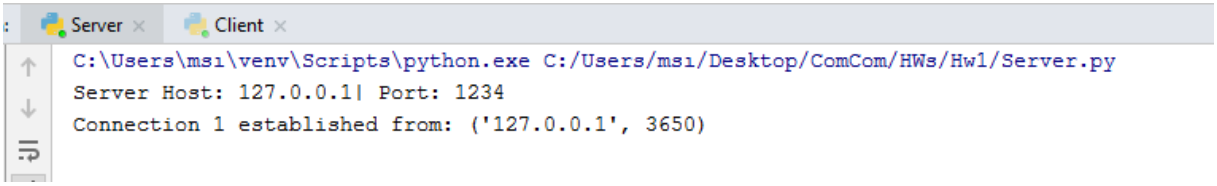
BLG 433E
COMPUTER COMMUNICATIONS
2019-2020 FALL / PROJECT 1
SOCKET PROGRAMMING

REPORT

BEHLÜL GÜLMEZ

150140113

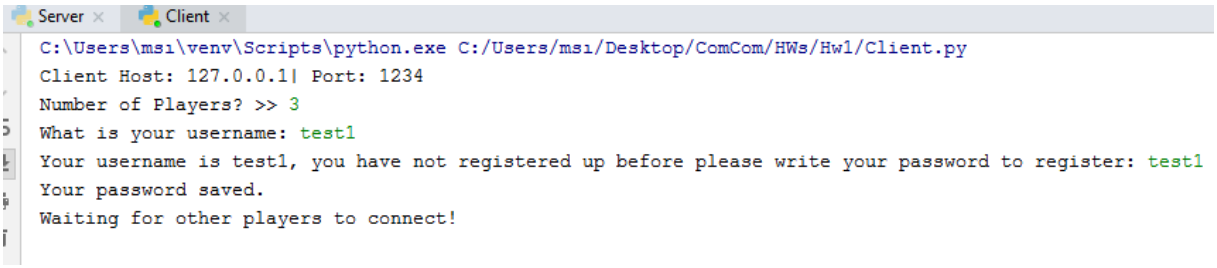
Server.py shows connections



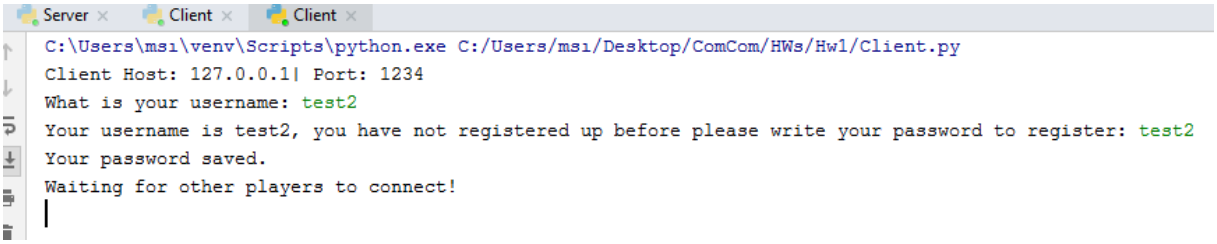
Client.py takes user's info to register or login. Also takes number for how many players will play.

I am testing with 3.

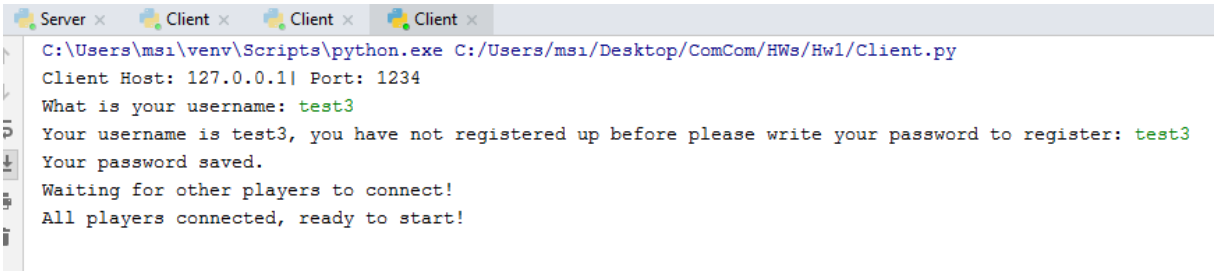
This is client 1.



