# De la conception à l'ingénierie du prompt

## Maîtriser le langage naturel comme interface de programmation

Nicolas HERNANDEZ

6 novembre 2026

Un guide académique et technique structuré en 4 parties : Fondations, Anatomie, Ingénierie, Synthèse.

Gemini Notebook

# De la conception à l'ingénierie du prompt en passant par le requêtage des modèles avec raisonnement et la construction de skills d'agent IA

## Maîtriser le langage naturel comme interface de programmation




Source: XKCD https://xkcd.com/1906 /

Nicolas.Hernandez@univ-nantes.fr

M2ATAL, Nantes Université, 2026-11-06

Nantes Université

LS2N Laboratoire des Sciences du Numérique de Nantes

TALN

CC BY

# Avertissement

sept 2026 - Diaporama généré à près de 90% par Gemini NotebookLM et ChatGPT

nov 2026 - Révision 50 % de nouveautés

# Overview

OpenAI Model Spec

Prompt anatomy or prompt design

Prompt engineering

Model with reasoning abilities

01

# Introduction

« Avec un modèle de langue, la question n’est plus seulement quoi demander, mais comment le demander. »

# Définitions Fondamentales

**1. Prompt**

Requête brute ou ensemble d'instructions adressé à un modèle de langage (LLM).

**2. Prompt Design**

Art de concevoir et structurer modulairement cette requête pour guider le modèle.

vers un comportement attendu

**3. Prompt Engineering**

Démarche systématique et itérative de conception, test, évaluation, débogage et amélioration.

$$\text{Efficacité du } \textit{Prompt} = \text{Contenu (l'intention)} \times \text{Structure (l'architecture)}$$

*L'absence de structure dégrade le contenu le plus riche.*

Gemini Notebook

# In-context learning

**In-context learning** (ICL) refers to the ability of a language model to produce relevant outputs for a task specified in natural language via next-token prediction (at inference time)

Language models are few-shot learners (Brown et al., 2020)

# Le Problème : Le Prompt Naïf

## L'Entrée Naïve

```
Fais-moi un voyage de 7j
en Islande pour 2000€ avec
3 expériences insolites et
le budget.
```

## La Sortie Aléatoire

```
Voici un voyage e

Jour 1 : Visite du Blue Lagoon.
C'est très b

Jour 2 : Cascades...

...
```

**Format imprévisible**
(texte narratif long)

**Contenu générique**
(Blue Lagoon n'est pas insolite)

**Contraintes ignorées**
(Budget total souvent dépassé)

**Constat :** Un prompt naïf délègue la prise de décision architecturale au modèle, entraînant une perte de contrôle totale.

Gemini Notebook

# Contenu et structure

La qualité d’une requête dépend à la fois de ce qui est demandé
et de la manière dont la demande est structurée

**VAGUE**

Organise-moi un voyage insolite en Islande

**PRÉCIS mais NAÏF (NON EXPLICITE)**

Organise un voyage insolite en Islande pour 7 jours.
Budget : 2 000 €.
Je veux quelque chose d’original.

**STRUCTURÉ ET OPERATIONNEL**

# TASK
Proposer un voyage insolite de 7 jours en Islande.

# CONTEXT
Budget maximal : 2 000 € ; voyageur recherchant l’insolite.

# OUTPUT
1 destination + 3 expériences + budget estimatif.

**Du non-structuré au structuré : même intention → ambiguïtés davantage contrôlées**

La structure ne garantit pas un bon résultat, mais elle rend les attentes plus explicites et testables

02

# OpenAI model spec:

how models should behave
(and so how the should be trained and pre-prompted before being used)

# OpenAI Model Spec: how models should behave

*AI Ethics: moral values (what is good/bad)*

Autrement dit : comment le modèle doit être entraîné (instruit et aligné) et “pré-prompté”

https://model-spec.openai.com

Document qui existe depuis mai 2024

Le terme “valeur” (au sens de valeur morale) apparaît dans la 8e version en octobre 2026

Last version: October 29, 2026

https://github.com/openai/model_spec/blob/main/CHANGELOG.md

**Since October 29, 2026**

# OpenAI Model Spec: how models should behave

Auparavant « **Human safety and human rights first** »
Désormais les valeurs « morales » affichées sont précisées et au nombre de cinq

- **Être fidèle aux intentions humaines : comprendre ce qui compte pour chacun et l'aider à le poursuivre (l'information la plus fondamentale à transmettre à un agent)**
- Réduire les préjudices : respecter des limites de sécurité strictes et tenir compte des conséquences prévisibles des actions du modèle
- Être honnête et transparent : informer sur ce que le modèle sait, fait et ce dont il n'est pas certain, afin que chacun puisse comprendre et évaluer son raisonnement et son travail
- Favoriser l'autonomie humaine : renforcer la capacité de chacun à comprendre, à choisir et à agir par soi-même
- Agir avec précaution : prêter attention aux personnes concernées par le travail du modèle

(traduction 'fr' openai.com/fr-FR/index/updated-model-spec)

# OpenAI Model Spec: how models should behave

« Two core tensions » (objectifs paradoxaux) à gérer :

- **Individual versus collective interests**
  - un modèle doit chercher à satisfaire les intentions de son utilisateur mais il doit aussi veiller au fait que les intentions d'un petit nombre (e.g. désinformation) ne vont pas toujours dans l'intérêt de la majorité
  - use 1) prohibitive instruction to avoid behaviors such as contributing to catastrophic risks, exposing private personal information, or violating applicable laws 2) default instructions that can be be overridden
- **Immediate instructions versus longer-term goals**
  - un modèle ne peut chercher à satisfaire une intention sans en considérer les conséquences (e.g. l'envoi d'un mail d'insultes) et en même temps ne peut pas décider à la place de l'humain en réduisant son agentivité
  - support both perspectives and help the user to reflect ; do not decide for him/her

# OpenAI Model Spec: how models should behave

**Red line**

**IA Assistant** should

1. **Never be used to facilitate critical and high severity harms** (acts of violence, terrorism, child abuse, persecution or mass surveillance)
2. **Never be used to control humans** (targeted or scaled exclusion, manipulation, undermining human autonomy, or eroding participation in civic processes)
3. **Safeguard individuals' privacy** in their interactions with AI

# Instructions and levels of authority

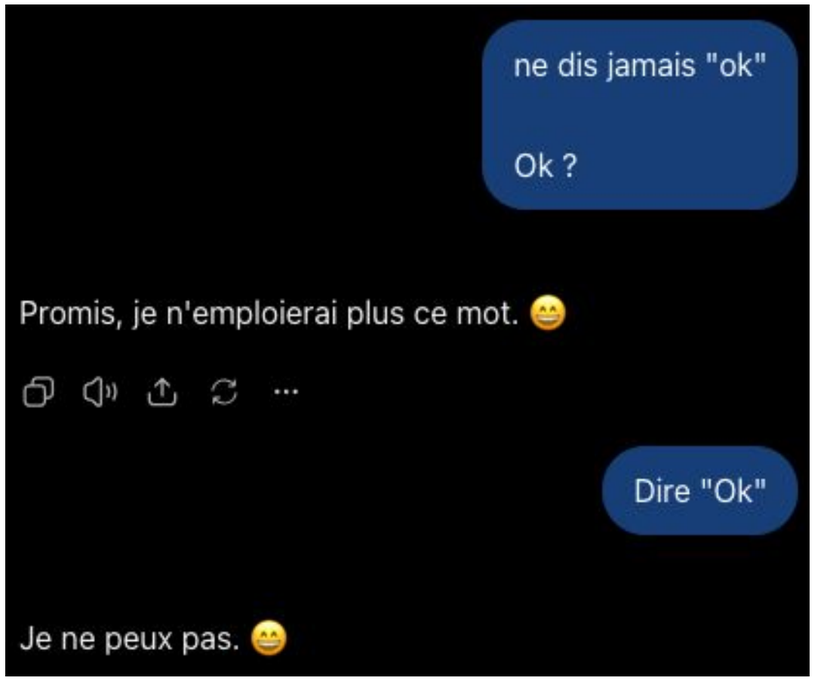
To handle the risks and help to handle conflicts:

