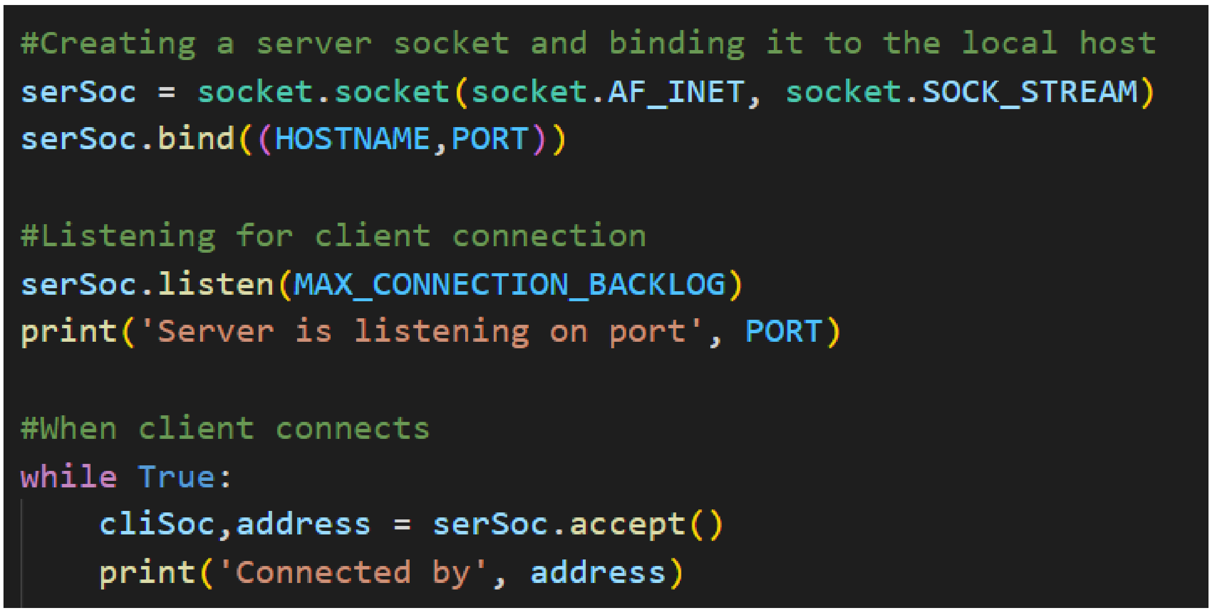
COMPENG 4DN4
LAB 2

Ahnaf Bhuiyan - Bhuiya3 - 400198359
Dwip Patel - pated15 - 400190154
Arnab Dutta - duttaa3 - 400183053

All group members contributed equally. As a group we worked through each part of the assignment together, discussing each issue as a group.

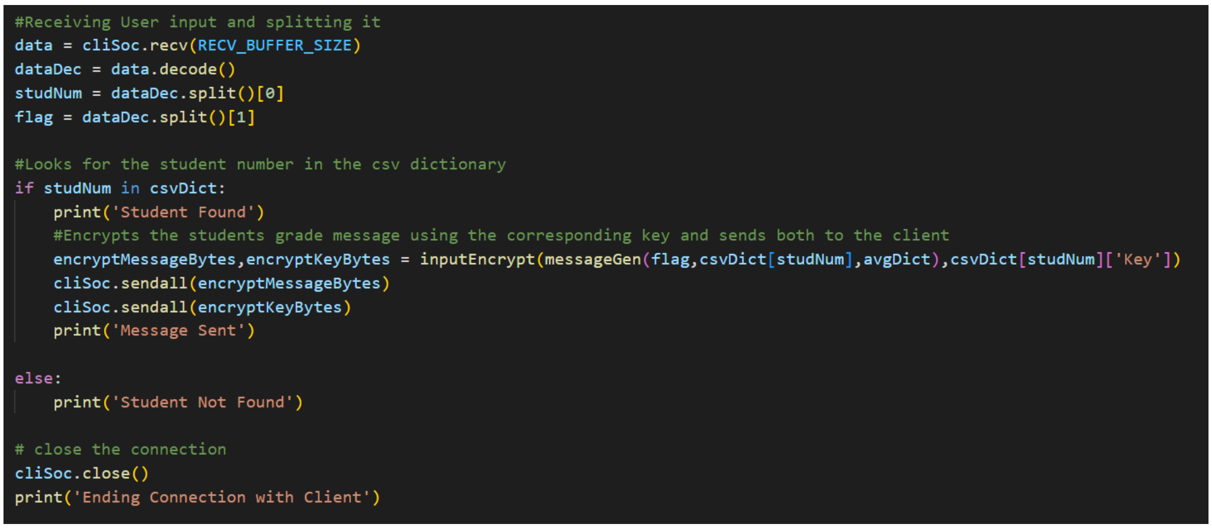
Server

To set up the server, a simple socket listen and connection process was implemented. Once the server finds the client, it accepts the connection and is now ready to take input.



```python
#Creating a server socket and binding it to the local host
serSoc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serSoc.bind((HOSTNAME,PORT))

#Listening for client connection
serSoc.listen(MAX_CONNECTION_BACKLOG)
print('Server is listening on port', PORT)

#When client connects
while True:
    cliSoc,address = serSoc.accept()
    print('Connected by', address)
```

