ISTANBUL TECHNICAL UNIVERSITY
COMPUTER COMMUNİCATİON
PROJECT-1

Tugay ACAR

150160511

Computer Engineering

Explanation of Project

1. How to run

-I used cmd for Windows.

-Path to your python.exe

-Then write console in order like that :

python C:path\to\Server.py

python C:path\to\Client.py

-(you can write Client.py infinitely)

2. Explanation of coding

```
def __init__(self, Port):
    # exceptions for creaitng of socket
    try:
        serverSocket = socket(AF_INET, SOCK_STREAM)
    except:
        exit(1)

    try:
        serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
    except:
        exit(1)

    try:
        serverSocket.bind(('', Port))
    except:
        exit(1)

    try:
        serverSocket.listen(45)
    except:
        exit(1)

    print("The server is ready")
```

Figure 1

In the figüre 1 : The init part in Server.py try to catch errors. If the server don’t has error, prints “The server is ready”

```
client
while
    #threading a new thread for every client who is joined
    connectionSocket, addr = serverSocket.accept()
    #add clients to our list
    client.add_client(connectionSocket,addr)
    threading.Thread(target=self.listen_Client, args=(connectionSocket, addr, client)).start()
    time.sleep(1)
```

PEP 8: block comment should start with '# '

Figure 2

İn the Figure 2: Add our clients to server and thread them. Also, i keep all clients in class.list()

```
def receive():
    #to reveive messages from server
    while True:
        try:
            msg = clientSocket.recv(1024)
            print(msg.decode('utf-8'))
        except OSError:  # Possibly client has left the chat.
            break

def send():
    #to send messages to server and then the other clients
    msg = input('')
    timestamp = ' '+str(time.ctime(time.time()))
    msg = msg + timestamp
    clientSocket.send(msg.encode())
    if msg == "{quit}":
        clientSocket.close()
        exit(0)

while True:
    #i wrote an algorithm like that
    if write_flag==1:
        send_thread = Thread(target=send)
        send_thread.start()
        time.sleep(1)
        write_flag=0
        read_flag=1
        continue
    if read_flag==1:
        receive_thread = Thread(target=receive)
        receive_thread.start()
        time.sleep(1)
        read_flag =0
        write_flag = 1
        continue
```

Figure 3

In the Figure 3: Send and Receive messages between clients and server.

```
#Send messages to all of other clients
def broadcast(msg, clients, conn):
    for i in clients.list_sockets:
        if conn != i:
            i.send(msg.encode())

class Server():
    def listen_Client(self, connectionSocket, addr, cc):
        #first i keep a nickname to understand who send message
        nickname = connectionSocket.recv(1024)
        name[connectionSocket] = nickname.decode('utf-8')
        #send a welcome message to only itself
        welcome_message = nickname.decode('utf-8') + ', Welcome to chat room '
        connectionSocket.send(welcome_message.encode())
        #send joining message to all of other clients
        message = nickname.decode('utf-8') + ', joined chat room '
        broadcast(message,cc,connectionSocket)
        while True:
            #server takes an input
            data = connectionSocket.recv(1024)
            # if data(message) != exit, it sends the data to all of other client
            if data != bytes("exit", "utf8"):
                # we have just one client, the client is alone in server. So i s
                if (len(cc.list_sockets) == 1):
                    msg = 'You are alone in server'.encode()
                    connectionSocket.send(msg)
                #if we have more clients than 1, sen messages to other clients
                data = name[connectionSocket]+': '+data.decode('utf-8')
                broadcast(data, cc,connectionSocket)
            else:
                #if message exit. Client exit
                connectionSocket.send(bytes("exited", "utf8"))
                connectionSocket.close()
                cc.list_sockets.remove(connectionSocket)
```

c

Figure 4

In the Figure 4: Server.py takes input from 1 client and send them other clients(broadcast)

