### **Lab 2: Decoding Sensor Data Using Python Virtual Subclasses**

**Exercise: Decoding Sensor Data using Python Virtual Subclasses**

Creating a lab exercise on virtual subclasses in Python for sensor data is an excellent way to teach students how to use virtual subclasses in practice, especially in scenarios where you want to make existing classes conform to an abstract base class (ABC) without changing their source code. In this exercise, Participants will create virtual subclasses to make different sensor data classes conform to a common interface.

**Exercise Description:**

Imagine you are developing a sensor data processing system for a smart home. Various types of sensors generate data, such as temperature sensors, humidity sensors, and motion sensors. To ensure a consistent interface for all sensor data types, we decide to use abstract base classes (ABCs) and virtual subclasses.

**Instructions:**

1. **Create an Abstract Base Class (ABC):** Begin by creating an abstract base class called SensorData with the following abstract methods:

* **get\_data(self):** An abstract method that simulates reading data from the sensor. This method should return a dictionary with sensor-specific data (e.g., temperature, humidity, motion status).

1. **Create Virtual Subclasses:** Implement virtual subclasses for different sensor data types (e.g., TemperatureData, HumidityData, MotionData) by using the @abstractmethod.register decorator.
2. **Implement Abstract Method:** In each of the virtual subclasses, implement the get\_data method to generate sensor-specific data according to the rules for each type of sensor data. For example, temperature data might include a "temperature" key with a floating-point value, humidity data might include a "humidity" key with a percentage value, and motion data might include a "motion\_detected" key with a Boolean value.
3. **Test the Classes:** Create instances of each of the virtual subclasses and demonstrate that the get\_data methods work as expected. Display the generated sensor data.
4. **Polymorphism:** Demonstrate polymorphism by creating a list of various sensor data objects (instances of the virtual subclasses) and calling their get\_data methods in a loop.

This exercise allows Participants to practice virtual subclasses, abstract methods, polymorphism, and class design while modeling different sensor data types in Python.