Brief Explanation of the Approach

The objective of this assignment is to design an AI agent that does not behave perfectly from the beginning, but instead improves its behaviour over time by learning from its own mistakes. Keeping this goal in mind, I implemented a simple Research Question Answering agent using Python and LangChain.

The agent is designed to answer user questions. These questions can broadly be divided into two types: factual questions and non-factual questions. Factual questions are those that depend on external or real-world information, such as questions related to capitals of countries, founders of companies, or names of CEOs. For such questions, the agent is expected to use a search tool before generating the final answer. Non-factual questions, such as general explanations or opinion-based questions, can be answered directly without using any tool.

In order to demonstrate learning, the agent is intentionally kept naive during the initial runs. In the early executions, the agent may skip using the search tool even when it is required, use the tool but ignore the output it receives, or generate the final answer too early. These mistakes are not treated as errors in implementation, but are intentionally allowed so that the system can observe incorrect behaviour and learn from it.

A search tool is implemented using LangChain's tool interface. The tool simulates external information retrieval by returning predefined answers for known factual queries. This allows the agent to interact with a tool in a controlled and predictable manner, making it easier to evaluate whether the tool was used correctly or not.

After each run of the agent, an evaluation step is performed. This evaluation checks whether the agent followed the expected behaviour for the given question. If the question is factual and the agent did not use the search tool, the run is marked as a failure. Similarly, if the agent uses the tool but ignores the output and produces an unrelated answer, it is also considered a failure. Each run is clearly classified as either a success or a failure, along with the specific type of mistake made.

All outcomes from each run are stored in a simple memory structure. This memory keeps track of the run number, the question asked, the success or failure status, and the type of mistake if a failure occurred. The purpose of this memory is not just logging, but also to enable learning across multiple runs.

The learning mechanism is implemented using a basic feedback loop. The system periodically checks the memory to identify recurring mistakes. When the same mistake occurs multiple times, the agent's behaviour is adjusted by introducing a constraint. For example, when the agent repeatedly skipped the search tool for factual questions, a rule was added to force the

agent to always use the search tool for such questions in future runs. This change directly affects how the agent behaves in subsequent executions.

As a result of this feedback loop, the agent shows improvement over time. While early runs contain multiple failures due to incorrect tool usage, later runs show fewer occurrences of the same mistakes. This demonstrates that the agent is learning from past failures and adjusting its behaviour accordingly. The improvement is gradual and realistic rather than sudden, which reflects how learning systems typically behave in practice.

The primary focus of this approach is not achieving perfect answers, but clearly demonstrating a learning and improvement process. The system is designed to be simple, readable, and easy to understand, with explicit evaluation and feedback steps. One limitation of the current approach is that learning is rule-based and limited to specific mistake patterns. With further development, this could be extended to handle more complex scenarios, additional tools, and more flexible learning strategies.