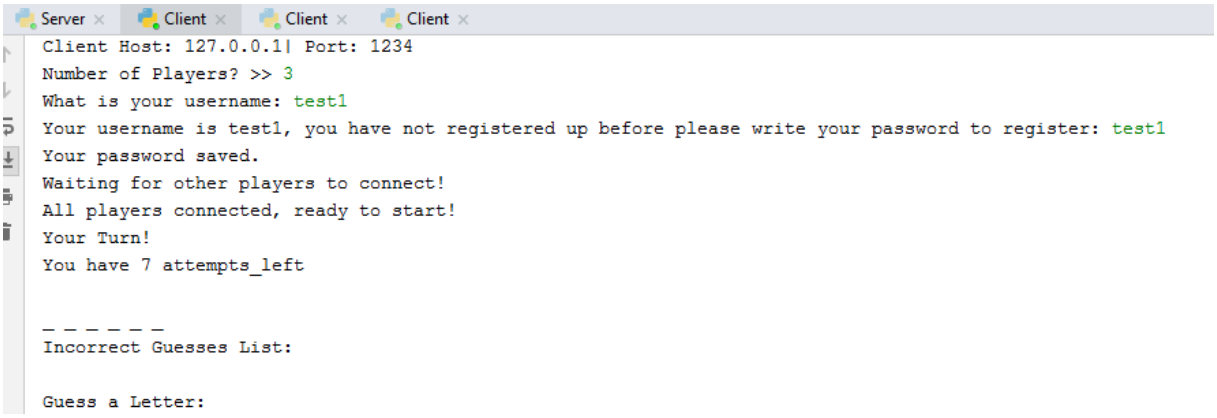
This is client 2.



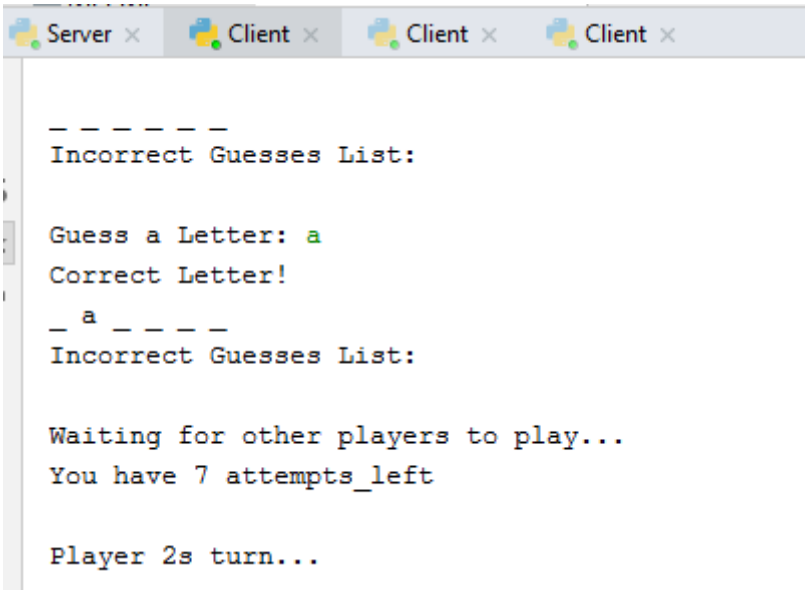
This is client 3.



