Python Final Project (v1.1)

creating a database server that communicates with clients representing a "water-stations"

The project will be made out of the the following four files:
server.py - represents the server
data.sqlite - SQLite 3 database used by the server to store data on the client (will be located in the same directory as the server)
client.py - Represents a single client. meant to be copied to different folders and executed on each.
status.txt - represents the data of the client, each status.txt is meant to be copied to the same folder as client.py and is meant to have different data inside (at least a different ID)

For the assignment you need to present at least three files: server.py, client.py & status.txt (the files can be zipped before sending)
As for data.sqlite, it can be generated by server.py on the first run as explained further below.

(see explanation for the client code in the next page)

Explanation for the Client Code

The client needs to run in a loop in which it reads data from its status.txt, connects to the server, and sends the data to the server, it needs to do so once every 60 seconds*

*(please use a global variable to store the number of waiting seconds so I can change it for project testing)

make sure the code closes status.txt after reading from it, this will allow changing the file from outside (using notepad/gedit) and the code will be able to read the changes. if you keep the file open, it might read the same data and won't see the changes.

status.txt will contain the data of a single water station using the following 3 lines

- the first line represents the station ID (some integer)
- the second line represent the state of Alarm1 (0 for OFF; 1 for ON)
- the second line represent the state of Alarm2 (0 for OFF; 1 for ON)

For example if the file contains the following lines:

```
123
0
1
```

Then it represents station ID 123, its first alarm is OFF and its second Alarm is ON.

The main concept is that while the client program is running, the user will be able to change the text manually using a program like notepad or gedit and save it in order to change what data the client will read in the next loop iteration.

If you want to have more clients, you copy client.py and status.txt to a different folder, change the ID and alarm status in the txt file (of the copy) and run the copy. repeat for each additional client you wish to have.

(note: alternatively, you can have multiples status files with different names like status1.txt status2.txt etc. and when you run the client before entering the loop ask the user what file to open, it will only open that file until the program is closed)

Tip for sending the data to the server.
The data is sent to the server using a socket. you can send it as an encoded string separating the values with space. like so:
"ID ALARM1 ALARM2"
for example we can send the following text to the server:
"123 1 0"

The server will then split the text into a list of 3 "words":
index 0 will contain the station_id
index 1 will contain alaram1 value
index 2 will contain alarm2 value

(see explanation for the server code in the next page)

Explanation for the Server

As one of the first actions of the server, it will open data.sqlite (this action will create the file if it didn't exist before) and then the server will create a table for the station data if it doesn't exist yet. You can use the following SQL Query which will make sure the table will only be created the first time the database file is created:

```
CREATE TABLE IF NOT EXISTS station_status (
    station_id INT,
    last_date TEXT,
    alarm1 INT,
    alarm2 INT,
    PRIMARY KEY(station_id) );
```

As can be seen above the database will contain a single table called station_status. and will contain the following columns:
:
station_id - for the station ID (will act as a primary key, meaning we only store a single line per station ID)
last_date - text representing the last date the station contacted the server
alarm1 - 0 or 1 representing if the alarm was on or off
alarm2 - 0 or 1 representing if the alarm was on or off

After creating the database (if it didn't exist) the server will enter a loop in which it will wait to receive client data and update the database each time it receives such data.

when setting the date (last_date) store it as text in the following format
`"YYYY-MM-DD HH:mm"`
So for example:
`"2019-01-20 10:30"`

Here is a sample code demonstrating how to get this formatted date string:

```
import datetime

last_date = datetime.datetime.now().strftime('%Y-%M-%d %H:%m')
print(last_date)
```

When receiving data from a station ID that doesn't exist yet in the database a new line will be **inserted**, when receiving data from a station ID that already exists its fields will be **updated.**

Normally this would require doing the following:

- use SELECT to check if the a line for that station ID exists
- if it exists, use an UPDATE query to change its alarm1 and alaram 2 to the received values from the client, and change its last_date using the method described above.
- if it doesn't exist, use and INSERT query to create a new line with the received ID, alarm1, alarm2 and the date string.

However, there's a method for doing both update and insert with the same query using **INSERT OR REPLACE:**

```
INSERT OR REPLACE INTO station_status
VALUES (?, ?, ?, ?)
```

Important notes:

- The server must be SQL-injection safe
- For testing purposes, the server socket will be binded to IP 127.0.0.1
- The server will use a single port
- add exception protections to your server code as required (make sure the server doesn't crash or hang just because a client closed connection or crashed)
- The server should be able to handle incoming data from several clients running at the same time each running from its own copy of client.py in a different folder, each with its own copy of status.txt.

Good Luck!