Synthese des Certifications Anthropic Academy

14 cours completes — Resume et concepts cles

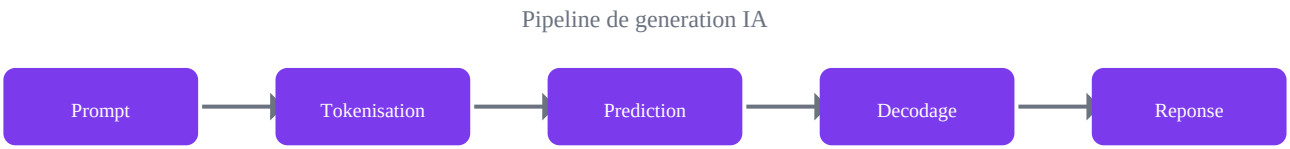
#	Cours	Categorie	Statut
1	Claude Code in Action	Developpement	Complete
2	Claude 101	Fondamentaux	Complete
3	Introduction to Claude Cowork	Productivite	Complete
4	AI Fluency: Framework & Foundations	IA Generale	Complete
5	Building with the Claude API	Developpement	Complete
6	Introduction to Model Context Protocol	Developpement	Complete
7	Model Context Protocol: Advanced Topics	Developpement	Complete
8	Claude with Google Cloud Vertex AI	Cloud / API	Complete
9	Introduction to Agent Skills	Developpement	Complete
10	Introduction to Subagents	Developpement	Complete
11	AI Capabilities and Limitations	IA Generale	Complete
12	AI Fluency for Educators	Education	Complete
13	AI Fluency for Students	Education	Complete
14	Teaching AI Fluency	Education	Complete

1. Comprendre l'IA : Capacites et Limites

Ce chapitre synthetise les cours **AI Capabilities and Limitations** et **AI Fluency: Framework & Foundations**. Ils forment la base theorique pour comprendre comment fonctionnent les modeles d'IA generative et comment collaborer efficacement avec eux.

1.1 Comment fonctionne l'IA generative

Les modeles comme Claude fonctionnent par **prediction du token suivant** (Next Token Prediction). Entraines sur d'enormes corpus de texte, ils apprennent a predire le mot le plus probable dans une sequence. Ce mecanisme simple donne naissance a des comportements emergents complexes : raisonnement, creativite, et resolution de problemes.



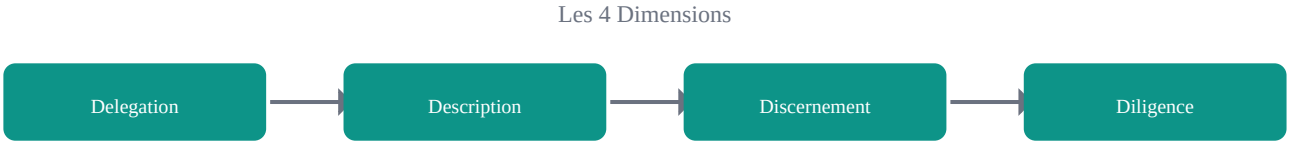
Le cours identifie **4 proprietes fondamentales** des modeles d'IA :

Propriete	Description	Implication pratique
Prediction	Le modele predit le token suivant le plus probable a partir du contexte	Peut generer du texte plausible mais factuellement incorrect (hallucinations)
Connaissance	Savoir encode lors de l'entrainement, avec une date limite de connaissance	Fiable sur les faits stables, moins sur l'actualite recente
Memoire de travail	Fenetre de contexte limitee (ex: 200K tokens pour Claude)	Les conversations longues peuvent perdre le fil ; structurer les prompts
Dirigeabilite	Sensibilite aux instructions et au format du prompt	Un prompt bien structure ameliore drastiquement la qualite des reponses

Cle : Quand ces proprietes entrent en conflit (ex: connaissance vs dirigeabilite), le modele peut produire des resultats inattendus. Identifier la source du probleme permet de le corriger avec une technique adaptee.

