CONNEQT

The Agent Loop: How AI Thinks, Decides & Acts

The Complete Visual Guide to Autonomous AI Architecture

الوكيل: كيف يفكر الذكاء الاصطناعي ويقرر ويتصرف حلقة

الدليل المرئي الشامل لبنية الذكاء الاصطناعي المستقل

2026 Edition | conneqtme.com

1. What Is the Agent Loop?

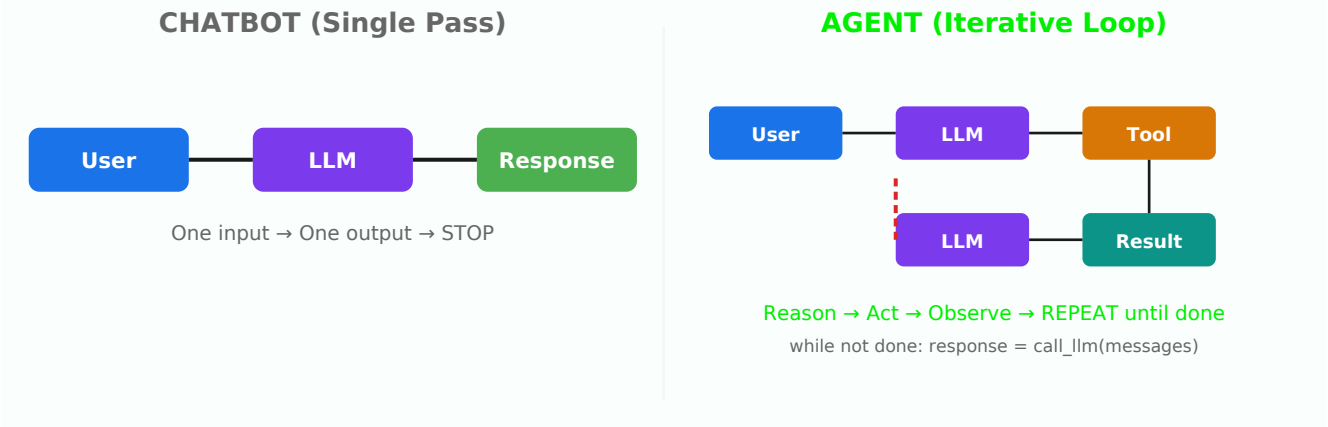
١. ما هي حلقة الوكيل؟

An Agent Loop is the continuous iterative cycle through which autonomous AI systems operate. Unlike a chatbot (one input → one output → stop), an agent runs in a persistent loop: it perceives the situation, reasons about the best next step, acts on that decision, observes the result, learns from the outcome, and repeats — until the goal is achieved or a stop condition is met.

خطوة تالية ويتصرف بناء على ذلك القرار ويلاحظ النتيجة ويتعلم من الحصيلة ويكرر — حتى يتحقق الهدف أو يتحقق شرط التوقف.
على عكس روبوت المحادثة (إدخال واحد → مخرج واحد → توقف)، يعمل الوكيل في حلقة مستمرة: يدرك الموقف ويفكر في أفضل
حلقة الوكيل هي الدورة التكرارية المستمرة التي تعمل من خلالها أنظمة الذكاء الاصطناعي المستقلة.

