**Documentation: Smart AC Agent Class**

**Overview**

The Smart AC Agent class simulates an intelligent air conditioning (AC) agent. The agent reads room temperatures, stores them in a log file, and decides whether to **turn ON** or **turn OFF** the AC based on the room's current temperature compared to the desired temperature. Additionally, it can make decisions based on historical data (i.e., previous temperatures and actions).

The agent is designed to ensure that it maintains a comfortable temperature, as specified by the desired temp.

**Class: Smart Agent**

**Constructor: `init(self, desired\_temp, log\_file="temperature\_log.txt")**

This is the constructor method that initializes the Smart AC Agent.

* **Parameters**:
  + desired\_temp (int or float): The target temperature that the agent aims to maintain in the room.
  + log\_file (str, optional): The name of the log file where temperature data will be saved. The default value is "temperature\_log.txt". The file will be created if it doesn't exist.
* **Functionality**:
  + Sets the desired\_temp and log\_file as instance variables.
  + Ensures the existence of the log file (if the file doesn't exist, it will be created).

**Method: sense\_temperature(self, room\_temp, room\_name="")**

This method allows the agent to sense the current temperature of a room and record it in the log file.

* **Parameters**:
  + room\_temp (int or float): The current temperature of the room being sensed.
  + room\_name (str, optional): The name of the room where the temperature is being measured (e.g., "Living Room", "Bedroom"). The default value is an empty string.
* **Functionality**:
  + Sets the current room's temperature and name as instance variables (self.current\_temp, self.current\_room).
  + Appends the temperature reading (along with the room name) to the log file.

**Method: decide\_action(self)**

This method determines whether to **turn ON** or **turn OFF** the AC based on the current room temperature and the desired temperature. It also checks whether the current temperature has been logged before.

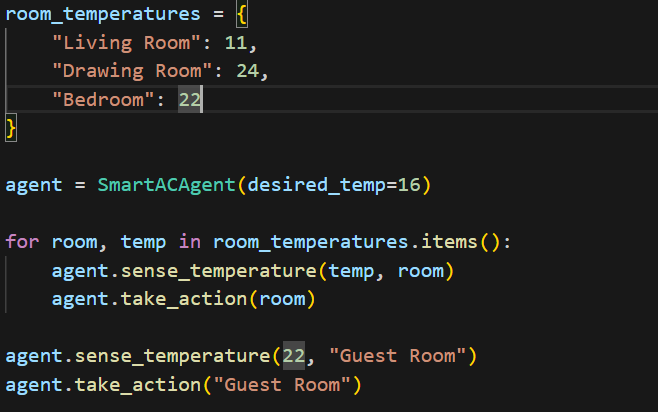
* **Returns** (str): The action the agent decides to take. This will either be:
  + "Turn ON the AC" if the current temperature is above the desired temperature.
  + "Turn OFF the AC" if the current temperature is below or equal to the desired temperature.
  + If the temperature has been seen before in the history, the action will be prefixed with **"(History)"**.
* **Functionality**:
  + Reads the temperature history from the log file.
  + Checks if the current room and temperature have been logged before.
  + Decides whether to turn the AC on or off based on the current temperature.
  + If the temperature has been seen before, the action is prefixed with "(History)".

**Method: take\_action(self, room\_name="")**

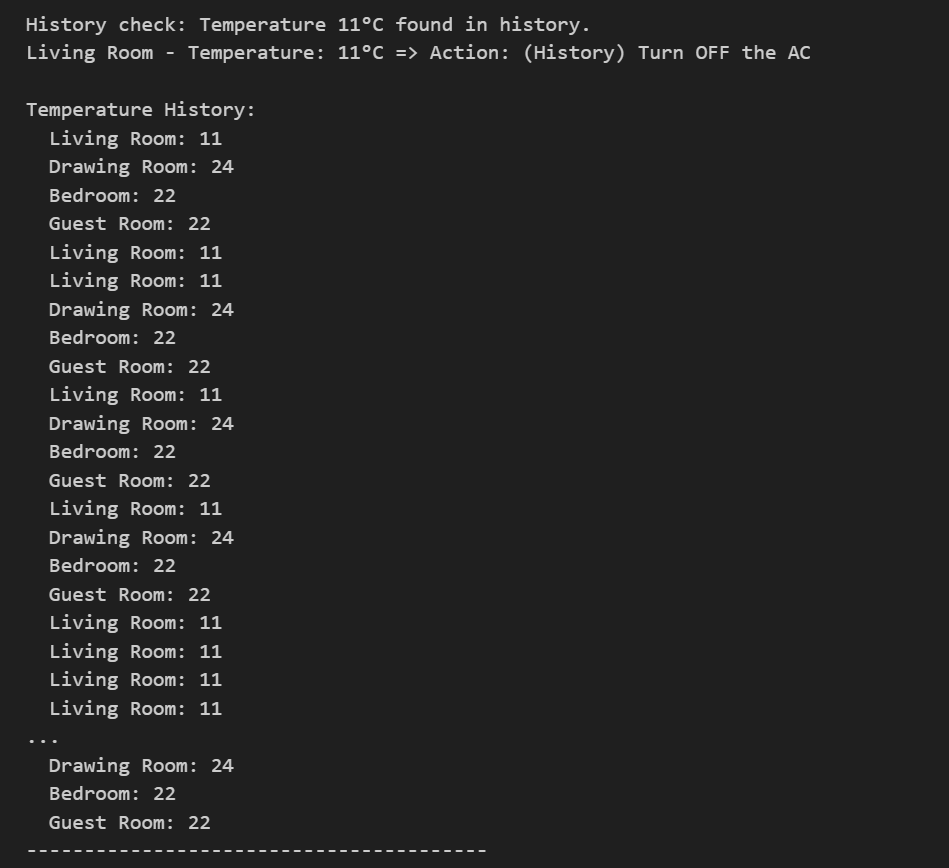
This method prints the decision made by the agent and shows the entire temperature history from the log file.

* **Parameters**:
  + room\_name (str, optional): The name of the room where the action is being taken. The default is an empty string.
* **Returns**: None (prints the action and history to the console).
* **Functionality**:
  + Calls decide\_action() to determine the action (whether to turn the AC on or off).
  + Prints the current temperature, the action taken, and the history of temperature readings for all rooms.

**Example Usage:**



**Expected Output Example:**

****

**Explanation of Output:**

1. **History Check**: The first few rooms are sensed, and the agent makes its decision to turn the AC on or off based on the desired temperature. When the Guest Room is sensed, the agent recognizes that the temperature (22°C) was recorded earlier and adds (History) to the action.
2. **Temperature History**: The output includes the temperature history for each room as logged in the file. The history keeps a record of each room’s temperature and whether the agent acted based on historical data.

**Key Concepts:**

* **Temperature Sensing**: The agent continuously senses the current room temperature and saves it for historical analysis.
* **Action Decision**: Based on the current temperature compared to the desired temperature, the agent makes the decision to turn the AC on or off.
* **History Usage**: The agent can refer to previously seen temperatures and append "(History)" to its actions when deciding based on past data.

**Limitations & Future Improvements:**

1. **History Check Accuracy**: The agent currently checks for exact room and temperature pairs. In cases where temperatures are close but not identical, the agent might miss out on meaningful trends.
2. **Temperature History Size**: The agent's history is stored in a plain text file. For large-scale applications or more sophisticated agents, a database might be better suited for storing and analyzing temperature data.
3. **Additional Features**:
   * Implement features like setting a temperature range (e.g., keeping the room temperature within a certain threshold).
   * Implement a more advanced decision-making algorithm based on multiple factors like time of day, room occupancy, etc.