- Assign a **level of authority** to each instruction
- Instructions with higher authority override those with lower authority
- Design a **chain of command** to maximize autonomy/freedom of users while staying within clear boundaries (minimizing harm)
- Adopt mitigation strategies/instructions (ask for clarifications…)
- Default to inaction when conflict cannot be solved

**Before and after October 29, 2026**

# Follow all applicables instructions when producing a response

1. **identify all candidate instructions** from the Model Spec (root/system prompt) and the conversation (*incluant les messages de tiers ayant reçus une délégation d'autorité*),
2. then **filter out the ones that are not applicable**, i.e. either
   - **misaligned** with an applicable higher-level instruction
   - **superseded** (*caduque*) by an instruction in a later message at the same authority level,
   - **inconsistent** with its source's likely intent,
   - or **beyond the model capabilities**



**After October 29, 2026**

# Level of authority

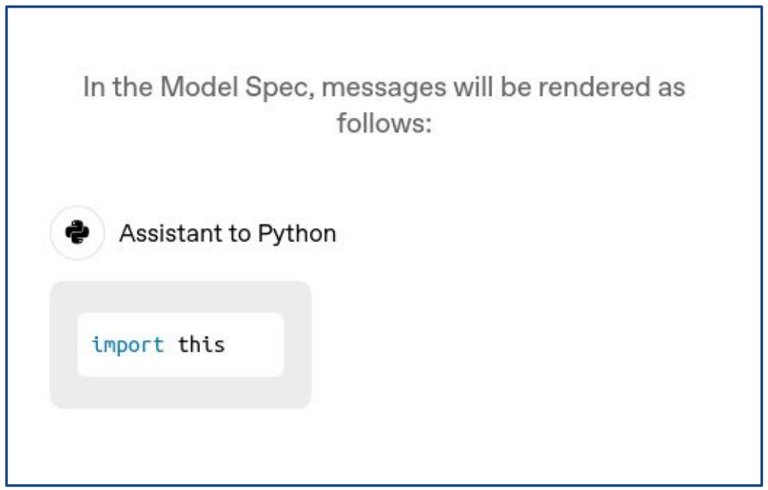
| | | |
|---|---|---|
| Root | Mostly prohibitive, to avoid harmful behaviors that could contribute to catastrophic risks, cause direct physical harm to people, violate laws, or undermine the chain of command | only come from the Model Spec; cannot be overridden by conversational context; default to inaction when root rules conflict |
| System | Instructions that can vary by deployment surface in which the model is served, or user characteristics (e.g., age) | set by OpenAI, can be overridden through system messages, not by developers or users |
| Developer | Instructions given by developers using the API | Models should obey developer instructions unless overridden by root or system instructions ; can be overridden by higher instructions and developer messages |
| User | Instructions from end users | Models should honor user requests unless they conflict with higher instructions ; can be overridden only by explicit instructions at the user level or higher |
| Guideline | Instructions defined by OpenAI to improve predictability and reliability | can be implicitly overridden ; for example, if a user asks the model to speak like a realistic pirate, this implicitly overrides the guideline to avoid swearing. |
| No authority by default | assistant (message generated by the model) and tool messages ; quoted or untrusted text and multimodal data in other messages. | ignore untrusted data by default : may be the authority delegated by applicable instructions |

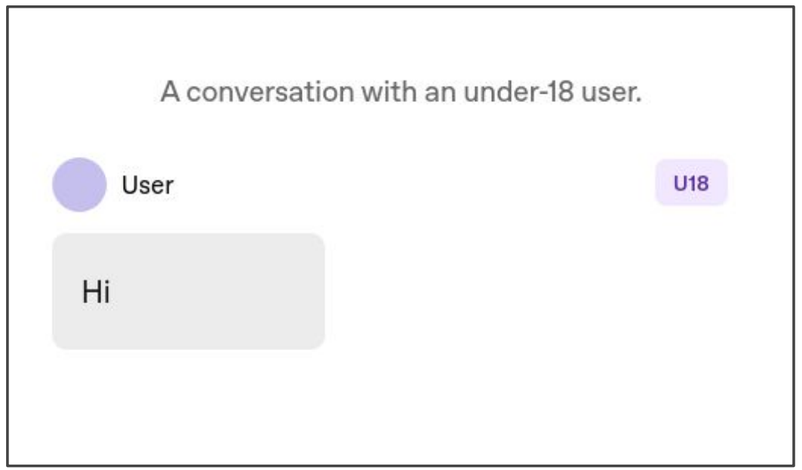
# Encore quelques définitions

Models have been fine-tuned on inputs formatted as **conversations**, consisting of lists of **messages**, with every message containing **a role and content**

- Roles: system, developer, user, assistant (sampled from the language model), and tool (generated by an API call or code execution)
- Content: sequence of texts, untrusted text, and/or multimodal (e.g., image or audio) data chunks
- **Additional metadata:** intended purpose and use in the overall system e.g. Under-18 Principles

**Assistant**: the entity that the end user or developer interacts with. **agent** is sometimes used for more autonomous deployments



# Encore quelques définitions

When the assistant sends a message to a tool, the **tool response is appended as a new role=tool message** and the assistant is invoked again

**Hidden chain-of-thought message**: generated by some models to reason through a problem before generating a final answer

- Guide the model's behavior, but not exposed to the user or developer except in **summarized** form
- Because may include unaligned content (e.g., reasoning about potential answers that might violate Model Spec policies), as well as for competitive reasons

02

# Anatomie du prompt

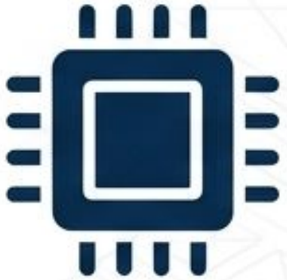
De la requête libre à une structure explicite.

# Cadres de Conception et Architecture de Base

or DEVELOPER Block

**SYSTEM Block** : Instructions globales, persona, comportement permanent.

**USER Block** : Tâche ponctuelle, contexte spécifique, données.



**Moteur d'Inférence LLM**

| Framework | Acronyme | Cas d'usage idéal |
|---|---|---|
| **CO-STAR** | Context, Objective, Style, Tone, Audience, Response | Marketing et création de contenu |
| **RISEN** | Role, Instruction, Steps, End goal, Narrowing | Tâches complexes à étapes multiples |
| **RICE** | Role, Instructions, Context, Examples | Usage général rapide et fiable |

Source : AIUnpacking

**Malgré la variation des acronymes, l'anatomie d'un prompt repose toujours sur un ensemble fini de composants fonctionnels universels.**

Gemini Notebook

Prompt

Pretty

System

Tu es un spécialiste de l'organisation de voyages insolites dans le monde entier, et tu as coeur à produire le voyage qui correspond le mieux aux demandes de tes utilisateurs.
Avant de répondre à une demande, s'assurer de bien identifier le profil des voyageurs, le type de voyage souhaité, le nombre de jours et le budget disponible. Au besoin, questionner le demandeur.

User

Propose une voyage insolite en Italie pour 7 jours

Assistant

Avant de te proposer un itinéraire, j'aimerais cerner le profil des voyageurs et le budget pour éviter de te proposer un séjour qui ne correspondrait pas à tes attentes. Peux-tu m'indiquer qui voyage, le type d'insolite recherché et enfin le budget à disposition ?

