Report for Project #1: Warmup: Socket programming 101

1. Team Members:

   Yuchen Wei (yw780) and Tiange Zhang (tz196)

2. Collaboration:

   We use GitHub to collaborate between the two of us, updating and communicating the parts we have completed and the changes and improvements we have made to the code. The primary reference materials we used in the process were from previous Python programming courses we participated in and our experience in learning Python independently. We divided parts 1-4 of the project, as well as the fifth part and the report, into two people's jobs to complete and share the problems encountered in the process.

3. What did you observe after running step 2 above? Can you explain why you see what you see?

   After removing the two statements of time.sleep(), the program starts to give exceptions in both threads, client and server. The exception usually occurs after both fuctions prints "socket created" and a "Done" in random orders. Sometimes the program runs successfully, but the content printed is still disordered, that "Done" is printed in the middle of the procedure.

   From the code, we can see that the server function is a bit more complicated than the client function, thus may take longer time to proceed. As a result we think when the first sleep statement is removed, what happened on those two threads which runs simultaneously, the client function has already started to request data from the server, but the server function is yet prepared to send any message, and thus gives the error.

   In addition, because both functions are still relatively small, it is sometimes possible that the server function finishes earlier than the client, and the program would run successfully. However, due to the lack of the second sleep statement, "Done" is printed immediately after the client thread starts, which is why we would see “Done” printed in the middle of the procedure.

4. Is there any portion of your code that does not work as required in the description above? Please explain.
   After comparing the results of our tests with the samples included in the project and the description, we believe that our code fulfills all the requirements of the description and can be reflected by the tests.

5. Did you encounter any difficulties? If so, explain.
   We found that the final output of the txt file would automatically have a blank line and then another line of text when we were working on part 5. At first, we tried to use print to check the contents of the reversed file to ensure it was fully functional before outputting it into the file. This test passed without a problem; then, we tried to output it to a txt file and found that although the content was the same as the sample, there was a formatting problem. Yuchen Wei soon realized that the original input txt file was being read with a "/n" line switch code, and after being inverted, it was inverted to keep a blank line before outputting the next inverted text. We then solved the problem by removing the line break "/n" before the server passing it to the client and then reversing and outputting it.

6. 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.
   We learned from Step 2 that it is crucial to synchronize or wait for progress while using multiple threads. Especially when one of the threads require information or results from other threads. Sleep is a good option, but not reliable enough. We believe that there are better ways to synchronize progress in each thread, and hualt and wait if needed.

   Through reading the given code, We also learned how to make two hosts communicate with each other via socket in python, though how each line of code works is still a bit unclear. We learned that when sending and receiving string, we need to encode the string at sending and decode at receiving. Otherwise, what you would print would be string that makes no sense to human.