Chatbot vs Agent: The Fundamental Difference

روبوت المحادثة مقابل الوكيل: الفرق الجوهري

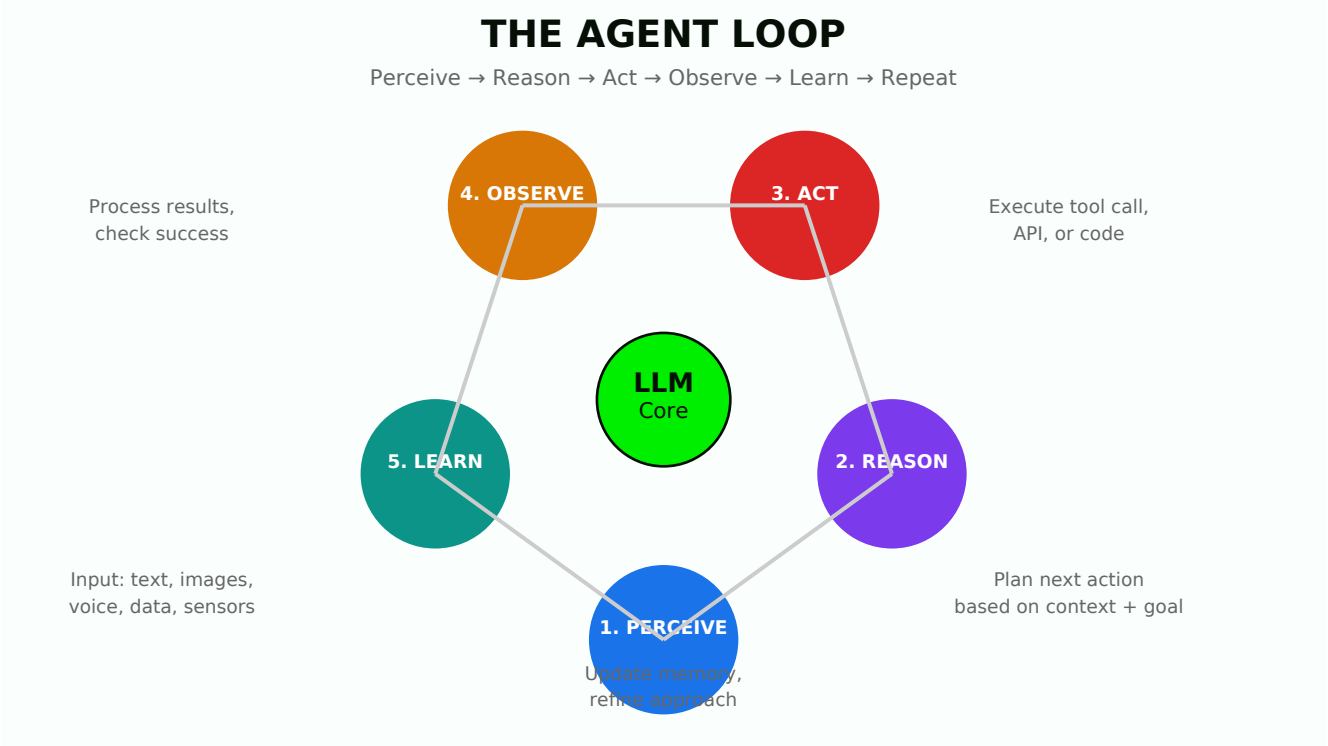


The pseudocode that powers every agent loop in existence is remarkably simple: while not done: response = call_llm(messages). If the response contains tool_calls, execute them, append the results, and loop again. If not, the task is complete. This simple pattern enables agents to solve problems that require multiple steps, multiple tools, and multiple rounds of reasoning.

الحلقة. إذا لم تحتو فالمهمة اكتملت. هذا النمط البسيط يمكن الوكلاء من حل مشاكل تتطلب خطوات وأدوات وجولات استدلال متعددة.
بشكل لافت: (while not done: response = call_llm(messages). إذا احتوت الاستجابة على tool_calls نفذها وأضف النتائج وكرر
الكود الزائف الذي يشغل كل حلقة وكيل في الوجود بسيط

2. The 5-Phase Agent Loop

٢. حلقة الوكيل ذات الخمس مراحل



Phase	What Happens	Example	المرحلة
1. Perceive	Captures input — text, images, voice, API data, sensor readings	User asks: 'Find the most cited AI paper from 2026'	الإدراك
2. Reason	LLM analyzes context, determines which tool to call and with what parameters	Agent reasons: 'I need to search academic papers — use search API'	الاستدلال
3. Act	Executes the chosen tool — API call, code execution, file operation, web search	Calls search API with query 'most cited AI paper 2026'	التنفيذ
4. Observe	Processes the tool's response, evaluates results, checks if goal is met	Receives 15 papers. Identifies top result with 340 citations.	الملاحظة
5. Learn	Updates context, refines approach, decides whether to loop again or stop	Needs full abstract — loops back to Act: calls document retrieval tool	التعلم

3. Core Components of an Agent Loop

٣. المكونات الأساسية لحلقة الوكيل

Component	Role	المكون
Orchestrator	The control center — manages the loop, routes between phases, handles errors, enforces stopping conditions (max iterations, token budget, goal achieved)	المنسق
Context Engine	Manages the conversation history, tool results, and accumulated knowledge across iterations. Handles context window limits via compaction and summarization.	السياق محرك
Memory System	Short-term (current session messages), long-term (persistent knowledge across sessions), and working memory (current task state). Critical for multi-session agents.	الذاكرة نظام
Reasoning Engine	The LLM itself — processes context, decides actions, generates responses. Quality depends on model capability (Opus > Sonnet > Haiku for complexity).	الاستدلال محرك
Tool Registry	Available tools the agent can call: APIs, code execution, file operations, web search, databases. Defined with name, description, and JSON Schema parameters.	الأدوات سجل
Feedback Loop	Evaluates outcomes, assesses performance, and adjusts future behavior. Includes success/failure detection, reward signals, and self-correction mechanisms.	الراجعة التغذية حلقة
Safety Layer	Guardrails, permission boundaries, rate limits, and human-in-the-loop checkpoints. Prevents runaway loops, excessive spending, and unauthorized actions.	الأمان طبقة

4. Agent Loop Patterns

٤. أنماط حلقة الوكيل

Pattern 1: ReAct (Reason + Act)

The most common pattern. The agent alternates between reasoning (thinking about what to do) and acting (executing a tool). Each cycle: Thought → Action → Observation. Used by Claude Code, ChatGPT, and most production agents. Simple, reliable, and well-understood.