1.2 Le Framework 4D de l'AI Fluency

Le framework 4D structure les competences humaines necessaires pour collaborer efficacement avec l'IA. Il se decompose en deux boucles complementaires :



Boucle 1 : Delegation → Diligence

Delegation consiste a decider quelles taches confier a l'IA. Les taches ideales sont celles ou l'IA apporte un gain significatif (rapidite, volume) tout en permettant a l'humain de verifier facilement le resultat. **Diligence** est le complement : verifier, valider et prendre la responsabilite du resultat final. L'IA n'est pas un substitut au jugement humain.

Boucle 2 : Description → Discernement

Description est l'art de formuler des instructions claires pour l'IA : contexte, format souhaite, contraintes, exemples. **Discernement** consiste a evaluer de maniere critique les reponses de l'IA : detecter les erreurs, les biais, les hallucinations, et iterer sur le prompt pour ameliorer le resultat.

Exemple concret — Delegation vs. Non-Delegation :

Bonne delegation	Mauvaise delegation
Resumer un rapport de 50 pages en points cles	Prendre une decision strategique pour l'entreprise
Generer 10 variantes d'un email marketing	Diagnostiquer un probleme medical sans verification
Convertir du code Python 2 en Python 3	Ecrire un contrat legal sans relecture juridique

2. Claude 101 et Claude Cowork

2.1 Les fondamentaux de Claude

Claude est l'assistant IA developpe par Anthropic. Le cours Claude 101 couvre les bases : la premiere conversation, l'amelioration des resultats, l'application desktop, les projets, les artefacts, les skills et les connecteurs.

Modeles disponibles : Claude propose plusieurs modeles adaptes a differents usages. Opus 4.6 pour les taches complexes, Sonnet 4.6 pour un bon equilibre performance/cout, Haiku 4.5 pour les taches rapides et legeres.

Techniques pour de meilleurs resultats :

Technique	Description	Exemple
Contexte clair	Donner le role et le but	"Tu es un expert data. Analyse ce dataset..."
Structure XML	Utiliser des balises pour structurer	"<contexte>...</contexte> <tache>...</tache>"
Exemples (few-shot)	Montrer le format attendu	"Voici un exemple de sortie souhaitee : ..."
Chain of thought	Demander de raisonner etape par etape	"Reflechis etape par etape avant de repondre"
Temperature	Ajuster la creativite (0=deterministe, 1=creatif)	temperature=0 pour du code, 0.7 pour du creatif

Projets et Artefacts : Les projets permettent d'organiser les conversations par theme et d'ajouter des connaissances personnalisees (documents, instructions systeme). Les artefacts sont des contenus structurables (code, documents, visualisations) que Claude genere et que l'utilisateur peut editer, copier ou partager.

2.2 Claude Cowork : delegation de taches

Cowork est le mode de travail collaboratif de l'application desktop Claude. Il permet de deleguer des taches multi-etapes a Claude qui travaille de maniere autonome sur les fichiers reels de l'utilisateur.

Boucle de travail Cowork

Fonctionnalites cles :

- **Plugins** : extensions installables qui ajoutent des MCPs, skills et outils
- **Taches planifiees** : automatiser des taches recurrentes (ex: rapport quotidien)
- **Acces fichiers** : Claude travaille sur les fichiers reels via un dossier selectionne
- **Sous-agents** : delegation de sous-taches a des agents specialises en parallele
- **Connecteurs** : integration avec Slack, Google Drive, GitHub, etc.

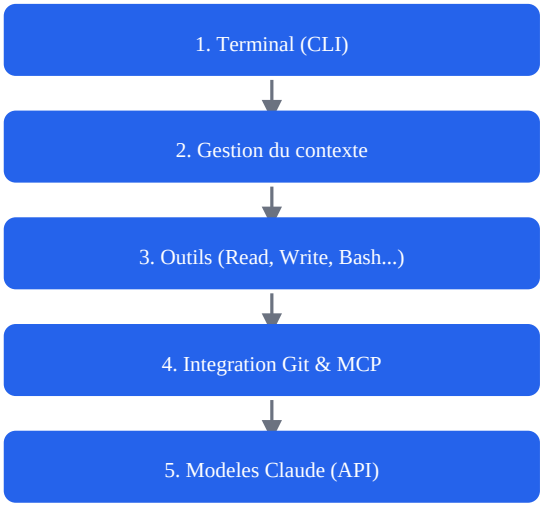
Principe : Cowork suit le principe Plan-Execute-Connect : planifier la tache, executer chaque etape, connecter les resultats aux outils externes. L'utilisateur garde le controle et peut intervenir a chaque etape.

