PYTHON & DATA : API, INTELLIGENCE ARTIFICIELLE

Prompt Engineering API LLM

Apprenez à interroger et intégrer des LLMs dans vos projets via les APIs OpenAI et Anthropic.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
3j (21h)	~~900.00 €~~ **540.00 €**	**Intermédiaire**	**Français**	**1-6**	**Formation**

1 PUBLIC VISÉ

- Développeurs web (frontend, backend ou fullstack) souhaitant intégrer des LLMs dans leurs applications métier ou leurs outils internes.
- Développeurs Python ayant déjà consommé des APIs REST et souhaitant passer à l'intégration concrète d'OpenAI ou Anthropic.
- Profils techniques ayant déjà expérimenté ChatGPT ou Claude en mode "chat" et qui veulent aller plus loin via l'API.
- Freelances et développeurs en agence cherchant à proposer des fonctionnalités IA à leurs clients.

2 PRÉREQUIS

- Bonne maîtrise de Python (fonctions, dictionnaires, appels HTTP avec requests ou httpx) ou de JavaScript/Node.js (fetch, async/await).
- Avoir déjà consommé une API REST avec authentification par clé (expérience avec n'importe quelle API tierce suffisante).
- Notions de base sur les formats JSON.
- Aucune connaissance en machine learning ou en mathématiques requise.

3 OBJECTIFS PÉDAGOGIQUES

- Comprendre le fonctionnement d'un LLM : tokens, contexte, température, probabilités
- Maîtriser les techniques de prompting : zero-shot, few-shot, chain-of-thought, rôles système
- Consommer les APIs OpenAI et Anthropic en Python et JavaScript : authentification, paramètres, gestion des erreurs
- Structurer les sorties d'un LLM : JSON, XML, extraction fiable de données
- Implémenter des patterns avancés : multi-turn conversation, RAG simplifié, tool use / function calling
- Intégrer un LLM dans une application web réelle (API FastAPI ou endpoint Next.js)
- Maîtriser les coûts, la latence et les bonnes pratiques de mise en production

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) Comprendre les LLMs et maîtriser les fondamentaux du prompting

Module 1 Comment fonctionne un LLM (2h)

Ce que le développeur doit savoir (sans les maths) (1h)

- Qu'est-ce qu'un LLM : prédiction du token suivant, pas de "compréhension"
- La notion de contexte (context window) : taille, implications, ce que le modèle "voit"
- Les tokens : ce que c'est, comment les compter, pourquoi ça compte pour la facturation
- Température et top-p : contrôler la créativité vs la déterminisme
- Différences pratiques entre les modèles : GPT-4o, Claude Sonnet, Claude Haiku, GPT-4o-mini
- Les hallucinations : pourquoi elles se produisent, comment les limiter
- Cas pratique : observer l'effet de la température sur des sorties répétées

L'écosystème des APIs LLM (1h)

- OpenAI : modèles disponibles, endpoint /v1/chat/completions, tarification par token
- Anthropic : modèles Claude, endpoint /v1/messages, différences structurelles avec OpenAI
- Créer ses clés API, configurer les variables d'environnement, bonnes pratiques de sécurité
- Les SDKs officiels : openai (Python/JS) et anthropic (Python/JS)
- Cas pratique : premier appel API en Python vers OpenAI et Anthropic, comparaison des structures de réponse

Module 2 Techniques de prompting fondamentales (3h)

Les rôles et la structure d'un prompt (1h)

- Le trio system / user / assistant : rôles et responsabilités
- Écrire un bon system prompt : persona, contraintes, format de sortie, ton
- L'ordre des instructions : ce qui est lu en premier vs en dernier
- Délimiteurs et structure : balises XML, markdown, numérotation
- Cas pratique : même tâche, trois system prompts différents

Zero-shot, few-shot et chain-of-thought (1h)

- Zero-shot : formuler une instruction claire sans exemple
- Few-shot prompting : guider le modèle avec des exemples entrée/sortie
- Construire des exemples pertinents : diversité, cas limites, format cohérent
- Chain-of-thought (CoT) : demander au modèle de "réfléchir à voix haute" avant de répondre
- Zero-shot CoT : l'instruction magique et pourquoi elle fonctionne
- Cas pratique : extraction d'entités avec et sans few-shot

Prompts pour tâches spécifiques (1h)

- Classification de texte : sentiment, intention, catégorie
- Résumé : longueur contrainte, format bullet, résumé exécutif
- Transformation de données : convertir du texte en JSON structuré
- Génération de code : demander du code commenté, testé, dans un style donné
- Correction et révision : relecture, reformulation, traduction
- Cas pratique : pipeline de traitement de tickets support

Module 3 Structurer les sorties (2h)

Obtenir du JSON fiable (1h)

- Pourquoi les LLMs produisent parfois du JSON invalide
- Instructions de format dans le prompt : schéma explicite, exemples, contraintes
- Le mode JSON d'OpenAI (response_format: { type: "json_object" })
- Les tool_use / function_calling pour forcer un schéma de sortie
- Valider et parser la réponse : json.loads, gestion des erreurs, retry automatique
- Cas pratique : extraire des données structurées depuis des emails non formatés