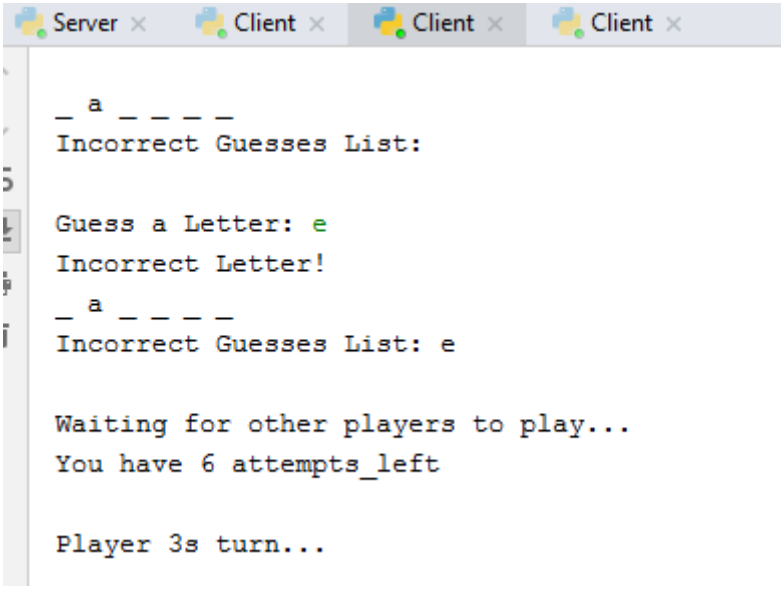
After everyone show up game start. Server asks clients for letter in order. Firstly client1.



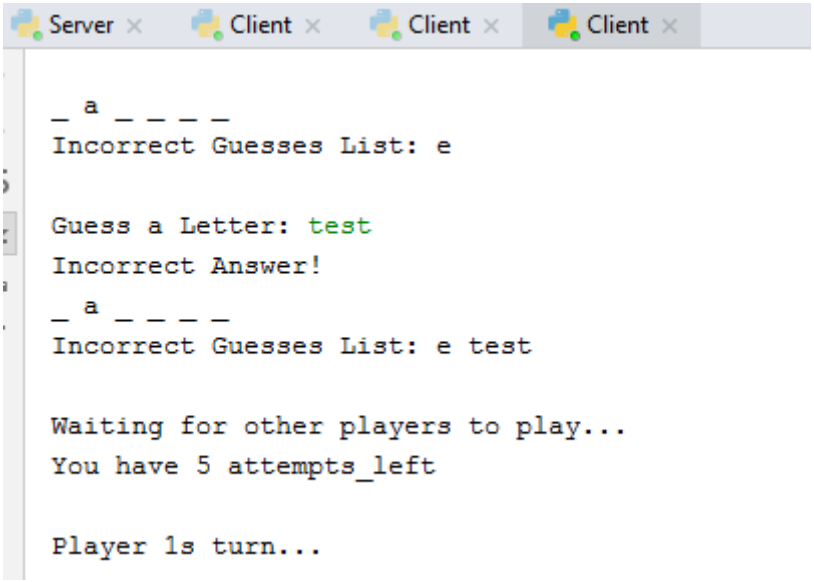
After make a guess next player will play.



