardspot(board, checking, row, column):
       if board[row][column] == checking:
```



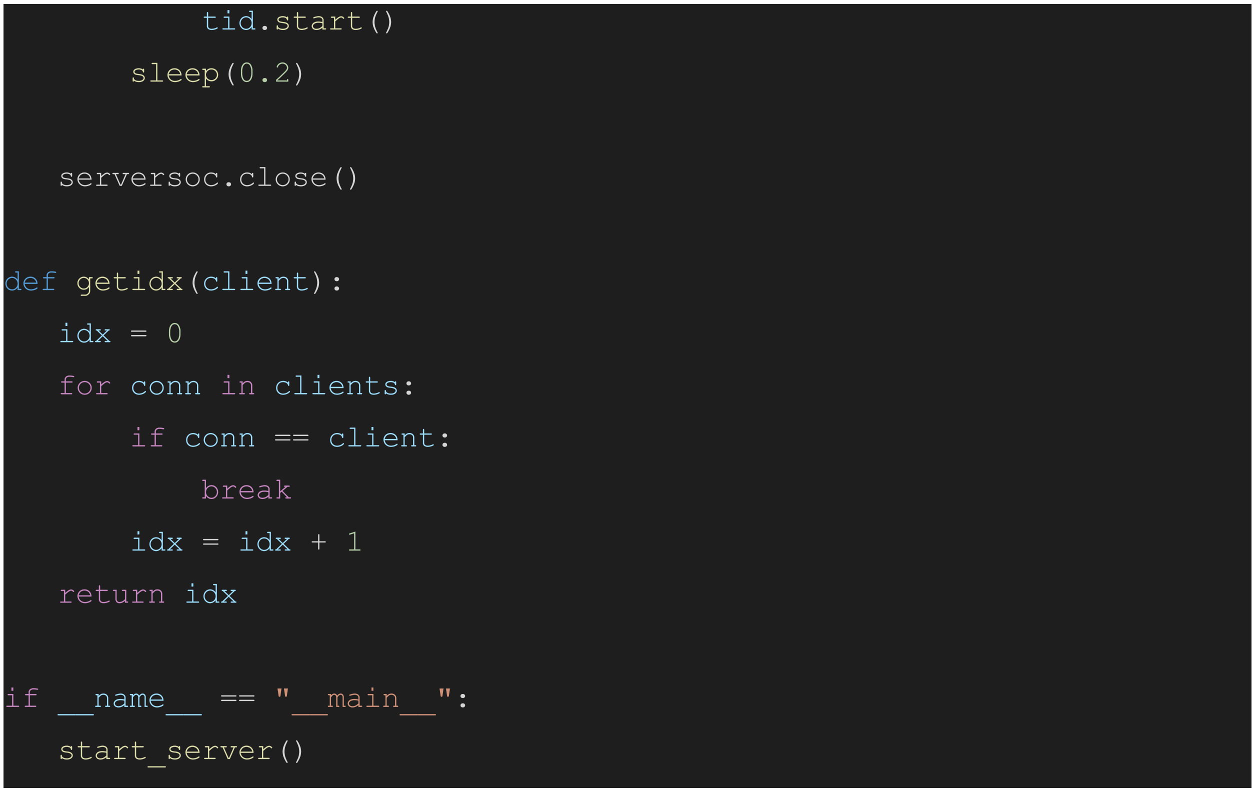
```
            return 0
        else:
            return 1

def recieve_message(client, addr):
    global player1health, player2health
    print("Started thread ", threading.current_thread())

    client_name = client.recv(1024).decode('utf-8')
    print(client_name)

    if len(clients) < 2:
        client.send("welcome1".encode('utf-8'))
    else:
        client.send("welcome2".encode('utf-8'))

    client_names.append(client_name)

    p1 = clients[0].recv(4096).decode('utf-8')
    print(threading.current_thread(), p1)
    p2 = clients[1].recv(4096).decode('utf-8')
    print(threading.current_thread(), p2)

    n = 0
    for i in range(0, 7):
        for j in range(0, 7):
            player1board[i][j] = p1[n]
            player2board[i][j] = p2[n]
            n += 1

    print('Done setting up', threading.current_thread())

    while True:
```



```
        data = client.recv(1024).decode('utf-8')
        print(data)

        if not data:
            break
        if player1health == 0:
            print('Player2 won!')
            clients[0].send('You lost'.encode('utf-8'))
            clients[1].send('You win'.encode('utf-8'))
            break
        elif player2health == 0:
            print('Player1 won!')
            clients[0].send('You win'.encode('utf-8'))
            clients[1].send('You lost'.encode('utf-8'))
            break

        if client == clients[0]:
            clients[1].send("yourturn".encode('utf-8'))
            sleep(0.2)

            column = data[0]
            intcolumn = letters_to_numbers[column]
            row = data[1]
            introw = int(row)
            guess = column + row
            result = checkboardspot(player2board, 'O', introw,
intcolumn)
            # P1 hit P2
            if result == 0:
                print('HITP2')
                guess0 = guess + '+guesshit'
                clients[0].send(guess0.encode('utf-8'))
                guess1 = guess + '+ownhit'
                clients[1].send(guess1.encode('utf-8'))
                player2health -= 1
                print(player2health)
            # P1 missed P2
            else:
                print('MISSEDP2')
                guess0 = guess + '+guessmiss'
                clients[0].send(guess0.encode('utf-8'))
                guess1 = guess + '+ownmiss'
                clients[1].send(guess1.encode('utf-8'))

            clients[1].send(data.encode('utf-8'))
            clients[0].send('good'.encode('utf-8'))
        else:
            clients[0].send("yourturn".encode('utf-8'))
            sleep(0.2)

            column = data[0]
            intcolumn = letters_to_numbers[column]
            row = data[1]
            introw = int(row)
            guess = column + row
            result = checkboardspot(player1board, 'O', introw,
intcolumn)
            # P2 hit P1
            if result == 0:
                print('HITP1')
                guess0 = guess + '+ownhit'
                clients[0].send(guess0.encode('utf-8'))
                guess1 = guess + '+guesshit'
                clients[1].send(guess1.encode('utf-8'))
                player1health -= 1
                print(player1health)
```



```
            # P2 missed P1
            else:
                print('MISSEDP1')
                guess0 = guess + '+ownmiss'
                clients[0].send(guess0.encode('utf-8'))
                guess1 = guess + '+guessmiss'
                clients[1].send(guess1.encode('utf-8'))

            clients[0].send(data.encode('utf-8'))
            clients[1].send('good'.encode('utf-8'))

    idx = getidx(client)
    del client_names[idx]
    del clients[idx]
    client.close()
    print("Ended thread ", threading.current_thread())

def start_server():
    serversoc = socket.socket()
    port = 52063
    serversoc.bind(("localhost", port)) #"45.79.222.86"
    serversoc.listen(5)

    thnum = 1
    while True:
        if len(clients) < 2:
            print("Listening on port: ", port)
            client, addr = serversoc.accept()
            clients.append(client)
            tid = threading.Thread(name="thr_{}".format(thnum),
target=recieve_message, args=(client,addr,))
            thnum = thnum + 1
            tid.setDaemon(True)
```



```
            tid.start()
        sleep(0.2)

    serversoc.close()

def getidx(client):
    idx = 0
    for conn in clients:
        if conn == client:
            break
        idx = idx + 1
    return idx

if __name__ == "__main__":
    start_server()
```