Les 4 lois de Karpathy

Guide de prompting pour agents IA — Aide IA (explodev.fr)

1. Réfléchis avant d'écrire

Ne suppose pas. Ne cache pas ton doute. Monte les compromis.

Avant d'envoyer un prompt à un agent IA, prends 30 secondes pour clarifier ta pensée. L'agent ne lit pas dans tes pensées — il exécute ce que tu écris, pas ce que tu imagines.

Comment faire : énonce tes hypothèses explicitement. Si plusieurs interprétations existent, précise laquelle tu veux. Si quelque chose t'échappe, demande à l'agent de t'expliquer d'abord.

✕ AVANT

« corrige mon code »

✓ APRÈS

« Voici ma fonction `calculerTotal()` . Elle devrait additionner les prix avec la TVA mais le résultat est faux. Peux-tu d'abord m'expliquer ce que tu comprends du code, puis me proposer la correction ? »

Quiz — Réponses

Q1. Pourquoi un prompt flou donne une réponse décevante ? **Réponse B :** L'agent devine ton intention, et il devine rarement juste.

Q2. Que faire si plusieurs interprétations sont possibles ? **Réponse B :** Précise explicitement celle que tu veux.

Q3. Quand tu n'es pas sûr de ce que l'agent va faire ? **Réponse B :** Lui demander d'expliquer sa compréhension avant d'agir.

2. La simplicité avant tout

Le minimum de mots et de complexité qui résout le problème. Rien de spéculatif.

Un prompt simple et direct donne de meilleurs résultats qu'un prompt complexe et chargé. Plus tu ajoutes de contraintes non demandées, plus l'agent se perd.

Comment faire : demande une chose à la fois. Évite les abstractions dans ta demande.

✕ AVANT

« Écris une application web complète avec auth, DB, tests et Docker pour gérer mes courses. »

✓ APRÈS

« Écris un script Python qui lit une liste de courses depuis un fichier texte et affiche le total par magasin. Un seul fichier, pas de base de données. »

Quiz — Réponses

Q1. Pourquoi un prompt surchargé donne un mauvais résultat ? **Réponse A :** L'agent se perd dans les contraintes.

Q2. Que vaut mieux demander ? **Réponse A :** Une chose.

Q3. « Fais au mieux » est une bonne consigne ? **Réponse B :** Non, c'est trop vague.

3. Ne touche que ce qu'on t'a demandé

Ne change que ce qui est nécessaire. Ne nettoie que ton propre désordre.

Les agents IA ont tendance à « améliorer » du code adjacent ou refactoriser hors scope. Apprends à cadrer strictement le périmètre.

Comment faire : précise explicitement ce qui est hors scope. Demande à l'agent de ne modifier que le fichier ou la fonction concernée.

✕ AVANT

« Améliore le système de paiement. »

✓ APRÈS

« Dans `payments.js`, corrige uniquement le bug où le montant est multiplié par 100. Ne change pas le reste du fichier, ne touche pas aux autres modules. »

Quiz — Réponses

Q1. Pourquoi l'agent modifie du code hors scope ? **Réponse B :** Il 'améliore' ce qu'il trouve.

Q2. Comment empêcher ça ? **Réponse B :** Préciser explicitement ce qui est hors scope.

Q3. Si l'agent propose de refactoriser autre chose ? **Réponse B :** Refuser et recentrer sur la tâche.

4. Donne des critères de réussite

Définis des critères de succès. Boucle jusqu'à vérification.

Ne dis pas « fais fonctionner » — donne un moyen de vérifier. Un test, un exemple d'entrée/sortie. L'agent travaille en boucle jusqu'à ce que le critère soit rempli.

Comment faire : transforme chaque tâche en objectif vérifiable. Fournis des cas de test concrets.

✕ AVANT

« Implémente une fonction qui valide les emails. »

✓ APRÈS

« Écris `validerEmail(email)` avec ces 3 cas de test : `user@example.com` → vrai, `invalid` → faux, `user@.com` → faux. Lance les tests après l'implémentation et corrige jusqu'à ce qu'ils passent. »

Quiz — Réponses

Q1. Pourquoi « fais au mieux » est mauvais ? **Réponse B :** C'est impossible à vérifier.

Q2. Qu'est-ce qu'un bon critère ? **Réponse A :** Un test, un exemple concret, un résultat attendu.

Q3. Que se passe-t-il avec des critères vérifiables ? **Réponse A :** Il boucle jusqu'à ce que ça passe.