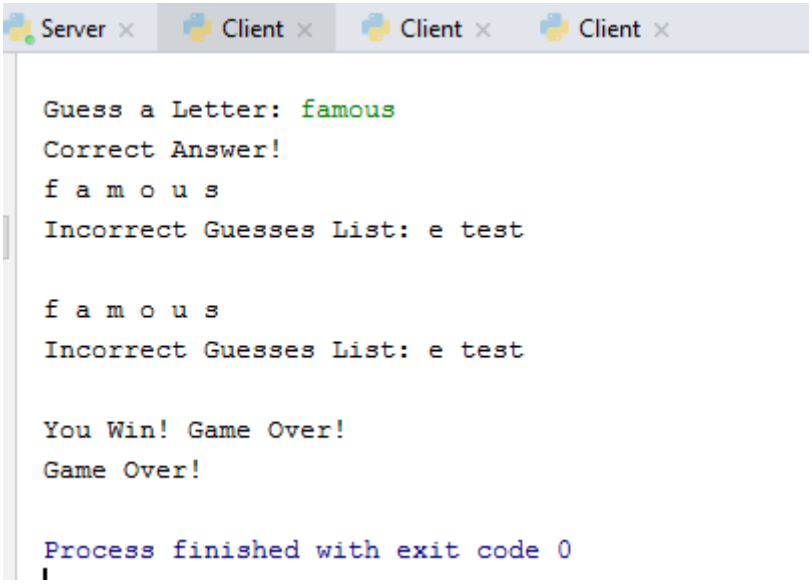
If guess wrong, attempts_left -= 1



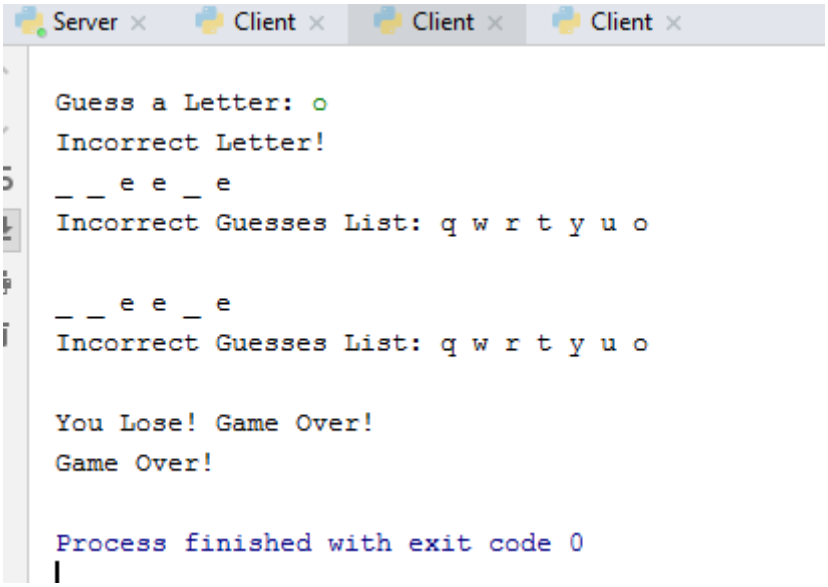
Player can guess a word too.



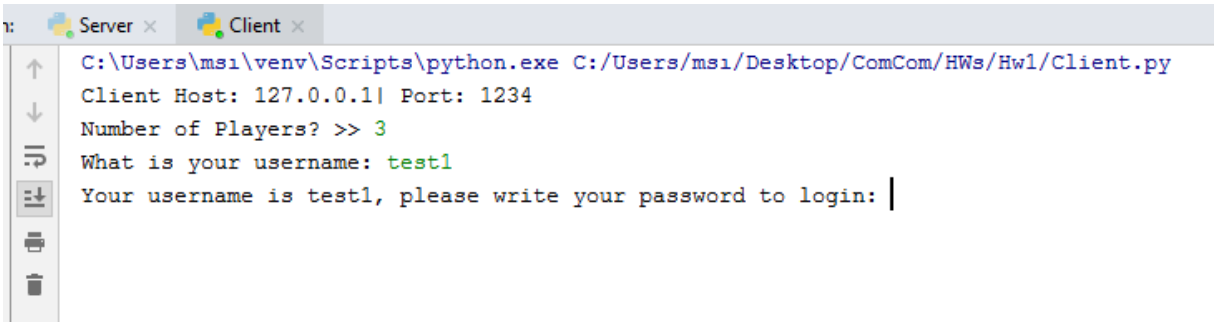
If one player guess correctly clients close.



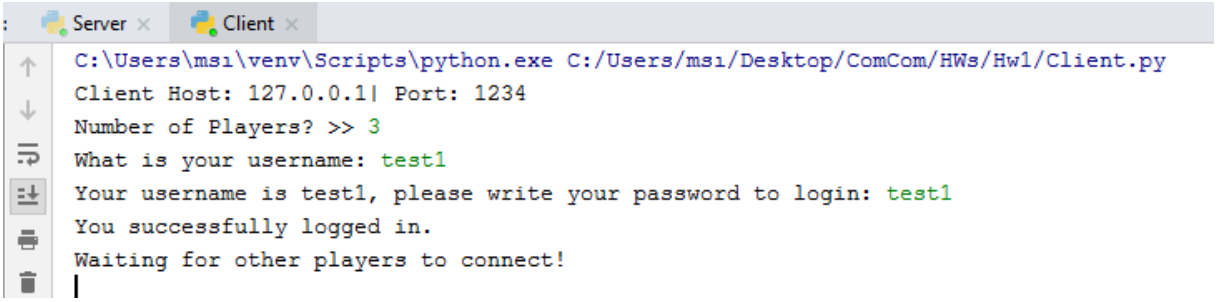
If attempts_left over cleints close too.