3. Claude Code, Agent Skills et Subagents

3.1 Claude Code : l'outil CLI agentique

Claude Code est un outil en ligne de commande qui integre Claude directement dans le workflow de developpement. Il peut lire, modifier et creer des fichiers, executer des commandes shell, interagir avec Git, et acceder a des outils externes via MCP.

Architecture de Claude Code



Gestion du contexte : Le fichier `CLAUDE.md` est charge a chaque conversation et contient les instructions permanentes du projet (conventions de code, architecture, dependances). Il peut exister a 3 niveaux : personnel (`~/.claude/CLAUDE.md`), projet (`.claude/CLAUDE.md`), et repertoire courant.

Hooks : scripts personnalises declenches automatiquement a des moments cles :

Hook	Declenchement	Cas d'usage
PreToolUse	Avant chaque outil	Bloquer certaines commandes, valider les actions
PostToolUse	Apres chaque outil	Formater le code, lancer des linters
Notification	Quand Claude notifie	Alertes Slack, sons personnalises
Stop	Fin de tour	Verifications finales, CI/CD

3.2 Agent Skills : instructions reutilisables

Les Skills sont des dossiers de fichiers markdown que Claude Code decouvre et utilise automatiquement. Chaque skill contient un fichier `SKILL.md` avec un frontmatter (nom, description) et des instructions detaillees.

Fonctionnement :

1. Claude compare la requete aux descriptions de skills disponibles
2. Si un match est trouve, le skill est charge dans le contexte
3. Claude suit les instructions du skill pour executer la tache
4. Seuls les skills pertinents sont charges (economie de contexte)

Feature	CLAUDE.md	Skills	Slash Commands
Chargement	Toujours (chaque conversation)	A la demande (matching auto)	Manuel (l'utilisateur tape la commande)
Portee	Globale au projet	Specifique a une tache	Specifique a une action
Ideal pour	Conventions permanentes	Expertise reutilisable	Actions ponctuelles

Exemple de SKILL.md :

```
---
name: pr-review
description: Reviews pull requests for code quality
---

## Instructions
1. Verifier la couverture de tests
2. Detecter les code smells
3. Valider les conventions de nommage
```

3.3 Subagents : delegation parallele

Les sub-agents permettent a Claude Code de deleguer des sous-taches a des instances isolees qui travaillent dans leur propre fenetre de contexte. Le resultat est renvoye sous forme de resume, gardant le contexte principal propre et cible.

Bonnes pratiques :

- Definir un format de sortie structure pour chaque sub-agent
- Limiter les outils accessibles (principe du moindre privilege)
- Prevoir un mecanisme de rapport d'obstacles
- Eviter la sur-delegation : les taches simples ne necessitent pas de sub-agent

Cas d'usage : Les sub-agents sont particulierement utiles pour : la revue de code multi-fichiers, la generation de documentation, les recherches paralleles dans une base de code, et les analyses de securite. Chaque agent travaille dans son propre contexte isole.

4. Developper avec l'API Claude

4.1 Acces et premiers appels

L'API Claude permet d'integrer les modeles Anthropic dans des applications. Deux points d'acces principaux : l'API Anthropic directe et Google Cloud Vertex AI. Le cours couvre les deux approches avec 93 modules au total pour Vertex AI.

Exemple d'appel API (Python) :

```
import anthropic

client = anthropic.Anthropic()
message = client.messages.create(
  model="claude-sonnet-4-6",
  max_tokens=1024,
  messages=[
    {"role": "user", "content": "Explique le RAG"}
  ]
)
```