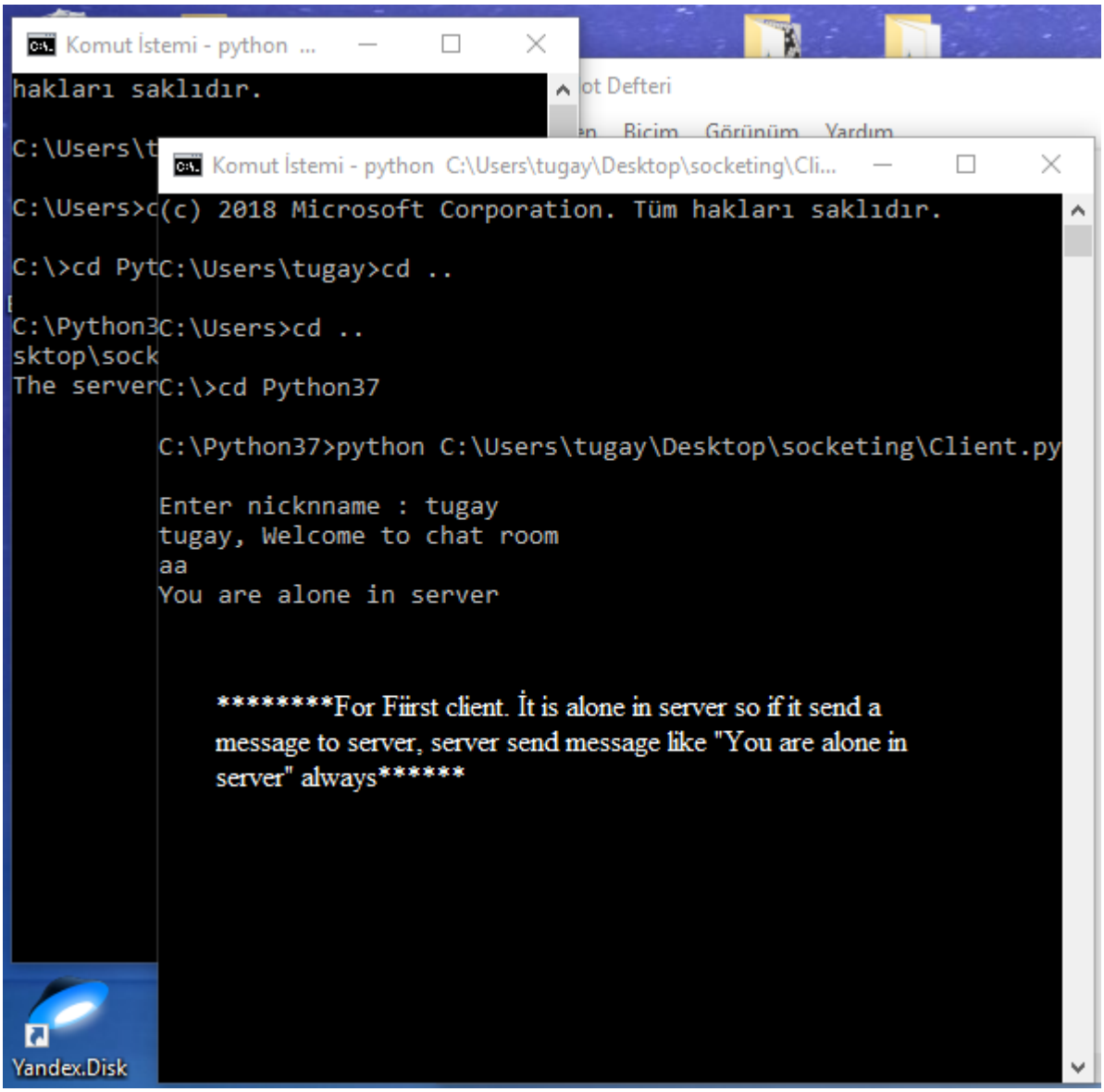

Figure 5

Figure 5: The first client joins the server. The client is alone in server. So server always send message like that “You are alone in server” ,if client send message to server.Takes a nickname and keep it.