Outputs XML et texte balisé (1h)

- Quand préférer XML au JSON : lisibilité, parsing côté LLM
- Demander des sorties avec balises personnalisées : <summary>, <action>, <confidence>
- Parser les balises XML en Python avec re ou xml.etree
- Cas pratique : générateur de fiches produit avec champs balisés (titre, description, tags, prix suggéré)

Jour 2 (7h) Patterns avancés et intégration dans une application

Module 4 Conversations multi-tours et gestion du contexte (2h)

Gérer l'historique de conversation (1h)

- Comment construire et maintenir un historique messages[]
- Limites de la context window : que faire quand l'historique devient trop long
- Stratégies de troncature : supprimer les anciens tours, résumer le passé
- Memory patterns : résumé glissant, mémoire externe (simple dict ou Redis)
- Cas pratique : chatbot de support technique avec mémoire de session

Injection dynamique de contexte (1h)

- Enrichir le system prompt à la volée : données utilisateur, données métier, date/heure
- Templating de prompts : f-strings, Jinja2, LangChain PromptTemplate
- Séparer les prompts du code : fichiers .txt, variables d'environnement
- Versionner ses prompts comme du code
- Cas pratique : assistant personnalisé qui adapte son ton selon le profil utilisateur injecté

Module 5 Function calling et Tool use (2h)

Le principe du function calling (1h)

- Qu'est-ce que le function calling : le LLM décide d'appeler une fonction, pas de l'exécuter
- Déclarer des outils : nom, description, paramètres JSON Schema
- Intercepter l'appel, exécuter la vraie fonction, renvoyer le résultat au modèle
- Différences OpenAI (tools) vs Anthropic (tools avec tool_use)
- Cas pratique : LLM capable d'appeler une API météo ou une base de données fictive

Patterns multi-outils et agents simples (1h)

- Chaîner plusieurs appels d'outils dans une même conversation
- Boucle agent : while tool_calls: execute → feed result → continue
- Limiter les risques : timeout, nombre max d'itérations, validation des paramètres
- Cas pratique : agent de recherche qui interroge une API puis formate un rapport

Module 6 RAG simplifié : augmenter le LLM avec ses propres données (2h)

Le principe du Retrieval-Augmented Generation (1h)

- Pourquoi un LLM ne "connaît" pas vos données métier
- L'idée du RAG : récupérer les passages pertinents, les injecter dans le prompt
- Embeddings : qu'est-ce que c'est, comment les générer (OpenAI text-embedding-3-small)
- Similarité cosinus : trouver les documents les plus proches d'une question
- Stocker des embeddings : approche simple avec numpy, introduction à ChromaDB

Implémenter un RAG minimal (1h)

- Pipeline : indexation → query → retrieval → prompt augmenté → réponse
- Chunking de documents : taille des chunks, chevauchement, stratégies
- Construire un prompt RAG : contexte injecté + question utilisateur
- Limites et pièges : chunk trop long, mauvaise récupération, hallucination malgré le contexte
- Cas pratique : mini-FAQ intelligente basée sur un fichier Markdown ou une documentation

Module 7 Atelier d'intégration (1h)

- Combiner les techniques vues : system prompt structuré + few-shot + function calling + historique
- Identifier les bons patterns selon le cas d'usage (chatbot, pipeline batch, agent)
- Exercice guidé : concevoir le prompt et l'architecture d'un cas d'usage apporté par les participants

Jour 3 (7h) Intégration production, coûts et projet de synthèse

Module 8 Intégrer un LLM dans une application web (3h)

Backend : endpoint LLM avec FastAPI ou Express (1h30)

- Créer un endpoint /chat qui proxifie les appels à l'API LLM
- Streaming des réponses : stream=True côté OpenAI, stream=True côté Anthropic, Server-Sent Events côté client
- Implémenter le streaming en FastAPI avec StreamingResponse et EventSourceResponse
- Gérer les erreurs API : rate limit (429), timeout, quota dépassé
- Cas pratique : endpoint FastAPI streamant les réponses d'un assistant intégré

Frontend : afficher les réponses en temps réel (1h)

- Consommer un stream SSE en JavaScript avec EventSource ou fetch + ReadableStream
- Afficher les tokens au fur et à mesure dans le DOM
- Indicateur de chargement, bouton d'annulation, gestion des erreurs réseau
- Cas pratique : interface chat minimaliste (HTML/JS ou composant React) connectée au backend

Sécurité et architecture (30min)

- Ne jamais exposer une clé API côté client : architectures correctes
- Rate limiting applicatif : limiter les appels par utilisateur
- Modération des entrées : l'API de modération OpenAI, filtres côté prompt
- Logging des appels : tracer les prompts, les réponses et les coûts pour audit

Module 9 Coûts, performance et mise en production (2h)