4.2 Prompt Engineering avance

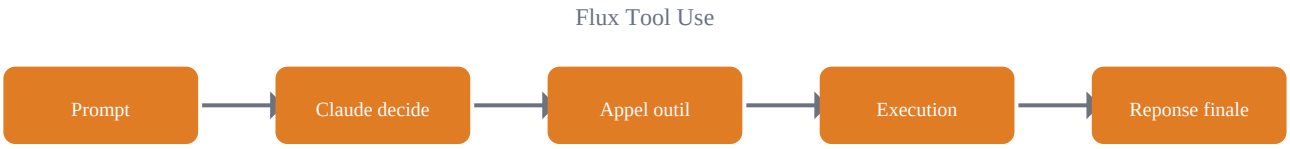
Le prompt engineering est la discipline de formulation des instructions pour obtenir les meilleurs resultats. Le cours couvre les techniques avancees :

Technique	Principe	Quand l'utiliser
Balises XML	Structurer le prompt avec des sections claires	Taches complexes avec plusieurs entrees

Technique	Principe	Quand l'utiliser
Few-shot learning	Fournir des exemples d'entrees/sorties	Format de sortie specifique, classification
Chain of thought	Demander un raisonnement explicite etape par etape	Problemes logiques, mathematiques, analyse
System prompt	Definir le role et les contraintes globales	Chaque appel API, personnalite coherente
Prompt caching	Reutiliser les prefixes de prompts frequents	Appels repetitifs avec contexte partage (reduction ~90% de latence)

4.3 Tool Use (Function Calling)

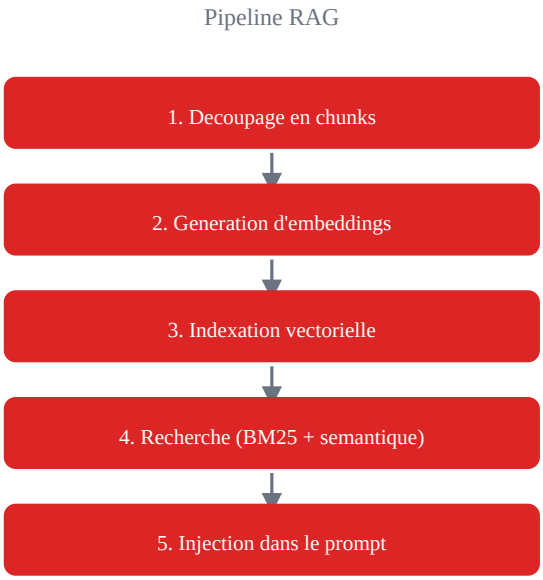
Le Tool Use permet a Claude d'appeler des fonctions externes definies par le developpeur. Claude decide quand utiliser un outil, genere les parametres au format JSON, et le developpeur execute la fonction puis renvoie le resultat.



Exemple : un outil `get_weather(city)` — Claude recoit la question "Quel temps fait-il a Paris ?", decide d'appeler l'outil avec `{"city": "Paris"}`, recoit le resultat `{"temp": 18, "condition": "nuageux"}`, et formule sa reponse finale : "Il fait 18 degres et nuageux a Paris."

4.4 RAG (Retrieval Augmented Generation)

Le RAG enrichit les prompts avec des documents externes pertinents pour ancrer les reponses dans des donnees factuelles. Le pipeline classique :



Le cours introduit egalement le **Contextual Retrieval** : technique ou chaque chunk est enrichi d'un contexte expliquant sa position dans le document original, ameliorant significativement la pertinence de la recherche.

Exemple de RAG avec recherche hybride :

```
# Recherche hybride : BM25 + semantique
bm25_results = bm25_search(query, chunks, k=20)
semantic_results = vector_search(query, embeddings, k=20)
# Fusion des scores avec Reciprocal Rank Fusion
merged = reciprocal_rank_fusion(bm25_results, semantic_results)
# Injection des top-K chunks dans le prompt
context = "\n".join([chunk.text for chunk in merged[:5]])
```

4.5 Fonctionnalites avancees

- **Vision** : Claude peut analyser des images (screenshots, graphiques, documents scannes)
- **PDF natif** : traitement direct de fichiers PDF avec extraction de contenu
- **Citations** : generation de references vers les sources utilisees
- **Extended Thinking** : mode de reflexion approfondie pour les problemes complexes
- **Streaming** : reception progressive des reponses pour une meilleure UX
- **Prompt Caching** : mise en cache des prefixes pour reduire latence et couts

4.6 Evaluation de prompts

L'evaluation systematique des prompts est essentielle en production. Le workflow type :

