Blue AcademIA
Blue AcademIA est une marque de BlueMarketing
contact@blueacademia.fr
www.blueacademia.fr

Programme de formation : Créer un agent IA en Python

Vous développez, mais vos projets d'IA générative restent au stade de la démo. Les POC s'empilent, rien ne passe en production. Ce parcours de 10 jours vous apprend à concevoir un agent IA, à le coder en Python, à le connecter à vos données avec un pipeline RAG, puis à le tester, le déployer, le monitorer et à l'optimiser pour qu'il reste rentable en production (prompt caching, routing de modèles, RAG avancé).
Vous terminez par un panorama outillé des frameworks agentiques (LangGraph, Claude Agent SDK, LlamaIndex) pour savoir quand
adopter un framework et lequel.
Chaque brique est introduite puis immédiatement mise en pratique : 60 à 80 % du temps se passe sur le clavier. L'approche est 100 % pratique : vous repartez avec votre propre agent IA codé, testé, conteneurisé, déployé dans le cloud, optimisé et construit sur vos propres cas d'usage.

Durée : 70.00 heures ou 10.00 jour(s) si formation en présentiel.

Lieu : À distance ou en présentiel

Profils des stagiaires

- Développeurs et développeuses souhaitant concevoir et déployer des agents IA
- Ingénieurs logiciels, data engineers, profils techniques impliqués dans un projet d'IA générative
- Profils techniques en reconversion vers l'IA appliquée

Prérequis

- Avoir des bases en Python (variables, fonctions, structures de contrôle)
- Avoir un environnement de développement déjà installé (Python, un éditeur de code, Git).
- Aucune connaissance préalable en IA ou en machine learning n'est requise.
- Les pré-requis sont validés lors d'un échange avec le commanditaire du projet de formation et retranscrits dans le document de recueil des besoins. Il s'agit d'un document collaboratif partagé systématiquement avec le commanditaire à l'issue du premier entretien pour co-construire l'action de formation. Par ailleurs, nous adressons à chaque apprenant, 2 semaines avant le début de la formation, un questionnaire de préparation de la formation pour adapter au mieux la formation à ses besoins.

Objectifs pédagogiques

- Choisir le bon modèle de LLM selon le cas d'usage et rédiger des prompts fiables (RISE, Few-shot, Chain of Thought, auto-vérification anti-hallucination)
- Concevoir l'architecture de votre agent : cahier des charges, couches perception / raisonnement / action / mémoire, patterns réactif / ReAct / planificateur, sécurité by design
- Développer l'agent en Python : appels multi-provider aux APIs LLM (OpenAI, Anthropic, Mistral), tools, mémoire, pattern ReAct avec LangChain ou LlamaIndex

- Connecter l'agent à vos données et à votre système d'information : pipeline RAG complet, APIs, webhooks, outils métier (CRM, ticketing, messagerie)
- Tester et sécuriser l'agent : tests unitaires et d'intégration, LLM-as-Judge, gestion des erreurs
- Mesurer l'impact métier : KPIs, ROI, plan de déploiement et conduite du changement
- Optimiser les coûts et la latence en production : instrumentation (Langfuse / LangSmith), prompt caching, routing de modèles, RAG avancé (re-ranking, hybrid search)
- Choisir un framework agentique en connaissance de cause — LangGraph, Claude Agent SDK, LlamaIndex — et porter votre agent vers une architecture en graphe d'états
- Livrer votre agent IA en production, conteneurisé, déployé dans le cloud, monitoré et optimisé, applicable immédiatement

