📌 [redacted]

🐍 Python SmartHome Coursework FAQ

📅 Last updated: 10/03/2025

❓ **Q: Do I need to include the testing code in my submission?**

✅ **A:** Yes! **Including your testing code is essential.** Without it, you **cannot demonstrate functionality**, which will result in **losing marks**.

📌 **Coursework Instructions:**
🛠️ **Testing for each task will be part of the demo session.**
🎯 You **must demonstrate your implementation's functionality.**
📂 Your test functions should be **well-structured** and verify key features **before submission**.

📝 **For Tasks 1 - 3:**
🔹 **Use print statements** in your test functions to indicate **what the test is doing**.
🔹 **Use the values provided in the coursework** to ensure your tests align with expected requirements.

❓ **Q: Where should error messages appear for Task 4 and 5?**
🖥️ **A:** If you're writing error routines for the **GUI**:
⚠️ **Errors must appear in the GUI,** not in the console.

❓ **Q: How many files should I split my submission into?**
📁 **A:** You can choose your approach. Here are some suggestions:

📌 **Option 1:**
📂 **Task 1 - 3** → `backEnd.py`
📂 **Task 4 - 5** → `frontEnd.py` (Similar to the worksheets)

📌 **Option 2:**
📂 Each class in **its own file**
📂 All **testing files in one file** (Common in Java, but may get messy)
🧩 **If using inheritance**: You may prefer to **keep the superclass and subclasses in one file**.

📌 **Option 3:**
📂 **All code in one file**, but ensure **each task can be tested separately** for the demo session.

❓ **Q: The given class diagrams don't include getter/setter functions, but if I create private variables in my classes, I would need them. Should I ignore this and follow the diagrams exactly?**

✅ **A:** When using **private instance variables**, you will need to go beyond the basic class diagram by adding **properties** (getters/setters) that are not explicitly shown. While the class diagram provides **a guideline**, it doesn't always detail implementation choices like private instance variables and properties.

❓ **Q: In my test functions, I'm using `try-except` to print error messages. Should the class itself raise `ValueError` instead, or is printing messages in the test function acceptable?**

✅ **A:** It is **not recommended** to print error messages inside the class because you **cannot use console output to display errors in a GUI**. Instead:

1. **Raise the error in the class code** using `ValueError` (or another appropriate exception).
2. **Use a `try-except` block in your test code** to catch the error.
3. **For the GUI, handle errors properly** by displaying messages within the interface rather than printing to the console.

❓ **Q: The GUI interface is glitching when updating the toggle on/off function and when adding or deleting appliances. Is this a general issue with Tkinter, or is it a problem in my code?**

✅ **A:** This is **likely an issue within your code rather than a general Tkinter problem**. While Tkinter has limitations, glitches in UI updates are usually caused by **incorrect state management, lack of proper widget updates, or inefficient event handling**.

❓ **Q: Can you add another test function to test the harder requirement separately?**

❌ **A:** No. Instead of creating a separate test function, **add the testing into the same function for the task** and use **print statements** to make it clear which part is being tested.

❓ **Q: Can we use a database for the challenge task to store multiple smart homes' data?**

❌ **A:** No. You **must use a flat file format** such as **CSV or JSON** instead of a database library.

📌 **Coursework Requirements:**

- **All data must be stored in a file** for persistent management.
- **CSV is the recommended format**, but **JSON or other structured file formats** may also be used.
- **Do not use SQL databases or external DB libraries** – all saving and loading must be handled through **file reading/writing operations**.
- The system must ensure that **smart home data is saved when the application exits** and **restored when reopened** using file-based storage

---