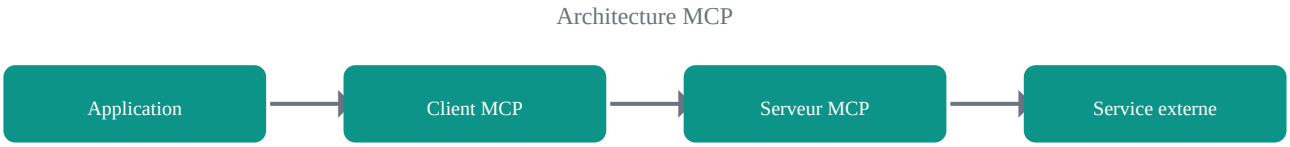
1. Definir des cas de test representatifs
2. Generer des datasets de test (manuels ou synthetiques)
3. Executer les evaluations avec des metriques definies
4. Grading automatise : par code (regex, exact match) ou par modele (LLM-as-judge)
5. Iterer sur le prompt jusqu'a atteindre les seuils de qualite

> **Best practice :** En production, combiner le grading par code (pour les criteres objectifs) et le grading par modele (pour les criteres qualitatifs) offre la meilleure couverture d'evaluation.

5. Model Context Protocol (MCP)

Le MCP est un protocole ouvert qui standardise la connexion entre les modeles d'IA et les services externes. Il couvre deux cours : l'introduction (14 lecons) et les sujets avances (15 lecons).

5.1 Architecture MCP



Le MCP deplace la charge de definition des outils vers des **serveurs specialises**. Au lieu d'ecrire des schemas JSON manuellement dans chaque application, on cree un serveur MCP reutilisable qui expose ses capacites via un protocole standardise.

5.2 Les trois primitives MCP

Primitive	Controle	Usage	Exemple
Tools	Le modele decide	Actions et calculs	get_weather(), create_issue()
Resources	L'application decide	Donnees en lecture seule	doc://readme, config://settings
Prompts	L'utilisateur decide	Instructions pre-construites	/format-doc, /review-code

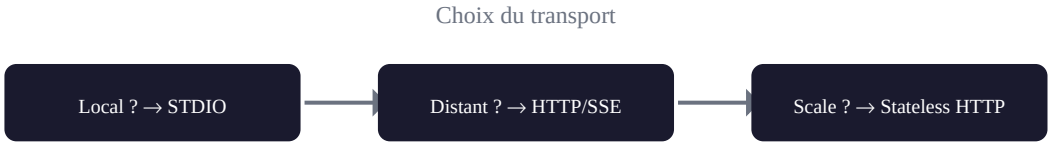
Exemple de serveur MCP (Python SDK) :

```
from mcp.server import FastMCP

mcp = FastMCP("mon-serveur")

@mcp.tool()
def lire_document(nom: str) -> str:
  """Lit le contenu d'un document"""
  return documents[nom]

@mcp.resource("doc://{nom}")
def document_resource(nom: str) -> str:
  return documents[nom]
```

5.3 Sujets avances MCP

Le cours avance couvre les mecanismes de production :

Concept	Description
Sampling	Permet au serveur MCP de demander des appels au modele via le client, creant des workflows IA-dans-la-boucle
Notifications	Systeme de feedback en temps reel : logs, progression, changements de ressources via des callbacks
Roots	Systeme de permissions pour accorder l'acces a des repertoires specifiques du systeme de fichiers
Transport STDIO	Communication par entree/sortie standard, ideal pour les outils locaux
Transport StreamableHTTP	Server-Sent Events (SSE) sur HTTP pour les deploiements distants et le scaling horizontal



Exemple de sampling (serveur demande au modele) :

```
@mcp.tool()
async def analyser_code(fichier: str) -> str:
  code = lire_fichier(fichier)
  # Le serveur demande au client d'appeler le LLM
  result = await ctx.session.create_message(
    messages=[{"role": "user",
      "content": f"Analyse ce code: {code}"}],
    max_tokens=500
  )
  return result.content[0].text
```

Notifications et progression : les serveurs MCP peuvent emettre des notifications en temps reel via `ctx.info()`, `ctx.warning()` et `ctx.report_progress(current, total)`. Cela permet au client d'afficher une barre de progression pour les operations longues.