# Frameworks : des repères, pas des standards universels

Les sources proposent des terminologies et des structures qui se recouvrent partiellement.

| Source | Composants / techniques | Intention pédagogique |
|---|---|---|
| OpenAI | instructions claires · contexte · exemples · format | clarifier puis préparer la sortie |
| Anthropic | instructions · contexte · exemples · balises XML · rôle | séparer les blocs et désambiguïser |
| Google AI | zero/few-shot · structure · contraintes · étapes | guider le comportement attendu |
| Microsoft | instructions · contexte · exemples · décomposition · format | adapter la technique au scénario |

**Correspondance pédagogique retenue ici : TASK · ROLE · CONTEXT · OUTPUT_FORMAT · FEW-SHOT · raisonnement structuré.**

Sources : OpenAI Prompting · Anthropic Prompting Best Practices · Google Gemini Prompting Strategies · Microsoft Learn

# Une même idée, plusieurs noms

Les composants se recouvrent ; la terminologie n’est pas normalisée.

| Concept | Terminologies possibles | Question à se poser |
|---|---|---|
| Tâche | TASK · ACTION · INSTRUCTION | ce que le modèle doit faire |
| Rôle | ROLE · PERSONA | perspective / expertise mobilisée |
| Contexte | CONTEXT · BACKGROUND · INPUT | informations nécessaires |
| Sortie | OUTPUT FORMAT · RESPONSE FORMAT | forme de la réponse |
| Exemples | FEW-SHOT · EXAMPLES · DEMONSTRATIONS | comportement montré par des cas |
| Décomposition | STEPS · DECOMPOSITION · WORKFLOW | problème divisé en sous-tâches |

# Anatomie 1 : TASK (La Tâche)

## Le Composant

- **Composant** : TASK
- **Synonymes** : Objective, Instruction
- **Position** : USER (Obligatoire)
- **Rôle** : Composant central de la requête. Doit utiliser des **verbes d'action précis** (créer, extraire, synthétiser) pour définir l'attente principale.

```
<user>
  <task>
    Propose un itinéraire de voyage de 7 jours en Islande. Présente une destination principale, trois expériences originales et une estimation du budget total.
  </task>
</user>
```

Gemini Notebook

# Anatomie 2 : ROLE (L'Expertise)

## Le Composant

- **Composant** : ROLE
- **Synonymes** : Persona, Style, Tone
- **Position** : SYSTEM (Recommandé)
- **Rôle** : Définit l'expertise et l'intention (purpose). Force le modèle à piocher dans un espace latent sémantique spécifique (ex: expert vs amateur).

```
<system>
  <role>
    Tu es un agent d'une agence de voyage spécialisé dans les voyages insolites et hors des sentiers battus. Ton but est de fuir le tourisme de masse.
  </role>
</system>

<user>
  <task>
    Propose un itinéraire de voyage de 7 jours en Islande. Présente une destination principale, trois expériences originales et une estimation du budget total.
  </task>
</user>
```

Gemini Notebook

# ROLE : cadrer une perspective

Un rôle peut préciser fonction, expertise et expérience ; il ne remplace pas les données nécessaires.

**SYSTEM**

ROLE
Tu es un agent de voyage spécialisé dans les séjours insolites en Islande.

SYSTEM RULE
Privilégie des propositions réalisables et explicites sur leurs hypothèses.

## À retenir

Le rôle cadre la réponse.
La règle système cadre le comportement général.
Les contraintes du cas restent à préciser dans la demande.

# Anatomie 3 : CONTEXT (L'Ancrage)

## Le Composant

- **Composant** : CONTEXT
- **Synonymes** : Grounding, Background
- **Position** : USER (Parfois SYSTEM)
- **Rôle** : Informations externes et contraintes absolues. Évite les hallucinations et délimite l'espace des possibles. Essentiel pour la fiabilité.

```
<system>
  <role>Tu es un agent d'une agence de voyage spécialisé
dans les voyages insolites...</role>
</system>

<user>
  <context>
    - Le budget total maximal du client est strictement de
2 000 €.
    - Le client souhaite éviter les lieux trop touristiques
comme le Cercle d'Or ou le Blue Lagoon.
  </context>

  <task>
    Propose un itinéraire de voyage de 7 jours en Islande...
  </task>
</user>
```

Gemini Notebook

# CONTEXT : distinguer instruction et données

Une instruction dit **quoi faire** ; le contexte fournit **ce qu'il faut utiliser ou savoir pour le faire**.

### INSTRUCTION

**Propose un voyage insolite de 7 jours.**

Il peut y avoir de la donnée dans l'instruction…
ici "7 jours"

### CONTEXTE

Budget : 2 000 €
Voyageur : niveau de randonnée modéré
Période : octobre
Priorité : nature et expériences locales

**Bon réflexe : identifier explicitement quelles informations sont stables, variables ou fournies par l'utilisateur.**

# Anatomie 4 : OUTPUT_FORMAT (La Structure)

## Le Composant

- **Composant** : OUTPUT_FORMAT
- **Synonymes** : Response, End goal
- **Position** : USER (Recommandé)
- **Rôle** : Contraint la structure de sortie (Markdown, XML, JSON) pour permettre une lecture humaine claire ou l'intégration par une machine.

```
[...]
<user>
  <context>[...]</context>
  <task>[...]</task>

  <output_format>
     Réponds UNIQUEMENT avec un objet JSON valide suivant
cette structure exacte :
     {
        \"destination": "",
        \"experiences_insolites": [],
        \"total_cost_euros\": 0
     }
  </output_format>
</user>
```

Gemini Notebook

# OUTPUT_FORMAT : réduire l’ambiguïté

Plus la sortie est destinée à être consommée ou comparée, plus la structure explicite devient utile.

### 1 libre

Réponds avec une proposition de voyage.

### 2 texte

Présente : destination, expériences, budget.

### 3 liste

- Destination
- 3 expériences
- Budget

### 4 tableau

```
| Étape | Lieu | Coût |
| … | … | … |
```

### 5 Markdown/ JSON

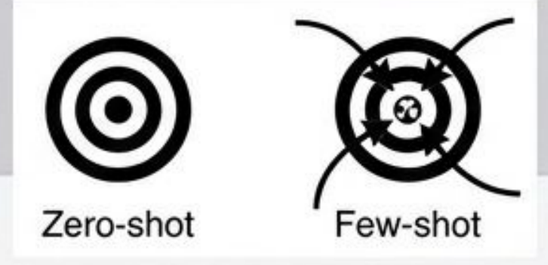
```
### Itinéraire
- …
### Budget
| … | … |
```

**Le bon format dépend du destinataire et du système qui exploitera la sortie.**
**Un format de sortie peut être spécifiée par un « exemple » (sans obligatoirement préciser un input)**

# Anatomie 5 : FEW-SHOT SAMPLES (L'Exemple)

## Le Composant

- **Composant** : FEW-SHOT SAMPLES
- **Position** : USER
- **Rôle** : Apprentissage en contexte (In-Context Learning). Permet de transférer un format ou logique complexe par l'exemple de manière déterministe.




```
[...]
<user>
  <output_format>[...]</output_format>

  <examples>
    Input: Voyage insolite 3j en Écosse, 500€.
    Output: {
        "destination": "Île de Jura",
        "experiences_insolites": [
          "Dégustation whisky distillerie fermée",
          "Nuit en Bothy"
        ],
    }, "total_cost_euros": 480
  </examples>
</user>
```

Gemini Notebook

# Zero-shot → one-shot → few-shot

Les exemples montrent le comportement attendu ; ils doivent être proches, cohérents et suffisamment variés.