Maîtriser les coûts (1h)

- Comprendre la facturation : input tokens vs output tokens, prix par modèle

Calculer le coût d'un cas d'usage avant de développer : outil de simulation

- Réduire les coûts sans sacrifier la qualité : choisir le bon modèle selon la tâche
- Prompt compression : supprimer le verbiage, condensifier les exemples few-shot
- Caching de prompts : OpenAI Prompt Caching, Anthropic Prompt Caching
- Cas pratique : audit de coût d'un pipeline existant et propositions d'optimisation

Performance et fiabilité (1h)

- Latence : p50 vs p95, ce qui l'impacte (longueur du prompt, longueur de la réponse, modèle)
- Traitement batch : envoyer plusieurs requêtes en parallèle avec asyncio.gather
- L'API Batch d'OpenAI : traitement asynchrone à moitié prix pour les workloads non temps-réel
- Évaluation de la qualité des prompts : construire un jeu de tests, automatiser les évals
- Monitoring en production : logger latence, coût, taux d'erreur, satisfaction utilisateur
- Cas pratique : refactoring d'un pipeline séquentiel en pipeline asynchrone

Module 10 Projet de synthèse (2h)

Réalisation d'une mini-application LLM de bout en bout

- **Cahier des charges** : construire une fonctionnalité IA concrète au choix parmi une liste proposée (assistant de rédaction, analyseur de contrat, générateur de tests unitaires, FAQ intelligente, résumeur de tickets...)
- **Fonctionnalités attendues** : system prompt structuré et versionné, au moins une technique avancée (function calling, RAG, streaming ou multi-turn), endpoint backend exposant la fonctionnalité, estimation du coût par appel
- Étape 1 : conception du prompt et choix de l'architecture
- Étape 2 : implémentation du backend (FastAPI ou Express)
- Étape 3 : intégration frontend (interface minimaliste)
- Étape 4 : présentation et revue collective
- Retour formateur individualisé sur les choix de prompting et d'architecture

5 COMPÉTENCES VISÉES

- Écrire des prompts efficaces et robustes adaptés à des tâches métier réelles
- Consommer les APIs OpenAI et Anthropic en Python et JavaScript avec gestion des erreurs
- Structurer les sorties LLM en JSON ou XML exploitable dans une application
- Implémenter les patterns avancés : function calling, RAG, streaming, multi-turn
- Intégrer un LLM dans un backend FastAPI ou Express avec streaming SSE
- Estimer, surveiller et optimiser les coûts d'un usage LLM en production

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (live coding avec les APIs OpenAI et Anthropic sur des cas d'usage réels), méthode interrogative (analyse des sorties du modèle et discussion des choix de prompting) et méthode active (exercices d'écriture de prompts, itérations et projet de synthèse fil rouge). L'accent est mis sur la pratique immédiate : chaque technique est testée et confrontée à ses limites dans la même session.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Notebooks Jupyter et scripts Python annotés avec exercices et corrections par module
- Environnement de développement : VS Code + Python 3.12+ + Node.js 20+ + comptes OpenAI et Anthropic (crédits de test fournis)
- Pour le distanciel : visioconférence (Zoom ou équivalent), partage d'écran, chat en direct
- Accès à la plateforme pédagogique LaPolaris (supports, ressources, émargement)

8 MODALITÉS D'ÉVALUATION

- En cours de formation : exercices de prompting corrigés et comparés à chaque module
- En fin de formation : réalisation d'une mini-application LLM de bout en bout
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Prompts structurés, clairs et adaptés à la tâche : system prompt, rôles, format de sortie
- Sorties JSON ou balisées valides, parsées et gérées sans erreur dans le code
- Intégration API fonctionnelle avec gestion des erreurs et retry
- Architecture backend/frontend correcte sans exposition de la clé API côté client
- Estimation de coût réaliste et pertinente pour le cas d'usage implémenté

10 MODALITÉS DE VALIDATION

Attestation de fin de formation délivrée à l'issue du parcours, conditionnée à une assiduité d'au moins 80 % et à la réalisation du projet de synthèse. L'attestation précise les objectifs atteints et les compétences acquises.

11 SUIVI ET ACCOMPAGNEMENT

- Feuilles d'émargement signées par demi-journée (présentiel) ou émargement numérique (distanciel)
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via la vérification du dépôt GitHub de chaque apprenant en fin de parcours

12 CONDITIONS D'ACCÈS

Formation accessible sur inscription directe, sans prérequis administratif particulier. Le financement peut être pris en charge par l'employeur dans le cadre d'un plan de développement des compétences, ou en autofinancement avec possibilité de paiement en plusieurs fois.

13 DÉLAIS D'ACCÈS

Inscription possible jusqu'à **5 jours ouvrés** avant le début de la session. Pour toute demande urgente, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations sont accessibles aux personnes en situation de handicap. Pour toute situation nécessitant un aménagement (matériel, temporel ou pédagogique), nous vous invitons à nous contacter avant l'inscription afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr