Python Assignment 1
Smart Home IoT Automation Simulator

Description:

In this assignment, you need to develop a Python-based IoT simulator for a smart home automation system. The simulator should emulate the behavior of various IoT devices commonly found in a smart home, such as smart lights, thermostats, and security cameras. You will also create a central automation system that manages these devices and build a monitoring dashboard to visualize and control the smart home. This assignment will help you apply your Python programming skills, including OOP, data handling, real-time data monitoring, and graphical user interfaces (GUIs).

Objective:

- Create a Python-based IoT device simulator, incorporating concepts from previous classes like data structures, object-oriented programming, variables, methods, exception handling, and more. Implement file operations, apply standard library functions, and develop custom modules and/or packages for modularity.
- Implement basic data analytics and processing capabilities.
- Design a dashboard for user interaction, and visualization sensor data.

Part 1: IoT Device Emulation (25 %)

- Device Classes: Create Python classes for each type of IoT device you want to simulate, such as SmartLight, Thermostat, and SecurityCamera. Each class should have attributes like device ID, status (on/off), and relevant properties (e.g., temperature for thermostats, brightness for lights, and security status for cameras).
- Device Behavior: Implement methods for each device class that allows for turning devices on/off and changing their properties. Simulate realistic behavior, such as gradual dimming for lights or setting temperature ranges for thermostats.

Part 2: Central Automation System (25 %)

- Automation System Class: Create a central automation system class, e.g., AutomationSystem, responsible for managing and controlling all devices. It should provide methods for discovering devices, adding them to the system, and executing automation tasks, such as activating lights automatically when motion is detected.
- Include a randomization mechanism to simulate changing device states and properties over time for automatic configuration.
- Gathering sensor data(temp, brightness, and/or camera data) and storing it in a file for future analysis.

Part 3: Monitoring Dashboard (20%)

- Graphical User Interface (GUI): Create a GUI for monitoring and controlling the smart home system. You can use Python GUI libraries like Tkinter. The GUI should display the status and properties of each device, provide controls to interact with them and visualize data.
- Real-time Data Monitoring: Display real-time data from the simulated devices on the dashboard. This may include temperature for thermostats, motion detection status for cameras, and brightness levels for lights.
- User Interaction: Allow users to interact with devices through the GUI, such as toggling lights on/off, adjusting thermostat settings, and arming/disarming security cameras.
- Automation Loop: Implement a automation loop that runs periodically (e.g., every few seconds) to trigger automation rules, update device states.

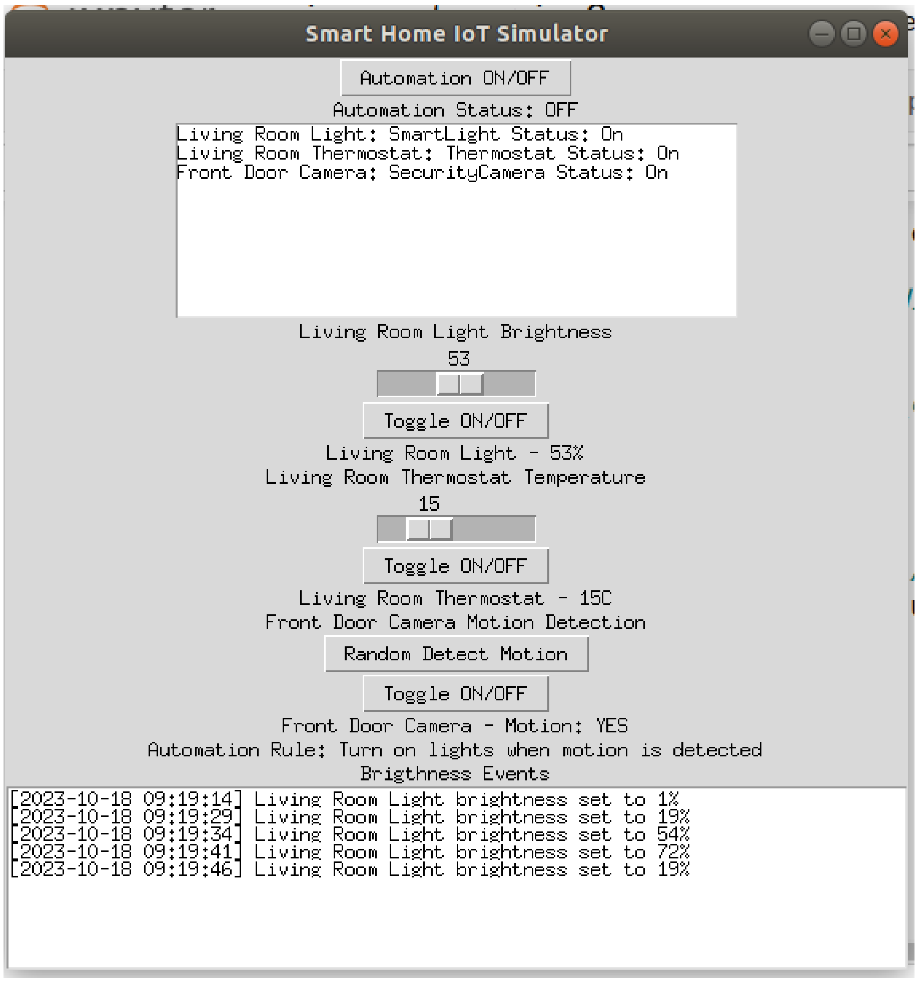
Part 4: Documentation (30%)