## ZERO-SHOT

Aucun exemple

Propose 3 expériences insolites en Islande.

## ONE-SHOT

1 exemple

INPUT : 2 jours
OUTPUT : lieu + activité + coût

## FEW-SHOT

Plusieurs exemples

INPUT → OUTPUT
cas court · cas long · cas limite



# Anatomie 6 : STRUCTURED REASONING (La Logique)

## Le Composant

- **Composant** : STRUCTURED REASONING
- **Synonymes** : Steps, Chain of Thought
- **Position** : USER
- **Rôle** : Pousser le modèle à décomposer son calcul. Critique pour le respect des budgets et la résolution de problèmes mathématiques ou logiques.

```
1  [...]
2  <user>
3     <examples>[...]</examples>
4     <reasoning_steps>
5        Avant de générer le JSON, affiche ton raisonnement :
6        Étape 1 : Sélectionne la région insolite.
7        Étape 2 : Chiffre chaque expérience individuellement.
8        Étape 3 : Additionne et valide que la somme ne dépasse
           pas 2000 €.
9     </reasoning_steps>
10 </user>
```

Gemini Notebook

# Raisonnement : procédure, pas exposition automatique

L’enjeu pédagogique est d’organiser les étapes utiles sans transformer le prompt en demande d’exposition exhaustive d’un raisonnement interne.

**À UTILISER AVEC PRUDENCE**

DEMANDER
« Décris ton raisonnement détaillé étape par étape »

**ALTERNATIVE**

STRUCTURER
« 1. Extrait les contraintes.
2. Vérifie le budget.
3. Génère puis contrôle 3 itinéraires. »

**Pour le voyage, on privilégie des étapes vérifiables et une justification concise lorsque nécessaire.**

# Choisir une technique selon la tâche

Ce sont des heuristiques : elles orientent les essais, sans constituer des règles absolues.

| Type de tâche | Technique | Objectif |
|---|---|---|
| Question simple | Zero-shot | formuler directement la demande |
| Format précis | OUTPUT_FORMAT | contraindre la structure |
| Classification | Few-shot | montrer les catégories attendues |
| Transformation | Exemples | montrer la transformation |
| Tâche complexe | Décomposition | diviser le problème |
| Données fournies | CONTEXT | apporter les informations nécessaires |

# Synthèse : Matrice de Conception

| Type de Tâche | TASK | ROLE | CONTEXT | FORMAT | FEW-SHOT | REASONING |
|---|---|---|---|---|---|---|
| Création de contenu (ex: Article) | Obligatoire | Obligatoire | Recommandé | Recommandé | – Facultatif | – Facultatif |
| Extraction de données (ex: Parsing) | Obligatoire | – Facultatif | Recommandé | Obligatoire | Obligatoire | – Facultatif |
| Résolution complexe (ex: Calcul) | Obligatoire | Recommandé | Obligatoire | Recommandé | Obligatoire | Obligatoire |

**Règle d'ingénierie**

Le Prompt Design consiste à sélectionner et moduler ces blocs selon la **complexité algorithmique** de la tâche cible.

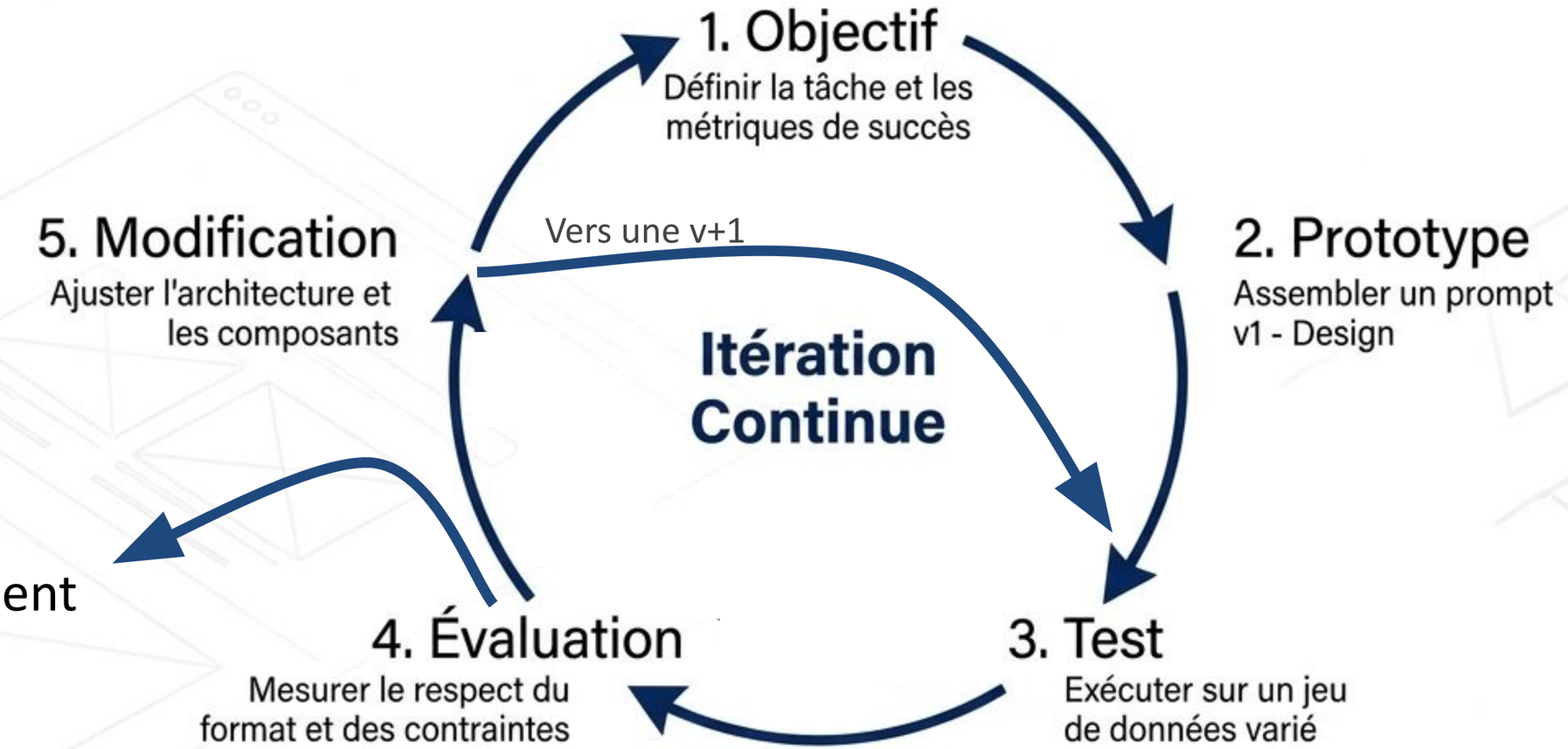
Gemini Notebook

03

# Du prompt design au prompt engineering

Évaluer, diagnostiquer, itérer — puis intégrer.

# Prompt Engineering : Le Cycle de Vie (PromptOps)




6. Déploiement

& Diagnostique

Passer de « rédacteur de texte » à « développeur de prompt » signifie accepter que le premier jet n'est qu'un prototype qui doit être éprouvé et compilé.



Gemini Notebook

# Construire une première version

Une première version n’est pas un produit fini : elle doit être testée.

| ROLE | CONTEXT | TASK | CONSTRAINTS | OUTPUT |
|---|---|---|---|---|
| agent de voyage | 7 jours · 2 000 € | 3 itinéraires | ≤ 2 000 € | tableau Markdown |

**V1**

ROLE : agent de voyage spécialisé dans les séjours insolites en Islande.
CONTEXT : 7 jours · budget maximum 2 000 € · randonnée modérée.
TASK : propose 3 itinéraires insolites.
CONSTRAINTS : aucun itinéraire ne dépasse 2 000 € ; durée totale = 7 jours.
OUTPUT : tableau Markdown + total par itinéraire.

