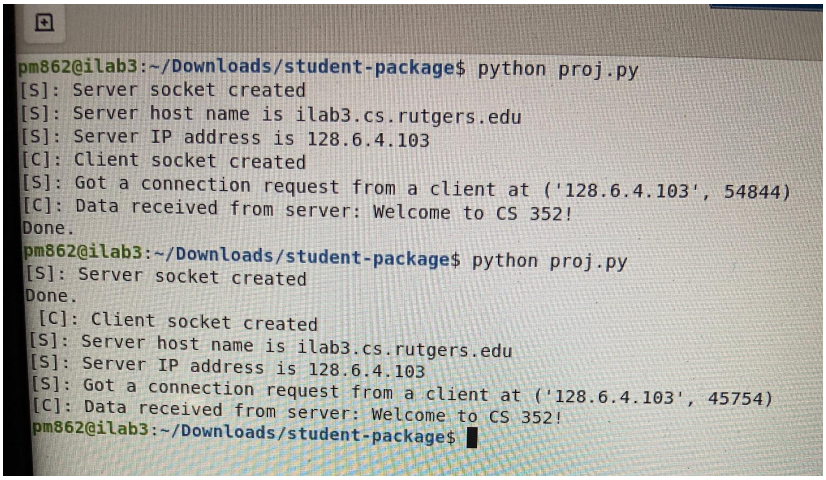
Important Note: Once the server and client files are run in the terminal, the output files will be generated in the respective folder. However, if the output files are not found, then please refresh the folder or exit and come back to that folder, and the output files should appear. Thank you!

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).
    a. Palak Mehta - pm862
2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.
    a. I did not collaborate with anyone else on this project, but I asked my recitation TA several clarifying questions on what the requirements of the project were. Part 3,4, and 5 were harder to understand as the steps overlapped with each other, so the clarification was really helpful. I also used https://docs.python.org/3/library/socket.html, https://realpython.com/python-sockets/, and the recitation 2 slides that the TA went over during lecture to understand how socket programming works. I also used webpages from python.org for python syntax references. The webpages I mentioned on sockets gave a background of sockets and client and server interaction, proving to be helpful in order to comprehend how to approach the project. The other python webpages were used to search for functions to reverse the given strings and turn the strings to upper case as well as how to initialize lists and manipulate them.
3. What did you observe after running step 2 above? Can you explain why you see what you see?
    a. I saw that the chronological order of the executed statements changed, meaning that eliminating the time.sleep statements gives less of a buffer time between statement executions. The server and client sockets are created back to back, meaning the files connection was acknowledged a lot quicker than if the sleep statements weren't there. For example, the image below shows the two different outputs. The first time I ran the file in the terminal, it was with the time.sleep() statements. The second time I ran the command, it was without the time.sleep() statements. As can be seen the Server and client print statements' ordering is altered.



4. You should name the output file from step 3 as out-proj.txt
    a. Done.
5. For each cases in step 4 please name the output file as outr-proj.txt (reversed string), and outup-proj.txt (upper case). Your output files must match exactly with the ones shown.
    a. Done.
6. Is there any portion of your code that does not work as required in the description above? Please explain.
    a. All the terminal outputs and the output files are as expected and the code executes the requirements of the project. In other words, the client is correctly reading and sending the messages. The server is then receiving and updating the message to the same, reversed, and uppercase. The server is then writing these updated messages into the respective output files and sending the updated messages back to the client. The client is then receiving these messages. However, one thing that is not necessarily mentioned in the project description that I did is printing out the updated (part3, reverse, and uppercase) messages in both the client and server. I noticed that if I did not print the messages in the server, then when I print the updated messages in the client, some messages all print in one line instead of separately. By printing the messages in both the client and server terminal, all the output files and terminal outputs are as expected.
7. Did you encounter any difficulties? If so, explain.
    a. Originally, my code was not working for part 5. I realized that I was rereading the input file's contents three times and this messed up the output. Some output files would have 6 lines, whereas another output file would have nothing. I realized that instead of repeatedly reading the file, I could read the input file's content once in the client file, pass those contents to the server file and store these messages in a list. Then, I can complete steps 4 and 5 without having to read the input file multiple times. This prevented any logic errors that I had in my code since now I can work with a list after sending and receiving the messages in the client and server respectively.

8. What did you learn from working on this project? Add any interesting observations not otherwise covered in the questions above. Be specific and technical in your response. Contact the course staff on Piazza if you have any questions.
    a. Another thing I also learned that was new from this project was that the accept() function creates a new socket when invoked. This is important because that is the socket that is then used to communicate with the client. It is different from the socket that is only listening and accepting new connections. Also, I learned about the process of sending and receiving communication between the client and server. This concept is new to me so initially I was sending and receiving the messages multiple times which caused several errors in my code as some messages were not being received in the right order. To remedy this, I decided to create an array that stores the contents of the input file and then perform the cases to update the messages using that array, rather than repeatedly sending and receiving the messages. This also helped minimize the latency of my program. Overall, I learned that there are multiple ways of approaching any problem. Often, the first solution is not effective so finding a solution that eliminates redundancy is most effective. By not repeatedly reading the input file, I avoided getting confused and messing up the different outputs. I also learned that keeping clean code is really important. Initially, I threw all my ideas and code in the files, but did not organize it. After coming up with another solution, I decided to organize my code better which helped me understand the goals and what I was doing wrong if there were issues.