- Documentation: Provide clear documentation for your code, including class descriptions, method explanations, and instructions on how to run the simulation and use the dashboard.
- Develop test cases to ensure that the simulator and automation system behave as expected. Test various scenarios, such as different automation rules and user interactions.

💡 Hints:

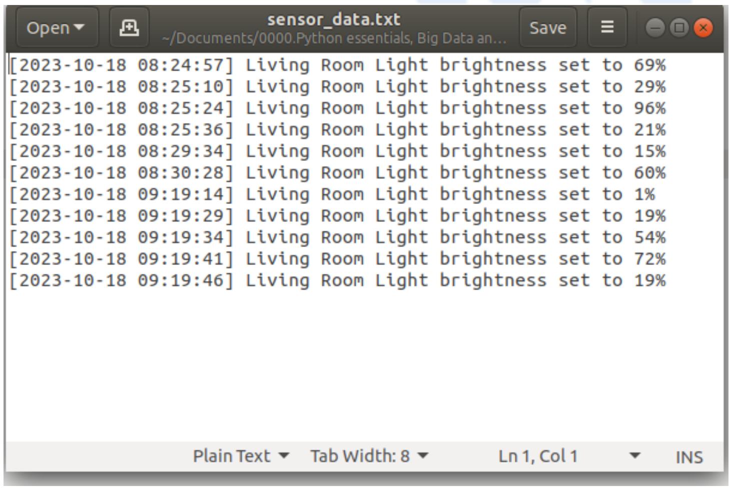
- Use Python's random library for generating sensor data.
- Utilize the statistics module for data analysis.
- Use a automation loop for automatic rule execution and automatic adjusting(create thread for concurrent execution)
- Timestamp sensor data with Python's datetime module.
- Design a GUI (try to be creative) using Tkinter.
- Document class attributes and methods.
- file handling for collecting sensor data.
- Feel free to use ChatGPT, but make sure to be able to understand and explain your solution!
- Do not hesitate to ask your instructor for help and ask for the template for the GUI!

Sample GUI:



```
Automation Status: OFF
Living Room Light: SmartLight Status: On
Living Room Thermostat: Thermostat Status: On
Front Door Camera: SecurityCamera Status: On
Living Room Light Brightness
53
Living Room Light - 53%
Living Room Thermostat Temperature
15
Living Room Thermostat - 15C
Front Door Camera Motion Detection
Front Door Camera - Motion: YES
Automation Rule: Turn on lights when motion is detected
Brigthness Events
[2023-10-18 09:19:14] Living Room Light brightness set to 1%
[2023-10-18 09:19:29] Living Room Light brightness set to 19%
[2023-10-18 09:19:34] Living Room Light brightness set to 54%
[2023-10-18 09:19:41] Living Room Light brightness set to 72%
[2023-10-18 09:19:46] Living Room Light brightness set to 19%
```

Sample sensor data:



```
[2023-10-18 08:24:57] Living Room Light brightness set to 69%
[2023-10-18 08:25:10] Living Room Light brightness set to 29%
[2023-10-18 08:25:24] Living Room Light brightness set to 96%
[2023-10-18 08:25:36] Living Room Light brightness set to 21%
[2023-10-18 08:29:34] Living Room Light brightness set to 15%
[2023-10-18 08:30:28] Living Room Light brightness set to 60%
[2023-10-18 09:19:14] Living Room Light brightness set to 1%
[2023-10-18 09:19:29] Living Room Light brightness set to 19%
[2023-10-18 09:19:34] Living Room Light brightness set to 54%
[2023-10-18 09:19:41] Living Room Light brightness set to 72%
[2023-10-18 09:19:46] Living Room Light brightness set to 19%
```