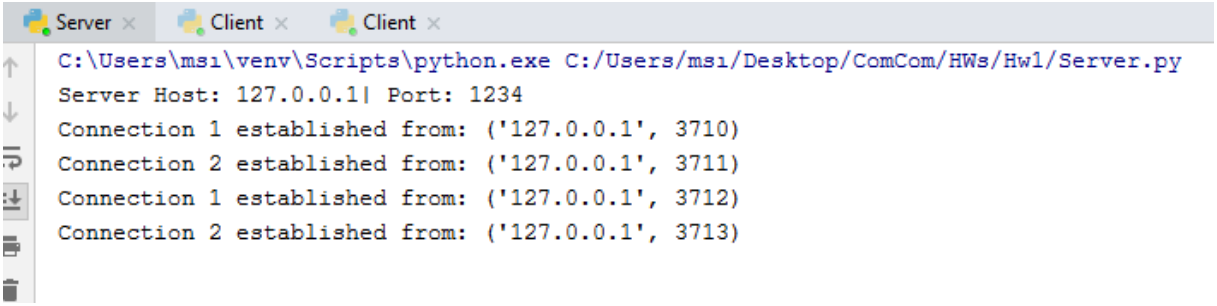
If same username tries to enter again. server asks for password to login.



If player writes true password, server accepts.



After clients close and new clients join to server, connection numbers starts form 1 again.



This class for game info and datas. Holds incorrect letters ect.

Server.py Client.py

```
class GameClass:
    word = ""
    gameString = ""
    numberOfIncorrectGuesses = 0
    listOfIncorrectLetters = 0
    turn = 1
    attempts_left = 7

    def __init__(self, word):
        self.listOfIncorrectLetters = []
        self.word = word
        for i in range(len(word)):
            self.gameString += "_"

    def guessLetter(self, letterGuessed):
        if len(letterGuessed) > 1:
            if letterGuessed == self.word:
                self.gameString = self.word
                return 'Correct Answer!'
            else:
                self.numberOfIncorrectGuesses += 1
                self.attempts_left -= 1
                self.listOfIncorrectLetters.append(letterGuessed)
                return 'Incorrect Answer!'
        else:
            if letterGuessed in self.listOfIncorrectLetters or letterGuessed in self.gameString:
                self.numberOfIncorrectGuesses += 1
                self.attempts_left -= 1
                return 'Incorrect Letter!'
            elif letterGuessed not in self.word:
                self.numberOfIncorrectGuesses += 1
                self.attempts_left -= 1
                self.listOfIncorrectLetters.append(letterGuessed)
                return 'Incorrect Letter!'
            else:
                gameString = list(self.gameString)
                for i in range(len(self.word)):
                    if self.word[i] == letterGuessed:
                        gameString[i] = letterGuessed
                self.gameString = ''.join(gameString)
                return 'Correct Letter!'

    def isOver(self):
```

This functions helps for send data to client.

```
def sendMessage(c, message):
    data = bytes([len(message)]) + bytes(message, 'utf8')
    c.send(data)


def sendDataForGuessLetter(c, game):
    Flag = 0
    data = bytes(game.gameString + " ".join(game.listOfIncorrectLetters), 'utf8')
    finalData = bytes([Flag]) + bytes([len(game.gameString)]) + bytes([len(" ".join(game.listOfIncorrectLetters))]) + data + bytes([game.numberOfIncorrectGuesses])
    c.send(finalData)
```