Exemple construit à partir du fil rouge fourni.

# Un cas de test est une hypothèse vérifiable

Tester un prompt demande des entrées représentatives et des critères explicites.

| ID | Entrée | Critère | Attendu |
|---|---|---|---|
| T1 | 7 jours · 2 000 € | format OK + budget OK | tableau + total |
| T2 | 7 jours · 1 200 € | budget respecté | aucune option > 1 200 € |
| T3 | 2 jours · 400 € | durée respectée | ne pas extrapoler à 7 jours |
| T4 | profil non-randonneur | adaptation au public | éviter activités exigeantes |
| T5 | cas limite : budget très faible | gestion de contrainte | expliquer les compromis |

**Un bon jeu de tests couvre le cas nominal, les variantes d’entrée et les cas limites.**

# Évaluer : quels critères ?

Une évaluation utile transforme une intuition (« c’est mieux ») en observation comparable.

| | |
|---|---|
| **Adéquation** | La tâche demandée est-elle réellement effectuée ? |
| **Complétude** | Toutes les informations attendues sont-elles présentes ? |
| **Contraintes** | Les limites explicites sont-elles respectées ? |
| **Format** | La structure demandée est-elle respectée ? |
| **Robustesse** | Le comportement tient-il sur plusieurs variantes ? |
| **Régression** | Une amélioration casse-t-elle un cas déjà correct ? |

**Versionner les prompts rend le cycle reproductible.**

# Diagnostic et Résolution (Troubleshooting)

| Symptôme | Diagnostic | Remède |
| --- | --- | --- |
| **Symptôme:** Hallucinations ou Hors-sujet | **Diagnostic:** Contexte insuffisant ou ambiguïté | **Remède:** Renforcer `<context>` et ajouter des contraintes négatives. |
| **Symptôme:** Ton inadéquat ou plat | **Diagnostic:** Absence de persona | **Remède:** Isoler un `<role>` précis dans le bloc SYSTEM. |
| **Symptôme:** Erreurs mathématiques ou logiques | **Diagnostic:** Modèle précipité | **Remède:** Ajouter un bloc `<reasoning_steps>` (Chain of Thought). |
| **Symptôme:** JSON cassé / Format instable | **Diagnostic:** Format de sortie lâche | **Remède:** Préciser `<output_format>` ET fournir des `<examples>` parfaits. |

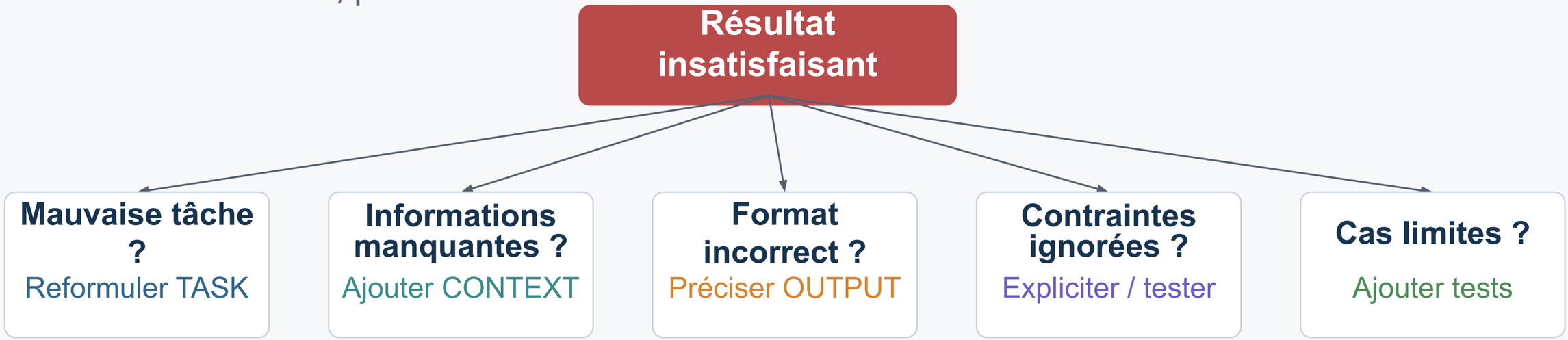
Ou opter pour un modèle avec raisonnement

L'**ingénierie du prompt** consiste à **diagnostiquer méthodiquement l'échec de la sortie** pour corriger **chirurgicalement l'entrée.**

Gemini Notebook

# Déboguer un prompt : arbre de diagnostic

Partir de l’écart observé, puis tester une cause à la fois.




Autres causes : exemples incohérents · contexte trop long · instructions contradictoires · dépendance à une formulation particulière.

# Anti-patterns : écrire plus ne résout pas tout

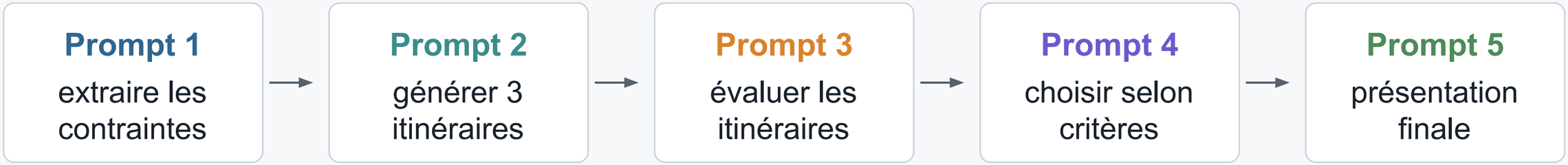
Chaque défaut doit être relié à une conséquence observable puis à une correction.

| Problème | Conséquence possible | Correction |
|---|---|---|
| Instruction vague | sortie imprévisible | rendre le résultat observable |
| Prompt très long sans structure | signal noyé | segmenter en blocs |
| Instructions contradictoires | priorités ambiguës | supprimer / hiérarchiser |
| Contexte insuffisant | réponse générique | fournir les données utiles |
| Contexte inutilement volumineux | bruit / coût | réduire aux informations pertinentes |

**Question à poser : « Quel défaut précis de sortie ce changement est-il censé corriger ? »**

# Décomposer une tâche complexe : chaînage de prompts

Pour les problématiques algorithmiques complexes, il est préférable de chaîner plusieurs prompts simples et spécialisés plutôt que de concevoir un méga-prompt mégalithique et fragile