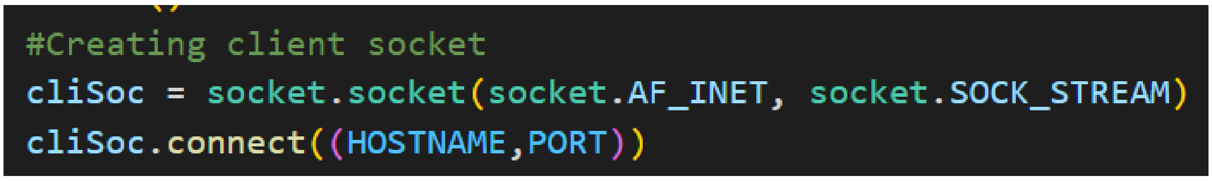
After the server is set up, it then takes the student ID and the wanted average from the user. Once receiving the input, the server then searches the created dictionary filled with student information to check if the given student number exists in the class. If it does, the server then creates the message with the average of the inputted assessment. The message is encrypted using the given key for the student and fernet. Once encrypted, the message and key are sent to the client to be output into the terminal.



```python
#Receiving User input and splitting it
data = cliSoc.recv(RECV_BUFFER_SIZE)
dataDec = data.decode()
studNum = dataDec.split()[0]
flag = dataDec.split()[1]

#Looks for the student number in the csv dictionary
if studNum in csvDict:
    print('Student Found')
    #Encrypts the students grade message using the corresponding key and sends both to the client
    encryptMessageBytes,encryptKeyBytes = inputEncrypt(messageGen(flag,csvDict[studNum],avgDict),csvDict[studNum]['Key'])
    cliSoc.sendall(encryptMessageBytes)
    cliSoc.sendall(encryptKeyBytes)
    print('Message Sent')

else:
    print('Student Not Found')

# close the connection
cliSoc.close()
print('Ending Connection with Client')
```

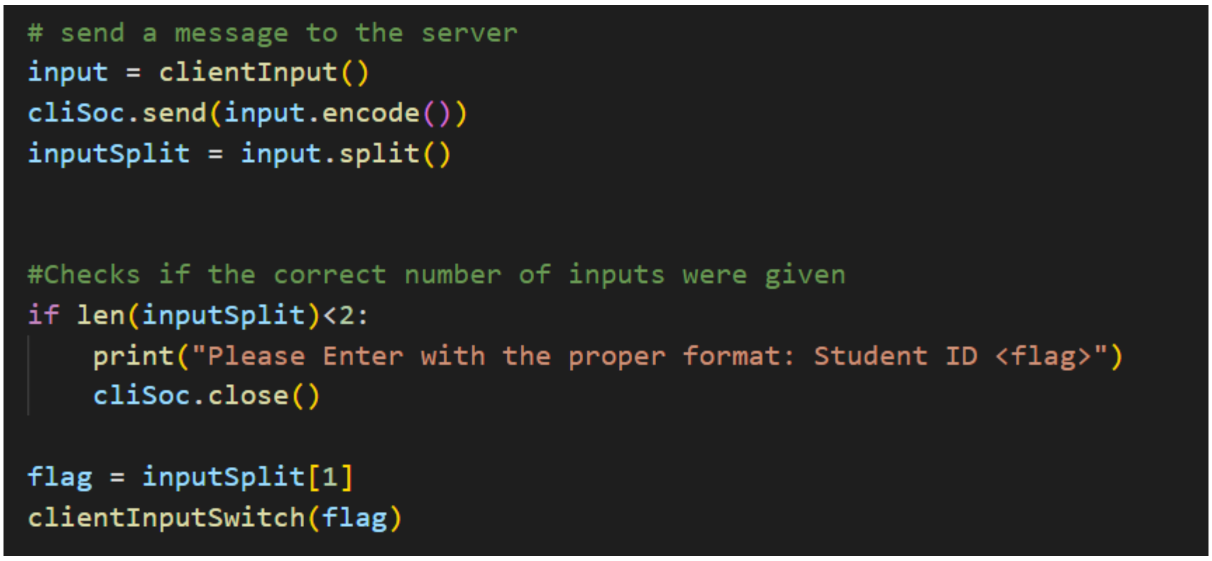
Client

The client starts by creating the socket and attempting to connect it to the server.



