Model-Based Reflex Agent

This document explains the working, concepts, and purpose of the ModelBasedReflexAgent implemented in the provided Python file. It highlights what is happening in the code, why certain steps are taken, and the professional reasoning behind each concept.

1. Concept of Model-Based Reflex Agent

* A Model-Based Reflex Agent improves decision-making by considering both the current state (sensor input) and memory of past actions.
* It uses stored knowledge (previous action) to avoid repeating unnecessary decisions.
* This type of agent is closer to real-world intelligent systems, as it adapts based on history and environment.

2. Class Initialization (\_\_init\_\_)

* Takes 'demanding\_temp' (desired room temperature) as input.
* Loads previously stored action from a file ('previous\_action.txt').
* Ensures persistence – so even after restarting the program, the agent remembers the last action.

3. Sensor Function

* Takes the current room temperature as input.
* Updates the agent’s perception of the environment.
* Acts as the sensory system of the agent, connecting the environment with the agent’s decision-making process.

4. Performance Function

* Compares current room temperature with the demanded temperature.
* If room temperature < demanding temperature → Action = 'Heater ON'.
* If room temperature ≥ demanding temperature → Action = 'Heater OFF'.
* Checks previous action to avoid redundancy. If the same action is required, it outputs: 'There will be no change and maintain the previous action'.
* Otherwise, it updates the action and saves it to the file for memory.

5. Actuator Function

* Executes the chosen action based on Performance function output.
* Displays both the current room temperature and the corresponding action.
* Acts as the output mechanism of the agent, interacting back with the environment.

6. Example Rooms and Execution

* A dictionary of rooms with their respective temperatures is provided.
* The agent checks each room temperature one by one.
* For each room, it senses the temperature, processes it, and performs an appropriate action.
* This demonstrates how the agent can be applied in real-world scenarios like smart buildings.

7. Why This Approach?

* Persistence ensures continuity in decision-making across multiple runs.
* Prevents unnecessary toggling of the heater, improving efficiency.
* Models real-world intelligent systems where history and context influence decisions.