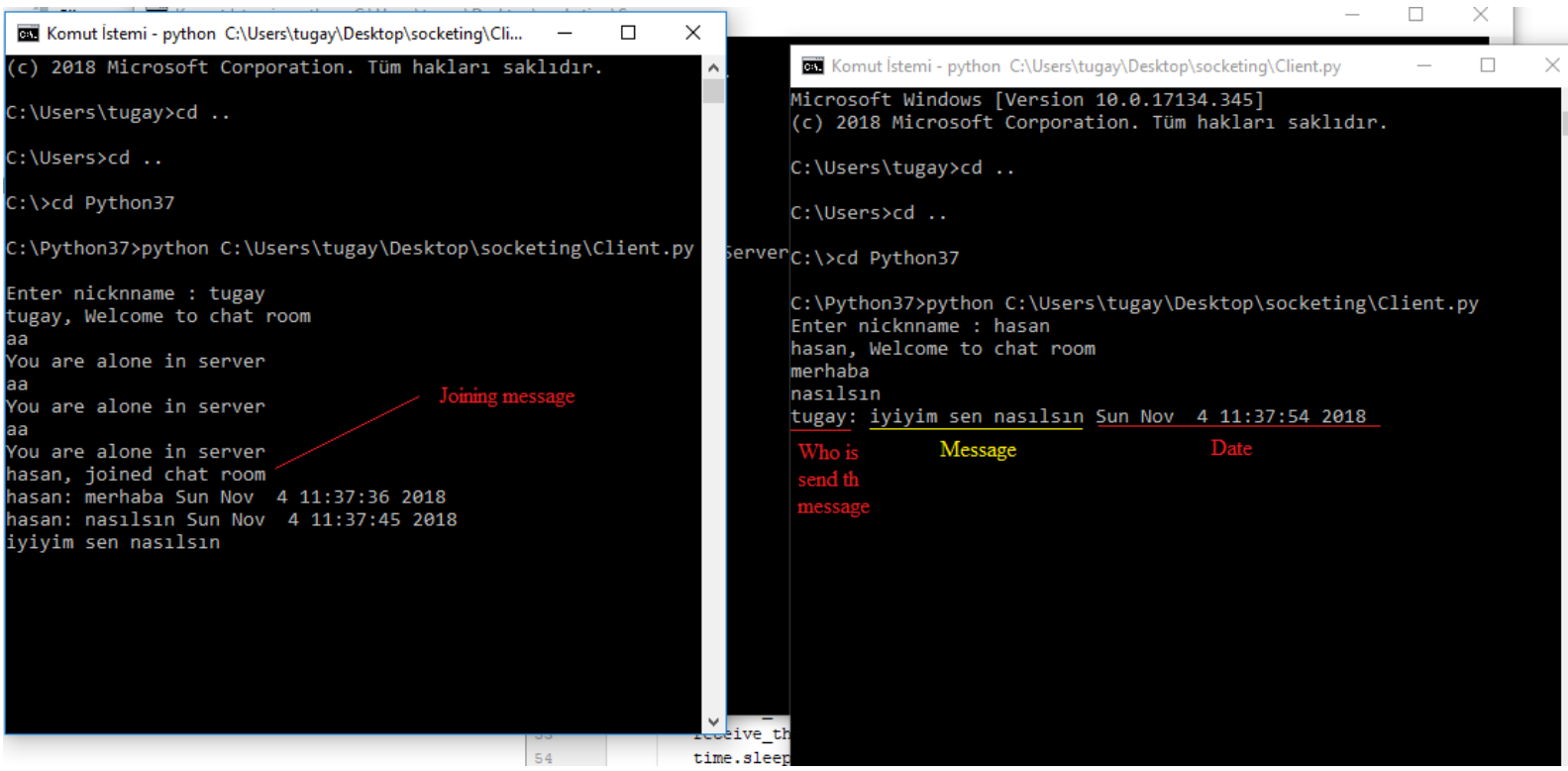

Figure 6

Figure 6: The second client joins the server. Firstly takes a nickname then server send client welcome message. After that, Server send joining message of Client 2(hasan) to other clients. If anyone send a messge, it will be transmitted to other clients as like that nickname + message + timestamp

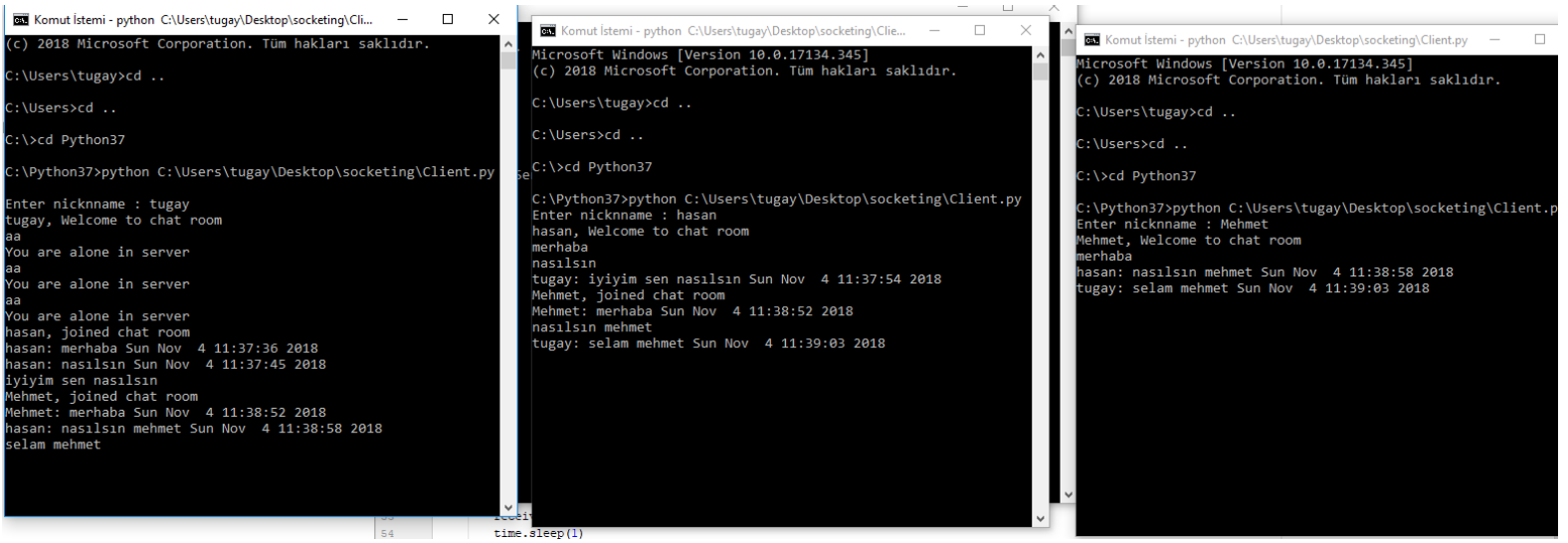

Figure 7

Figure 7: Same procedure with Figure 6. Takes a nickname, server send client welcome message and then send joining message to other clients. İf one of clients writes a message, server send them to all of other clients.