Contenu de la formation

- GenAI, LLM et prompt engineering avancé (1,5 jour)
    - La stack technologique : IA, machine learning, deep learning, IA générative ; fonctionnement des LLM (tokens, prédiction statistique, coût par token, fenêtre de contexte)
    - Hallucinations et biais : causes, exemples concrets et processus de vérification (auto-questionnement, vérification des sources)
    - Cartographie de l'écosystème : OpenAI, Anthropic, Mistral, Google, Hugging Face ; propriétaire vs open source, comparaison des modèles selon les cas d'usage
    - Prompt engineering du fondamental à l'avancé : framework RISE, Zero-shot, Few-shot, Chain of Thought, auto-vérification anti-hallucination
    - Paramétrage d'un LLM : température, top-p, max tokens, frequency penalty ; quand recourir au prompting, au RAG ou au fine-tuning (et ce qu'est le RLHF)
    - Sécurité et conformité : évaluation de la sensibilité des données, anonymisation, RGPD
    - Cas pratique appliqué : Comparatif de modèles sur trois cas d'usage métiers, optimisation des prompts et mesure de la qualité des réponses. QCM de validation du module.
- Conception fonctionnelle et architecture technique (1 jour)
    - Identifier et scorer les processus métiers automatisables : matrice valeur métier × faisabilité technique
    - Typologie des agents : réactif, avec raisonnement (pattern ReAct), planificateur — choisir le bon pattern
    - Le cahier des charges fonctionnel complet : problème métier, utilisateurs, cas d'usage, données, intégrations, KPIs
    - Schématiser l'architecture de l'agent : perception, raisonnement, action, mémoire ; concevoir les outils (tools) que l'agent peut appeler
    - Sécurité by design : validation des entrées, system prompt défensif, whitelist des tools, menaces spécifiques (prompt injection, jailbreak)
    - Données et parties prenantes : contraintes RGPD anticipées dès la conception, implication des acteurs métiers, techniques et juridiques
    - Cas pratique appliqué : Lancement du projet fil rouge : rédiger l'architecture complète de son agent — diagrammes, APIs nécessaires, workflows, KPIs. Livrables : cahier des charges et diagramme d'architecture. QCM de validation du module.
- Développement Python et prototypage de l'agent (2 jours)
    - Fondamentaux Python pour les agents (fonctions modulaires, gestion des exceptions, fichiers JSON / CSV / PDF, requêtes HTTP) et architecture de projet recommandée (main.py, llm.py, tools/, memory/, config.py)
    - Appels multi-provider aux APIs LLM (OpenAI, Anthropic, Mistral) : couche d'abstraction, system prompt, historique, sorties structurées en JSON exploitables par l'agent
    - Pattern ReAct (réfléchir → choisir → exécuter → observer) : implémentation minimale, puis orchestration avec LangChain ou LlamaIndex
    - Ajouter des tools : recherche web, lecture de PDF, requêtes SQL — des fonctions pures et testables

    - Gérer la mémoire : mémoire de session (historique, fenêtre de rappel) et mémoire persistante (base vectorielle, score de similarité)
    - Debug et observabilité : logs structurés, correction pas à pas des comportements indésirables ; accélérer le développement avec un assistant de code IA (GitHub Copilot, Cursor)
    - Cas pratique appliqué : Live coding avec le formateur puis lab fil rouge : développer le prototype complet de son agent, de la réception de la requête à la réponse ou l'action automatisée. Livrables : code Python fonctionnel, tools intégrés, mémoire, logs. QCM de validation du module.
- RAG et intégration dans le système d'information (2 jours)
    - Le pipeline RAG complet : ingestion, chunking, embeddings, stockage vectoriel (Chroma, FAISS, Qdrant), retrieval, augmentation du prompt
    - Nettoyer et structurer les données (PDF, CSV, JSON, SQL) pour les rendre exploitables par l'agent ; qualité du retrieval
    - RGPD et gouvernance du RAG : anonymisation des données sensibles, gestion des permissions
    - Intégration au SI : APIs REST, webhooks, connecteurs no-code (n8n, Make, Zapier) ; connexion à un CRM, un outil de ticketing (Jira, Zendesk), une messagerie (Slack, Teams) ou un ERP
    - Multimodalité : traitement de documents et d'images, transcription audio (Whisper)
    - Gérer les erreurs d'intégration : timeouts, données manquantes, formats inattendus
    - Cas pratique appliqué : Deux ateliers : construire un pipeline RAG sur un corpus documentaire métier, puis intégrer son agent fil rouge à un outil de l'entreprise (API authentifiée, workflow automatisé de bout en bout). QCM de validation du module.
- Tests, déploiement et mise en production + soutenance (2,5 jours)
    - Tester un agent aux sorties non déterministes : tests unitaires des tools et de la mémoire, tests d'intégration du pipeline
    - Évaluer la qualité des réponses : pertinence, fidélité, LLM-as-Judge, construction d'un corpus de test
    - Conteneuriser avec Docker : Dockerfile, build, run, logs — un déploiement portable et reproductible
    - Déployer dans le cloud : AWS Lambda, Azure Functions, GCP Cloud Run ou Scaleway ; checklist (clés d'API, authentification) et budget réel d'un agent
    - Monitorer en production : temps de réponse, taux d'erreur, coûts d'API, dashboards et alertes
    - KPIs métier (temps gagné, taux de résolution, coût par transaction), ROI, plan de déploiement et conduite du changement
    - Cas pratique appliqué : Écrire les tests (unitaires, intégration, LLM-as-Judge) de son agent fil rouge, le conteneuriser, le déployer sur une plateforme cloud et configurer le monitoring. Soutenance : démonstration live de l'agent, présentation de l'architecture, des KPIs et du plan de déploiement, évaluée sur une grille détaillée. QCM de validation du module.
- Optimisation et coaching (0,5 jour)
    - Mesurer avant d'optimiser : les trois métriques clés (TTFT, coût par requête, qualité), instrumentation avec Langfuse ou LangSmith, baseline chiffrée sur requêtes réelles
    - Prompt caching, le levier n° 1 : cache_control (Anthropic), auto-cache (OpenAI), structuration cache-friendly des prompts — jusqu'à -90 % de coût et -85 % de latence au premier token
    - Routing et cascade de modèles : classifier la difficulté des requêtes et router vers le bon modèle (petit modèle pour la classification, grand modèle pour le raisonnement)
    - RAG optimisé : re-ranking, hybrid search (BM25 + dense), contextual retrieval, query rewriting / HyDE
    - Cache sémantique : quand il est pertinent (FAQ, questions répétitives) — et quand il devient dangereux (réponses personnalisées, données temporelles)
    - Patterns de production : fallback multi-provider, rate limiting, retry avec backoff, budget alerts, circuit breaker
    - Cas pratique appliqué : Refactorer son agent fil rouge pour activer le prompt caching, mesurer coût et latence avant / après (tableau comparatif, traces d'observabilité) et dérouler la checklist prod-ready.
- Frameworks agentiques — LangGraph, Claude Agent SDK et au-delà (0,5 jour)
    - Panorama des frameworks agentiques : orchestration par graphes (LangGraph, Pydantic AI), SDK fournisseurs (Claude Agent SDK, OpenAI Agents SDK), multi-agent (CrewAI, AutoGen, Smolagents), RAG-centric (LlamaIndex)

