TED UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

CMPE 472

Computer Networks

Section 03

Batuhan İşcan 11863132364

Table of Contents

Files

The server and the client are run from the python files with the corresponding names. I have used the socket, random and pandas modules in working with this assignment.

The server and client were also ran through a Jupyter Notebook as a means of visualising and testing the codes.

The necessary code descriptions were included as comment lines.

Code snippets

I. server.py

A. Imports

```
import socket
import random
import pandas as pd
```

B. handle_request function

1. Definition

```
def handle_request(connection, address, real_temp):
    #This method should process the client's guess
    #and respond with the appropriate message based on the guess accuracy
    attempts = 0
    # The accepted margin of error is calculated to refrain from calculating them multiple times during the loop
    ten_percent = real_temp * 0.1
    # The attmepts count is initialized as 0 on every subsequent connection to a client
    # The while loop is used to limit the attempts to 3
    while attempts < 3:
        # The server receives the guess sent by the client
        guess = connection.recv(1024).decode()
        # If the client sends "END", which is case insensitive, the connection is severed and
        # the handle request function will return "TERMINATE" which will terminate the server
        if guess.upper() == "END":
            connection.send("Server shutting down.".encode())
            connection.close()
            return "TERMINATE"  # Could also use a boolean value
```

2. Try catch block

```
    # The try catch block is in case the guess is not a number, which would cause an error trying to strip it as a float
    try:
        guess = float(guess)
        # If the guess is exactly correct, there is a special success message
        if abs(guess - real_temp) == 0:
            print(f"Client {address} guessed {guess}, which was the correct temperature.")
            connection.send("Exactly correct!".encode())
            break
        # If the guess is within the 10% tolerance range, the server sends a message of approval
        # The connection between server and client is severed and the while loop is broken
        elif abs(guess - real_temp) <= ten_percent:
            print(f"Client {address} guessed {guess}, which was accepted within the 10% tolerance range of {real_temp}.")
            connection.send("Correct!".encode())
            break
        # The attempts count is incremented before the hint is sent to the client
        attempts += 1
        # If the attempts reached 3, the real temperature is sent to the client
        if attempts >= 3:
            connection.send(f"Temperature was {real_temp}".encode())
            break
        # If the loop was not broken by now, the hint is sent to the client based on the guess being bigger or smaller than the real temperature
        hint = ""
        if guess > real_temp:
            hint = "Lower"
        else:
            hint = "Higher"
        connection.send(hint.encode())
    # The error is caught and handled, the loop asks for a new guess
    except ValueError:
        connection.send("Invalid input.".encode())
# The connection is closed after the loop is broken
connection.close()
# Notification in the server that a connection was closed
print(f"Connection from {address} closed\n")
```

C. serve_forever function

```python
def serve_forever():
    #In this method, load the weather data, randomly select a city,
    #and wait for the client to guess the temperature of the chosen city

    # The table is loaded, the rows are made into lists
    weather_data = pd.read_excel("./weathers.xlsx")
    cities = weather_data.City.tolist()
    temperatures = weather_data.Temp.tolist()
    # The server is initialized as localhost on port 8888 as specified on the requirements
    HOST = "localhost" #'127.0.0.1'
    PORT = 8888

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((HOST, PORT))
    server.listen(1)
    # Affirmation that the server listening has commenced
    print(f"Server listening on HOST: {HOST}, PORT: {PORT}")
    # Since the function is supposed to "serve forever", the while loop runs indefinitely, accepting connections from clients
    while True:
        connection, address = server.accept()
        # Notification in the server that a connection with a client was established
        print(f"Connection from {address} established\n")

        # Randomly select a city and its temperature
        # Picks a random city for each connection by a client
        city = random.choice(cities)
        real_temp = temperatures[cities.index(city)]

        # Send the city name to the client, asking for a guess
        connection.send(f"Predict the temperature in {city}:".encode())
        # The guesses are handled in the handle_request function
        result = handle_request(connection, address, real_temp)

        if result == "TERMINATE":
            print(f"Client {address} requested to end the session.")
            break  # Exit the loop to terminate the server
    server.close()  # Close the server socket

# The running of the file calls the serve_forever function
if __name__ == '__main__':
    serve_forever()
```

II. client.py

A. main function

```python
import socket

def main():
    #Connect to the server, receive the city information
    #prompt the user for a temperature prediction and send it to the server
    # Establish the client and connect it to the server by the same host and port
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(("localhost", 8888))
    # The client halts until receiving a message, which is the city name with the question that is sent when a client connects
    message = client.recv(1024).decode()
    # The client console shows the message
    print(message)
    # The while loop allows for the client to guess until the server ends the connection or the client types "END"
    while True:
        guess = input("Enter your guess (or type 'END' to quit): ")
        # The guess is sent to the server
        client.send(guess.encode())
        # The "END" command is case insensitive, the while loop is broken if TRUE and the client is closed after the loop
        if guess.upper() == "END":
            break
        # The client listens for the server's response to its guess
        response = client.recv(1024).decode()
        # The list of responses without variations were put in a list of lambda functions to lessen the conditionals
        responses = {
            "Exactly correct!": lambda: print("Congratulations! You guessed the correct temperature."),
            "Correct!": lambda: print("Congratulations! You guessed the correct temperature."),
            "Lower": lambda: print("Try a lower temperature"),
            "Higher": lambda: print("Try a higher temperature"),
            "Invalid input.": lambda: print("Invalid input. Please enter a number."),
        }
        # The reresponse is shown to the client, if the guess is successful or ends in a failure, the loop is broken
        if response.startswith("Temperature was"):
            print(response)
            break
        elif response in responses:
            responses[response]()
            if response == "Correct!":
                break
    # The client is closed after the loop is broken
    client.close()
# The running of this file calls the main method by the magic method
if __name__ == '__main__':
    main()
```

III. Terminal Usage Examples

A. Upon the initialization of the server, the server prints out the affirmation that it is listening

```
PS C:\Users\user\Documents\VSC Projects\CMPE 472\PA1> & C:/Users/user/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/user/Documents/VSC Projects/CMPE 472/PA1/server.py"
Server listening on HOST: localhost, PORT: 8888
```

B. Upon connection from a client, the server prints a notification of the connection and sends the client a random city, asking them to predict the temperature

```
PS C:\Users\user\Documents\VSC Projects\CMPE 472\PA1> & C:/Users/user/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/user/Documents/VSC Projects/CMPE 472/PA1/client.py"
Predict the temperature in Chicago:
Enter your guess (or type 'END' to quit):
```

```
Server listening on HOST: localhost, PORT: 8888
Connection from ('127.0.0.1', 51465) established
```

C. Attempt higher/lower than the actual temperature

```
Predict the temperature in Athens:
Enter your guess (or type 'END' to quit): 32
Try a lower temperature
Enter your guess (or type 'END' to quit): 24
Try a higher temperature
Enter your guess (or type 'END' to quit):
```

D. Accepted by the tolerance range

```
Predict the temperature in Athens:
Enter your guess (or type 'END' to quit): 32
Try a lower temperature
Enter your guess (or type 'END' to quit): 24
Try a higher temperature
Enter your guess (or type 'END' to quit): 27
Congratulations! You guessed the correct temperature.
```

E. Failed after 3 tries

```
Predict the temperature in Athens:
Enter your guess (or type 'END' to quit): 15
Try a higher temperature
Enter your guess (or type 'END' to quit): 32
Try a lower temperature
Enter your guess (or type 'END' to quit): 25
Temperature was 28
```

F. Used the “END” command

```
Predict the temperature in Madrid:
Enter your guess (or type 'END' to quit): END
```

G. The connections and their terminations are listed as well as the response to the ‘END’ command on the server console