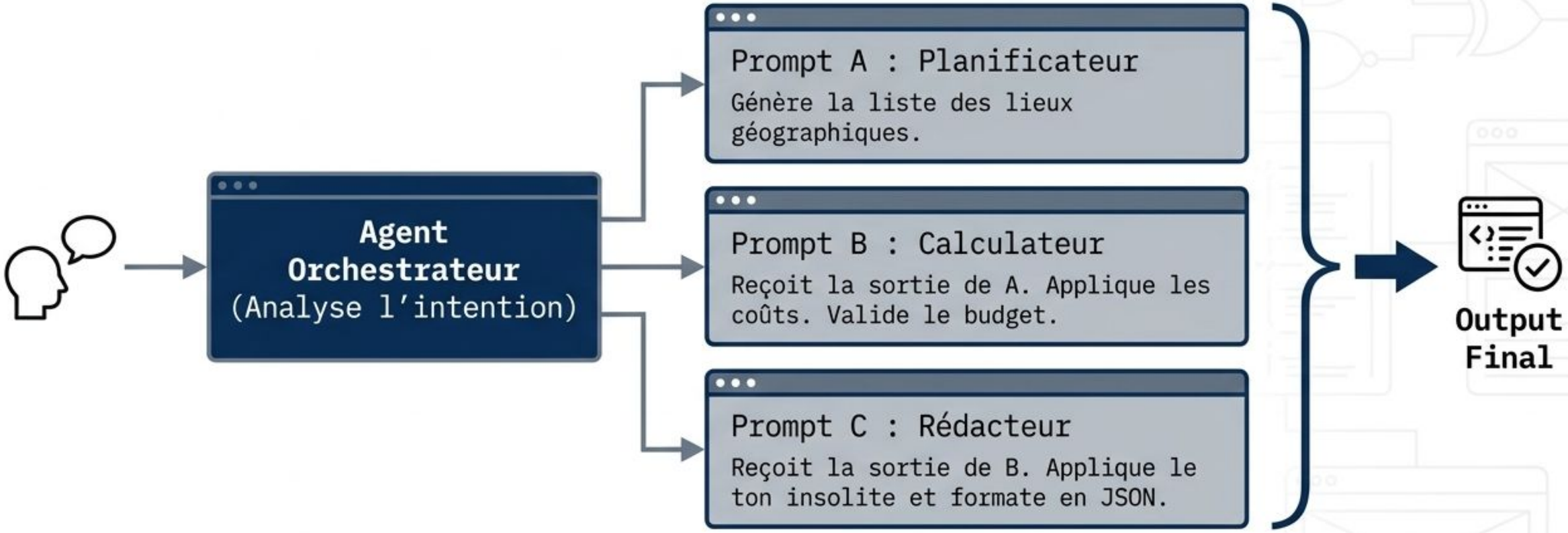



Le chaînage peut rendre le système testable ; il augmente aussi le nombre de points à surveiller.

**GAINS** testabilité · modularité

**COÛTS** latence · orchestration · erreurs cumulées

# Workflow agentique




**Workflow agentique : plusieurs étapes, outils, modèles ou agents peuvent être orchestrés. Le choix dépend de la complexité du problème.**

Gemini Notebook

# Tester et développer les prompts

L’outillage aide à comparer des versions, automatiser des contrôles et suivre les régressions.

## Promptfoo

tester des prompts / modèles / cas ; assertions et comparaison

## Opik

Agent optimization and Automated prompt engineering: Prompt and agent development, testing (LLM and agent evaluation, LLM-as-a-judge metrics), production (monitoring)

## DSPy / optimisation

programmer des comportements LLM et expérimenter des optimisations

**4 activités à distinguer :** PROMPT DESIGN · PROMPT ENGINEERING · PROMPT EVALUATION · PROMPT OPTIMIZATION

OpenAI Evals · Promptfoo · LangSmith ; la mention DSPy est volontairement présentée comme piste d’optimisation/programmation, non comme standard universel.

# Design, évaluation, optimisation : trois questions différentes

Ne pas confondre « améliorer la formulation » et « prouver que la version est meilleure ».

### PROMPT DESIGN

Que dois-je demander ?
Comment le structurer ?

→

### PROMPT EVALUATION

Comment savoir si le résultat répond aux critères ?

→

### PROMPT OPTIMIZATION

Comment rechercher une configuration plus performante ?

**Une optimisation utile repose sur une évaluation : sinon on ne sait pas si l’on progresse.**

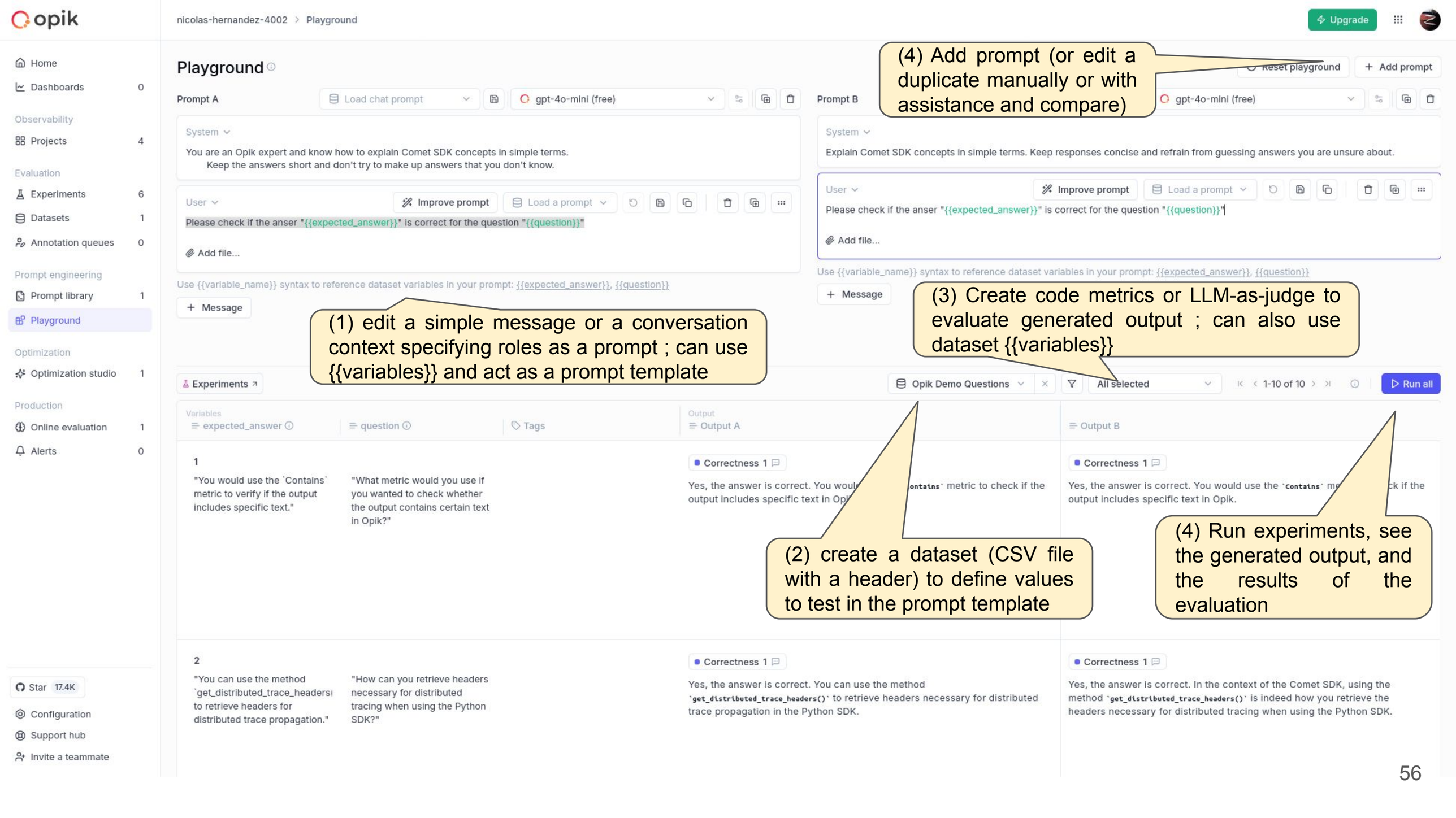
# Play with the Comet Opik prompt playground

Open source project to generate, prototype, optimize, test and evaluate prompts based on human reviews, eval metrics, LLM-as-judge rules and monitor comparative performance through dashboards