- Critères de choix : complexité du workflow, multi-agent réel ou non, place du RAG, prototype vs production — et les cas où un script Python sans framework suffit
- LangGraph en pratique : state, nodes, edges conditionnels, cycles, checkpointing, human-in-the-loop
- Claude Agent SDK : sub-agents, skills, intégration MCP native
- LlamaIndex pour le RAG agentique
- Les anti-patterns classiques à éviter (empiler les frameworks, multi-agent injustifié, magie non debuggable)
- Cas pratique appliqué : Porter son agent fil rouge vers LangGraph (graphe d'états), comparer avec la version développée from scratch, puis construire sa grille de décision framework pour ses propres projets.

Organisation de la formation

Equipe pédagogique

Christophe Ancey est formateur praticien en IA générative, ingénieur (ESIGEC, 2000), et développe et exploite lui-même des agents IA en production, du prototype Python au monitoring. Ce parcours condense sa pratique de terrain : architecture d'agents, pipelines RAG, tests et déploiement, optimisation des coûts et frameworks agentiques, qu'il a transmis à des équipes de développeurs d'ESN.

Moyens pédagogiques et techniques

- Supports pédagogiques commentés et partagés à distance
- Apports méthodologiques structurés
- Templates prêts à l'emploi
- Cas pratiques appliqués à l'entreprise
- Etudes de cas concrets
- Quiz
- Mise à disposition en ligne de documents supports à la suite de la formation

Dispositif de suivi de l'exécution de l'évaluation des résultats de la formation

- Feuilles de présence
- QCM de niveau application à l'issue des principaux modules
- Exercices et travaux pratiques corrigés avec retour individualisé
- Soutenance du projet fil rouge (démonstration live, architecture, KPIs, plan de déploiement) sur grille détaillée
- Formulaires d'évaluation de la formation
- 10 jours avant l'entrée en formation : Nous transmettons à chaque stagiaire un questionnaire de préparation de la formation permettant d'adapter au mieux la formation à ses besoins.
- Pendant la formation : Le contrôle de connaissances et des acquis se font de façon continue via des quiz en ligne avec notation instantanée et des cas pratiques à réaliser. En fin de formation, les stagiaires restituent leurs travaux avec feedback du formateur.
- A l'issue de la formation : Nous adressons aux stagiaires un questionnaire de satisfaction à chaud et d'auto-diagnostic. Nous mesurons ainsi leur satisfaction et constatons directement l'évolution des compétences en comparaison avec les données recueillies avant la formation. Nous adressons un questionnaire d'évaluation au formateur.
- 15 jours après la fin de la formation : Nous adressons un questionnaire d'évaluation au financeur (OPCO).
- Un mois après la formation : Nous adressons un questionnaire d'évaluation aux différentes parties prenantes : commanditaire /manager.
- 3 mois après la formation : Nous transmettons à l'apprenant un questionnaire d'évaluation à froid pour évaluer l'impact de la formation dans son activité professionnelle.

Tarif de la formation

À partir de 14000.00 €HT
TVA non applicable, art. 261-4-4 du CGI