This is main game function. Until game is over, Works continuesly. Sends strings to clients and takes guesses from clients. Than compare them wirh actual word.

```
def multiplayerGame(c, player, game):
    global sign
    global full

    while True:
        currentTurn = game.turn
        while game.turn != player:
            if currentTurn != game.turn:
                sendMessage(c, " ".join(list(game.gameString)))
                sendMessage(c, "Incorrect Guesses List: " + " ".join(game.listOfIncorrectLetters) + "\n")
                sendMessage(c, "You have " + str(game.attempts_left) + " attempts_left" + "\n")
                sendMessage(c, 'Player ' + str(game.turn) + 's turn...')
                currentTurn = game.turn
            continue

        isOverCheck = game.isOver()
        if isOverCheck != '':
            sendDataForGuessLetter(c, game)
            sendMessage(c, isOverCheck)
            sendMessage(c, "Game Over!")
            game.changeTurn()
            break

        sendMessage(c, 'Your Turn!')
        sendMessage(c, "You have " + str(game.attempts_left) + " attempts_left" + "\n")

        sendDataForGuessLetter(c, game)

        receivedLetterLength = int(c.recv(1)[0])
        receivedLetter = c.recv(receivedLetterLength)
        letterGuessed = receivedLetter.decode('utf-8')

        sendMessage(c, game.guessLetter(letterGuessed))

        sendMessage(c, " ".join(list(game.gameString)))
        sendMessage(c, "Incorrect Guesses List: " + " ".join(game.listOfIncorrectLetters) + "\n")

        isOverCheck = game.isOver()
        if isOverCheck != '':
            sendDataForGuessLetter(c, game)
            sendMessage(c, isOverCheck)
            sendMessage(c, "Game Over!")
            game.changeTurn()
```

This function is where thread start and take username password from clients.

```
def Thread(c, sign):
    global totalNumberOfPlayers
    global full
    global game

    player = sign
    if player == 1:
        word = word_list[random.randint(0, 24)]
        game = GameClass(word)

    sendMessage(c, 'What is your username: ')
    lengthOfText = int(c.recv(1)[0])
    userName = c.recv(lengthOfText).decode('utf8')

    if userName not in user_password:
        registerChecker = 1
        c.send(bytes([registerChecker]))
        sendMessage(c, 'Your username is ' + userName + ', you have not registered up before please write your password to register: ')
        lengthOfText = int(c.recv(1)[0])
        text = c.recv(lengthOfText).decode('utf8')
        user_password[userName] = text
        sendMessage(c, 'Your password saved.')
    else:
        registerChecker = 0
        c.send(bytes([registerChecker]))
        sendMessage(c, 'Your username is ' + userName + ', please write your password to login: ')
        lengthOfText = int(c.recv(1)[0])
        text = c.recv(lengthOfText).decode('utf8')
        if user_password[userName] == text:
            passwordChecker = 1
            c.send(bytes([passwordChecker]))
        else:
            passwordChecker = 0
            c.send(bytes([passwordChecker]))
            sign -= 1
            if sign == 0:
                full = False
            c.close()

    sendMessage(c, 'Waiting for other players to connect!')
```

Main function of server.py.

```
if __name__ == '__main__':
    ip = "127.0.0.1"
    port = 1234

    serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print('Server Host: ' + ip + '| Port: ' + str(port))

    try:
        serverSocket.bind((ip, port))
    except socket.error as e:
        print(str(e))
    serverSocket.listen()

    c, address = serverSocket.accept()
    print("Connection " + str(sign + 1) + " established from: " + str(address))
    sign += 1

    c.send(bytes([sign]))
    totalNumberOfPlayers = int(c.recv(10).decode('utf-8'))

    start_new_thread(Thread, (c, sign))

    while True:
        while sign < totalNumberOfPlayers:
            c, address = serverSocket.accept()
            print("Connection " + str(sign + 1) + " established from: " + str(address))
            sign += 1
            c.send(bytes([sign]))
            if sign == 1:
                totalNumberOfPlayers = int(c.recv(10).decode('utf-8'))
            start_new_thread(Thread, (c, sign))
        continue
```