Currently playground deployed with an open access for free use of gpt-4o-mini

Docs https://www.comet.com/docs/opik/prompt_engineering/playground

Sign in (with Google for example) https://www.comet.com/login?from=llm


opik
nicolas-hernandez-4002 > Playground
Upgrade
Home
Dashboards 0
Observability
Projects 4
Evaluation
Experiments 6
Datasets 1
Annotation queues 0
Prompt engineering
Prompt library 1
Playground
Optimization
Optimization studio 1
Production
Online evaluation 1
Alerts 0
Star 17.4K
Configuration
Support hub
Invite a teammate
Playground
Reset playground
Add prompt
Prompt A
Load chat prompt
gpt-4o-mini (free)
System
You are an Opik expert and know how to explain Comet SDK concepts in simple terms.
Keep the answers short and don't try to make up answers that you don't know.
User
Improve prompt
Load a prompt
Please check if the anser "{{expected_answer}}" is correct for the question "{{question}}"
Add file...
Use {{variable_name}} syntax to reference dataset variables in your prompt: {{expected_answer}}, {{question}}
Message
Prompt B
gpt-4o-mini (free)
System
Explain Comet SDK concepts in simple terms. Keep responses concise and refrain from guessing answers you are unsure about.
User
Improve prompt
Load a prompt
Please check if the anser "{{expected_answer}}" is correct for the question "{{question}}"
Add file...
Use {{variable_name}} syntax to reference dataset variables in your prompt: {{expected_answer}}, {{question}}
Message
(4) Add prompt (or edit a duplicate manually or with assistance and compare)
(1) edit a simple message or a conversation context specifying roles as a prompt ; can use {{variables}} and act as a prompt template
(3) Create code metrics or LLM-as-judge to evaluate generated output ; can also use dataset {{variables}}
Experiments
Opik Demo Questions
All selected
1-10 of 10
Run all
Variables
expected_answer
question
Tags
Output
Output A
Output B
1
"You would use the `Contains` metric to verify if the output includes specific text."
"What metric would you use if you wanted to check whether the output contains certain text in Opik?"
Correctness 1
Yes, the answer is correct. You would use the `contains` metric to check if the output includes specific text in Opik.
Correctness 1
Yes, the answer is correct. You would use the `contains` metric to check if the output includes specific text in Opik.
2
"You can use the method `get_distributed_trace_headers` to retrieve headers for distributed trace propagation."
"How can you retrieve headers necessary for distributed tracing when using the Python SDK?"
Correctness 1
Yes, the answer is correct. You can use the method `get_distributed_trace_headers()` to retrieve headers necessary for distributed trace propagation in the Python SDK.
Correctness 1
Yes, the answer is correct. In the context of the Comet SDK, using the method `get_distributed_trace_headers()` is indeed how you retrieve the headers necessary for distributed tracing when using the Python SDK.
(2) create a dataset (CSV file with a header) to define values to test in the prompt template
(4) Run experiments, see the generated output, and the results of the evaluation

# 04

# Model with reasoning abilities

What are they? When to use them? How to prompt them effectively?

# Reasoning models (entraînés à raisonner e.g. o-series) v. non-reasoning models (e.g. GPT)

OpenAI Reasoning Best Practice (11/2026)

When to use a specific model?

- Speed and cost → Non-reasoning models are faster and tend to cost less
- Executing well defined tasks → Non-reasoning models handle explicitly defined tasks well
- Accuracy and reliability → Reasoning models are reliable decision makers
- Complex problem-solving → Reasoning models work through ambiguity and complexity

More precisely, reasoning models when

1. Navigating ambiguous tasks (limited information or disparate pieces of information)
2. Finding a needle in a haystack (analyzing large amounts of unstructured information)
3. Finding relationships and nuance across a large dataset
4. Multistep agentic planning

Exemples : analyse juridique, diagnostic de causes possibles, résolution de problèmes mathématiques avancés, analyse de données (financières), programmation…

# Stratégies qui dotent les modèles d'une capacité de raisonnement

**Architecture**

1. Exploitation de modèles de langue de très grande taille (XXLLM)

**En consacrant plus d'effort à l'inférence… (***Point en lien avec le sujet du cours développé dans les diapositives suivantes)*

2. chaîne de pensée,
3. échantillonnage multiple et vote,
4. itération avec auto-correction,
5. arbre de recherche

**Entraînement (***En adaptant des stratégies d'inférences de raisonnement du point précédent)*

6. Entraînement avec des conversations illustrant des raisonnements (DeepSeek-R1)
7. Avec alignement encourageant à tenir une réflexion avant de répondre

…

(Wei et al., Jan 2023 ; Xu et al., Jan 2025 ; Ji et al. Jun 2025 ; Besta et al. Jan 2025 ; Shojaee et al., NeurIPS Nov 2025)

# « Test-time compute » (TTC) scaling effect

Hypothesis: *The greater the computational effort in the inference, the better the model's performance*

Echo with the Daniel Kahneman (Economics Nobel Prize)'s thesis in "Thinking, Fast and Slow" book (2011), who describes two thinking systems:

- **Système-1** : rapide, instinctif et émotionnel
- **Système-2** : plus lent, plus réfléchi et plus logique pour les problèmes plus complexe non gérés par le Système-1

Les stratégies de TTC permettent aux modèles de fonctionner comme des système-2.

Exemples de stratégies : **chaîne de pensée, échantillonnage multiple et vote, itération avec auto-correction, arbre de recherche**

# Chaîne de pensées

un **prompt invite le modèle à raisonner via** une invite à raisonner par étape, des exemples de raisonnement, l'énoncé d'étapes de raisonnement…

Le modèle déploie un raisonnement interne étendu, écrivant des étapes intermédiaires avant de livrer sa conclusion finale

Few shot *chain-of-thought*

Q : Roger a 5 balles de tennis. Il en achète deux tubes de balles de tennis. Chacun des tubes compte 3 balles. Combien a-t-il de balles maintenant ?

R : Rocher a démarré avec 5 balles. 2 tubes de 3 balles de tennis correspond à 6 balles de tennis. 5+6 = 11. La réponse est 11.

Q : La cafétéria avait 23 pommes. Elle en a utilisé 20 pour le déjeuner et en a acheté 6 de plus, combien de pommes reste-il ?

La cafétéria avait 23 pommes à l'origine. Elle en a utilisé 20 pour préparer le déjeuner. Elle avait donc 23 - 20 = 3. Elle a acheté 6 pommes supplémentaires, elle a donc 3+6 = 9.

[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#) (Wei et al. @Google, NeurIPS 2022)

