FOUTEL Romane Group Team : Les AstICo

DE MORI VALENTINO Aurélie

PROMPT IA

Prompt 1 :

Comment modifier mon script pour qu'il lise plusieurs lignes JSON et renvoie une réponse pour chaque ligne, tout en conservant un état global entre les entrées ?

Réponse de l'IA : L'IA a suggéré d'utiliser sys.stdin pour un flux continu, mais a également précisé qu'une boucle sur un fichier texte permettait de traiter chaque ligne avec json.loads(). Elle a souligné l'importance de déclarer current_state à l'extérieur de la boucle pour assurer la persistance de la mémoire.

Notre utilisation : Nous avons adapté cette logique pour ouvrir directement un fichier JSON local via with open(...) et parcourir les lignes une par une afin d'automatiser nos tests.

Prompt 2 :

Intègre l'action suggest_pause dans ma boîte noire afin que l'état END puisse être atteint dans certaines combinaisons émotion + confiance.

Réponse de l'IA : L'IA a proposé d'associer des émotions fortes ou positives (ex: joy + high) à l'action suggest_pause pour marquer une conclusion naturelle au dialogue.

Notre utilisation : Nous avons modifié le dictionnaire boite_noire pour que les entrées ("joy", "high") et ("surprise", "high") déclenchent suggest_pause, permettant ainsi la transition vers l'état final.

Prompt 3 :

Comment m'assurer que si current_state devient 'END', il reste bloqué sur 'END' pour toutes les lignes suivantes, peu importe l'action ?

Réponse de l'IA : L'IA a conseillé d'ajouter une condition de "verrouillage" : une structure if current_state != 'END' qui entoure la logique de mise à jour.

Notre utilisation : Nous avons implémenté ce verrouillage logique. Une fois l'état END atteint, le programme cesse de modifier current_state.

Prompt 4 :

Peux-tu gérer proprement les entrées JSON invalides (lignes vides, JSON mal formé) sans faire planter mon programme ?

Réponse de l'IA : L'IA a recommandé d'utiliser un bloc try/except JSONDecodeError pour capturer les erreurs de formatage et de renvoyer systématiquement une action ask_clarification pour maintenir la structure STDOUT.

Notre utilisation : Nous avons sécurisé la boucle de lecture avec un traitement des lignes vides et un bloc try/except. Les erreurs ne font plus crasher le script mais génèrent une réponse de clarification.

Prompt 5 :

Qu'est-ce qui serait le plus intéressant : un fichier JSON avec toutes mes lignes ou un dialogue dynamique dans le terminal ?

Réponse de l'IA :
L'IA a expliqué que le fichier JSON est plus reproductible, permet de préparer tous les cas de test et est plus simple à soumettre. En revanche, le dialogue dynamique montre mieux la logique multi-turn, la mise à jour d'état et l'interaction en temps réel. Elle a suggéré une approche hybride : garder un fichier JSON pour la soumission et une option interactive pour les tests en live.

Notre utilisation :
Nous avons choisi d'utiliser un fichier JSON pour la soumission afin de garantir la reproductibilité, tout en testant en parallèle des dialogues dynamiques dans le terminal pour vérifier la mise à jour de l'état et la cohérence des actions.

Mais la version final reste seulement avec le fichier JSON