> **Outil :** Le MCP Inspector est un outil integre pour tester et debugger les serveurs MCP dans un navigateur avant de les deployer. Il permet de tester les tools, resources et prompts de maniere interactive.

6. Claude sur le Cloud et AI Fluency appliquee

6.1 Claude avec Google Cloud Vertex AI

Ce cours de 93 modules couvre l'integration complete de Claude via Vertex AI. La principale difference avec l'API directe est la methode d'authentification (service accounts GCP au lieu de cles API Anthropic) et quelques parametres specifiques.

Specificites Vertex AI :

- Authentification via `google-auth` et service accounts
- Endpoint regional (ex: us-east5, europe-west1)
- Memes fonctionnalites que l'API directe : tool use, RAG, vision, streaming
- Integration native avec les services GCP (BigQuery, Cloud Storage, etc.)
- Facturation unifiee via le compte GCP

Exemple de setup Vertex AI :

```
import anthropic

client = anthropic.AnthropicVertex(
```

```
  region="us-east5",
  project_id="mon-projet-gcp"
)
# Ensuite, meme API que le client direct
message = client.messages.create(...)
```

6.2 AI Fluency pour l'education

Trois cours complementaires adaptent le framework 4D a l'education :

Cours	Public	Focus
AI Fluency for Educators	Enseignants, concepteurs pedagogiques	Integrer l'IA dans la conception de cours et les evaluations
AI Fluency for Students	Etudiants	IA comme partenaire d'apprentissage et outil de planification de carriere
Teaching AI Fluency	Formateurs, responsables pedagogiques	Enseigner et evaluer les competences 4D en contexte academique

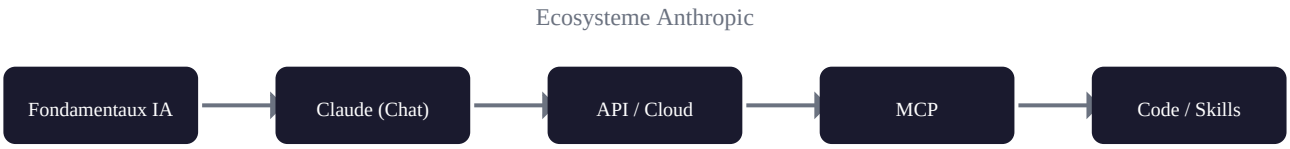
Concepts pedagogiques cles :

- **Boucle Delegation-Diligence** pour les enseignants : decider quelles taches pedagogiques deleguer a l'IA (generation de QCM, correction) tout en maintenant la rigueur
- **Boucle Description-Discernement** pour les etudiants : apprendre a formuler des requetes precises et a evaluer critiquement les reponses
- **Evaluation des 4D** : concevoir des exercices qui mesurent les competences de delegation, description, discernement et diligence
- **Impact disciplinaire** : comment l'IA transforme chaque discipline (droit, medecine, ingenierie, arts...) et quelles competences deviennent plus ou moins importantes

> **Principe cle :** L'IA ne remplace pas l'expertise humaine, elle la transforme. L'etudiant doit etre 'the human in the loop' : celui qui valide, contextualise et prend la responsabilite du resultat final.

Conclusion

Ces 14 certifications couvrent l'ecosysteme complet d'Anthropic : des fondamentaux de l'IA generative au deploiement en production via API et MCP, en passant par les outils de developpement (Claude Code, Skills, Subagents) et les applications en education.



Les points essentiels a retenir :

1. **Comprendre les limites** : la prediction de token, les hallucinations, la fenetre de contexte — pour mieux anticiper les comportements du modele
2. **Maitriser le framework 4D** : Delegation, Description, Discernement, Diligence — pour une collaboration humain-IA efficace et responsable
3. **Structurer ses prompts** : balises XML, exemples, chain of thought — pour des resultats precis et reproductibles
4. **Exploiter le MCP** : un protocole ouvert pour connecter Claude a n'importe quel service externe de maniere standardisee
5. **Automatiser avec les Skills** : capitaliser les bonnes pratiques dans des instructions reutilisables que Claude applique automatiquement