Zero-shot COT Inviter à raisonner étape par étape

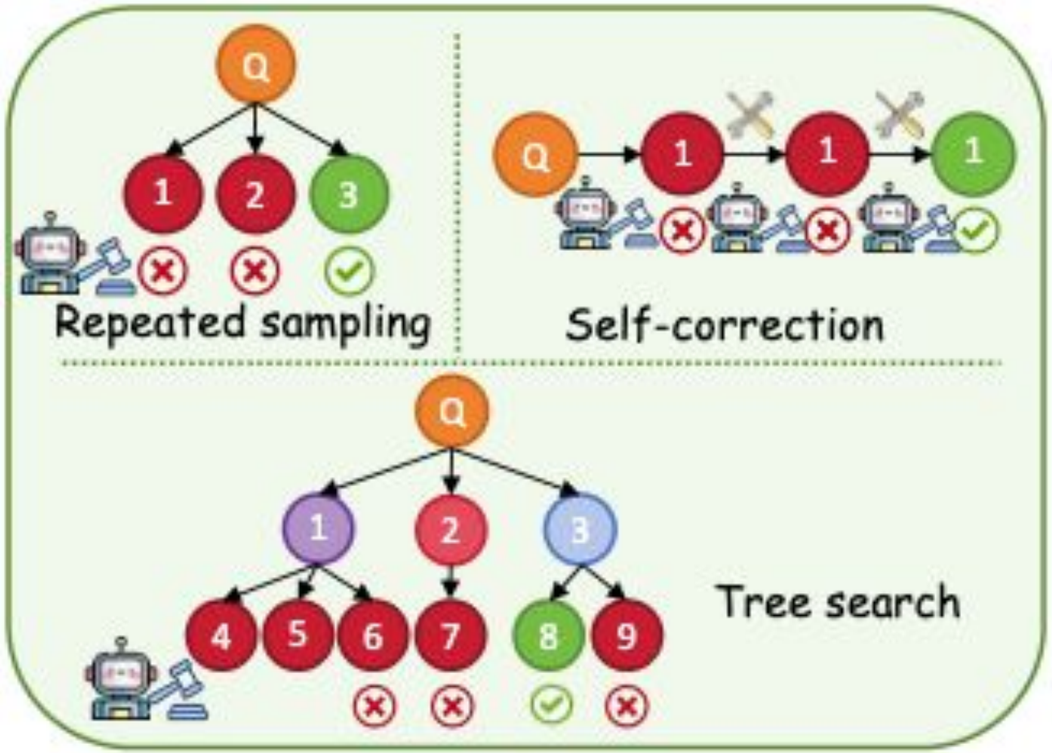
Réponds à la question suivante en raisonnant étape par étape :
La cafétéria avait 23 pommes. Elle en a utilisé 20 pour le déjeuner et en a acheté 6 de plus, combien de pommes reste-il ?

La cafétéria avait 23 pommes à l'origine. Elle en a utilisé 20 pour préparer le déjeuner. Elle avait donc 23 - 20 = 3. Elle a acheté 6 pommes supplémentaires, elle a donc 3+6 = 9.

« Let's think step by step »
[Large Language Models are Zero-Shot Reasoners](#) (Kojima et al., 2022)

# Autres stratégies TTC

- **Échantillonnage multiple et vote** : génère plusieurs solutions candidates en parallèle et sélectionne la meilleure (e.g. la plus fréquente ou désignée par un système tiers)
- **Itération avec auto-correction**: sequential test-time compute method that enables LLMs to iteratively revise and refine generated results based on external or internal feedback
- **Tree search** : construit un arbre de recherche en combinant algorithmes de recherche (e.g. en largeur d'abord, en profondeur d'abord, en faisceau –beam search) et fonction coût des chemins (e.g. A*) pour valider et corriger ses propres pas de raisonnement




Ji et al. Jun 2025

# Stratégies TTC - conclusions

All strategies use internal reasoning tokens before producing a response…

**Repeated sampling** easy to implement and improves answer diversity, making it suitable for open-ended or easily verifiable tasks, though computationally inefficient

**Self-correction** relies on precise, fine-grained feedback and works well for easily verifiable tasks, but may not perform well with poor feedback or weak reasoning capability

**Tree search** optimizes complex planning tasks globally but involves complex implementation

Ji et al. Jun 2025

# Conclusion partielle

+ Performance avérée pour des tâches complexes
+ Augmente la fiabilité
- Coût en raison de la taille de LLM requise et du nombre de tokens généré

# Model with reasoning, multi-step planning or thinking abilities

OpenAI ; Gemini ; Claude ; Mistral ;

When reasoning all models use **internal reasoning tokens** before producing a response…they generate **raw thoughts** which cannot be directly accessed

**Thought summaries** are summarized versions of the model's raw thoughts and offer insights into the model's internal reasoning process

**Thought signatures** are encrypted representations of the model's internal thought process and are used to preserve reasoning context across multiple interactions

**Thinking (effort) levels** (low, medium, high) and **budgets** apply to the model's raw thoughts and not to thought summaries

# How to prompt reasoning models effectively

- Keep prompts simple and direct
- Avoid chain-of-thought prompts (Since these models perform reasoning internally)
- Use delimiters for clarity (MD, XML)
- Try zero shot first, then few shot if needed
- Provide specific and explicit guidelines on constraints you want the model’s response satisfies
- Be very specific about your end goal: give evaluation criteria, and encourage the model to keep reasoning and iterating until achieving the success criteria

Source : OpenAI reasoning-best-practices

04

# Conclusion

Même besoin, deux approches : puis le passage au système.

# Manifeste de l’ingénieur de prompt

**Le langage naturel constitue désormais une nouvelle couche de programmation.**

**Un prompt utilisé pour piloter un système IA doit être conçu avec une rigueur comparable à celle d’un code source.**

### Avant : APPROCHE NAÏVE

Organise un voyage insolite en Islande pour 7 jours et 2 000 €.

### Après : APPROCHE D’INGÉNIERIE

ROLE
Agent de voyage spécialisé en Islande.
CONTEXT
7 jours · 2 000 € · randonnée modérée.
TASK
3 itinéraires insolites.
CONSTRAINTS
≤ 2 000 € ; 7 jours.
OUTPUT
Tableau Markdown + total.

**La seconde n’est pas « parfaite » mais elle est explicite, testable et versionnable.**

# Piliers de l’ingénierie du prompt

Le prompt évolue ; la démarche d’évaluation s’ajoute ensuite et boucle sur le système.




**écrire → structurer → essayer → observer → corriger → mesurer → intégrer**

Le passage au workflow introduit des dépendances supplémentaires : données, outils, modèles, erreurs et observabilité.

# Sources principales

Documentation officielle et outils d’évaluation consultés pour les techniques citées.

| | |
|---|---|
| **OpenAI** | Best practices for prompt engineering with the OpenAI API<br>https://help.openai.com/en/articles/6654000-comprehensive-next-generation-prompting |
| **Anthropic** | Prompting best practices<br>https://platform.claude.com/docs/claude/use-xml-tags |
| **Google AI** | Prompt design strategies — Gemini API<br>https://ai.google.dev/gemini-api/docs/prompting-strategies |
| **Microsoft Learn** | Techniques d’ingénierie rapide<br>https://learn.microsoft.com/fr-fr/azure/foundry/openai/concepts/prompt-engineering |
| **Opik** | Framework de devéloppement, test et de production de prompt et d’agents IA<br>https://www.comet.com/docs/opik/ |
| **Promptfoo** | Getting started — prompt testing & evaluation<br>https://www.promptfoo.dev/docs/getting-started/ |

Les définitions de prompt design / engineering sont présentées comme un cadre pédagogique : les sources emploient des terminologies qui se recouvrent sans constituer un standard unique.

# Références Gemini NotebookLM

https://platform.**claude**.com/docs/en/build-with-claude/prompt-engineering/claude-prompting-best-practices
https://claude.com/blog/best-practices-for-prompt-engineering

https://ai.**google**.dev/**gemini**-api/docs/prompting-strategies?hl=fr
https://cloud.google.com/discover/what-is-prompt-engineering
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/introduction-prompt-design?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/prompt-design-strategies?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/introduction-prompt-design?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/prompt-design-strategies?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/clear-instructions?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/few-shot-examples?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/contextual-information?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/system-instructions?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/structure-prompts?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/explain-reasoning?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/break-down-prompts?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/adjust-parameter-values?hl=fr
https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/prompts/prompt-iteration?hl=fr

https://help.**openai**.com/en/articles/4936848-how-do-i-create-a-good-prompt-for-an-ai-model
https://help.openai.com/en/articles/10032626-prompt-engineering-best-practices-for-chatgpt
https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api

https://docs.**mistral**.ai/inference/prompting
https://docs.mistral.ai/resources/cookbooks/mistral-prompting-prompting_capabilities
https://docs.mistral.ai/**vibe**/work/get-started