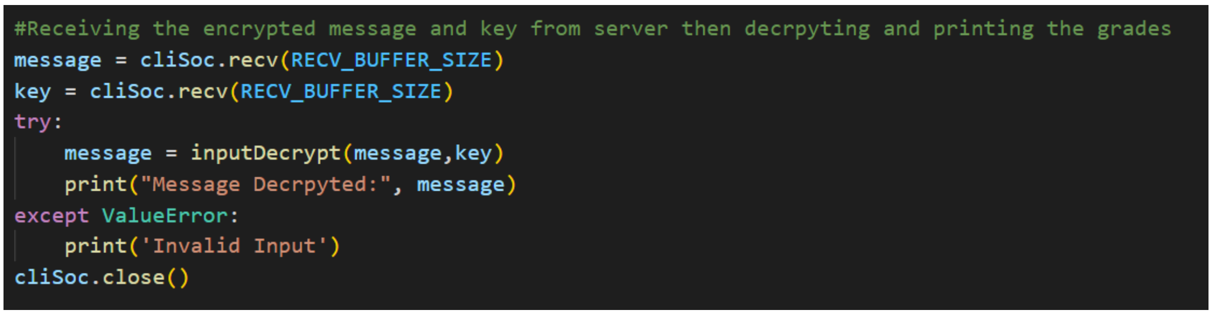
```python
#Creating client socket
cliSoc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
cliSoc.connect((HOSTNAME,PORT))
```

Afterwards the client takes the user input, sends it to the server and splits it in order to find what fetch message to print. It also checks whether the input was given in the proper format.



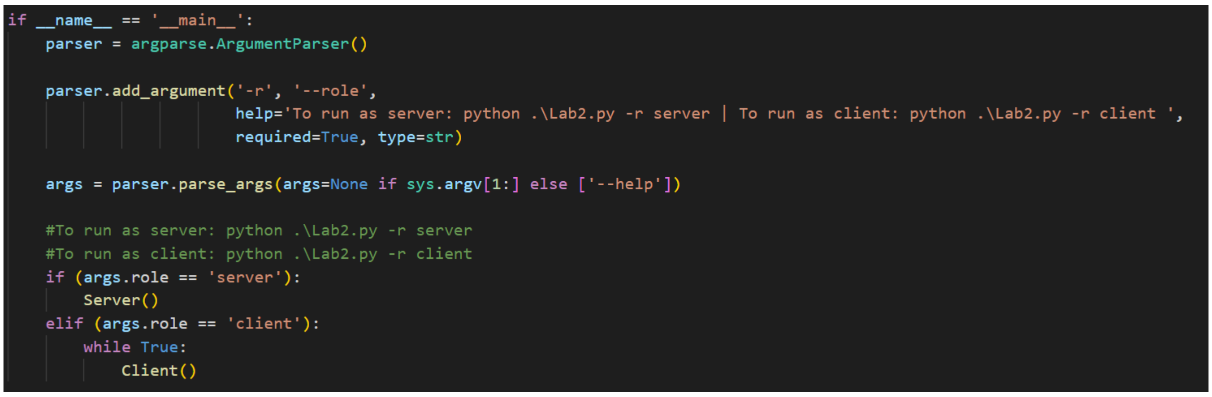
```python
# send a message to the server
input = clientInput()
cliSoc.send(input.encode())
inputSplit = input.split()


#Checks if the correct number of inputs were given
if len(inputSplit)<2:
    print("Please Enter with the proper format: Student ID <flag>")
    cliSoc.close()

flag = inputSplit[1]
clientInputSwitch(flag)
```

The client waits for the server to send the message and key. Once receiving them, it decrypts the message using fernet and the key. If the inputted data is in the correct format, the message will be successfully decrypted and printed on to the terminal.



```python
#Receiving the encrypted message and key from server then decrpyting and printing the grades
message = cliSoc.recv(RECV_BUFFER_SIZE)
key = cliSoc.recv(RECV_BUFFER_SIZE)
try:
    message = inputDecrypt(message,key)
    print("Message Decrpyted:", message)
except ValueError:
    print('Invalid Input')
cliSoc.close()
```

To set the mode which the script will run (server/client), an argparser is implemented.



```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()

    parser.add_argument('-r', '--role',
                        help='To run as server: python .\Lab2.py -r server | To run as client: python .\Lab2.py -r client ',
                        required=True, type=str)

    args = parser.parse_args(args=None if sys.argv[1:] else ['--help'])

    #To run as server: python .\Lab2.py -r server
    #To run as client: python .\Lab2.py -r client
    if (args.role == 'server'):
        Server()
    elif (args.role == 'client'):
        while True:
            Client()
```

Within the server and client functions, there are many helper functions which aid in fulfilling the requirements. Such functions include:

- CSV scraper and dictionary generator
- Message generator
- Message encryptor and decryptor
- Client input receiver
- Switch cases depending on the user input

Although these functions do not work on TCP clients or servers themselves, they help with organizing the code and make it easier to understand.