النمط ١: ReAct (استدلال + تنفيذ)

كل دورة: فكرة → إجراء → ملاحظة. يستخدمه Claude Code وChatGPT ومعظم الوكلاء الإنتاجيين. بسيط وموثوق ومفهوم جيدا. أشهر نمط. يتناوب الوكيل بين الاستدلال (التفكير فيما يجب فعله) والتنفيذ (تنفيذ أداة).

Pattern 2: Plan-Execute

The agent first creates a complete plan (list of steps), then executes each step sequentially. Better for complex tasks with known structure. Risk: plan may become stale if early steps produce unexpected results.

النمط ٢: خطط-نفذ

بالتسلسل. أفضل للمهام المعقدة ذات الهيكل المعروف. المخاطرة: قد تصبح الخطة قديمة إذا أنتجت الخطوات الأولى نتائج غير متوقعة. الوكيل يضع أولا خطة كاملة (قائمة خطوات) ثم ينفذ كل خطوة

Pattern 3: Multi-Agent Orchestration

A supervisor agent delegates subtasks to specialized agents running their own loops. The supervisor coordinates results, handles failures, and synthesizes the final output. Used for complex tasks like full-stack development (frontend agent + backend agent + test agent running in parallel).

النمط ٣: تنسيق متعدد الوكلاء

الخاصة. المشرف ينسق النتائج ويتعامل مع الأعطال ويجمع المخرج النهائي. يستخدم للمهام المعقدة مثل تطوير المكدس الكامل. وكيل مشرف يفوض المهام الفرعية لوكلاء متخصصين يشغلون حلقاتهم

Pattern 4: Recursive Self-Improvement

The agent evaluates its own output, identifies weaknesses, and re-runs with improvements. Hermes Agent uses this pattern — after completing a task, it synthesizes the experience into a permanent skill for next time.

النمط ٤: التحسين الذاتي التكراري

ويعيد التشغيل مع تحسينات. هيرميس إيجنت يستخدم هذا النمط — بعد إكمال مهمة يجمع التجربة في مهارة دائمة للمرة القادمة. الوكيل يقيم مخرجاته ويحدد نقاط الضعف

5. Real-World Agent Loop Examples

٥. أمثلة حقيقية لحلقة الوكيل

Agent	Loop Pattern	Max Iterations	Tools Used
Claude Code	ReAct with Extended Thinking	Unbounded (token budget)	bash, file edit, file create, web search
OpenAI Codex CLI	ReAct with sandbox	Per-turn (user returns control)	code execution, file ops, shell
Hermes Agent	ReAct + Self-Improvement	300 iterations per session	MCP servers, shell, browser, cron
OpenClaw	ReAct + Dreaming	Continuous (always-on)	25+ messaging channels, browser, APIs
ml-intern	Plan-Execute + Iteration	300 iterations (smolagents)	arXiv, HF Hub, GPU training, eval
Devin / Cursor Agent	Plan-Execute	Task-bounded	code, terminal, browser, git

6. When Does the Loop Stop?

٦. متى تتوقف الحلقة؟

Stop Condition	How It Works	التوقف شرط
Goal Achieved	The agent determines its task is complete — generates a final response without tool calls	الهدف تحقق
Max Iterations	Hard limit on loop cycles (e.g., 300 for ml-intern). Prevents infinite loops.	للتكرارات الأقصى الحد
Token Budget	Total tokens consumed hits a cap. Claude Code uses this — prevents runaway costs.	الرموز ميزانية
Human Intervention	Agent requests approval before continuing. Used for high-risk actions.	بشري تدخل
Error Threshold	Too many consecutive failures. Agent gives up and reports what went wrong.	الأخطاء عتبة
Time Limit	Wall-clock time exceeded. PostTrainBench uses 10-hour limit.	حد الوقت

Key Insight: The most common production failure is infinite loops — agents that keep calling tools without making progress. Always implement both max iteration AND token budget limits. Monitor cost per session in production.

الأقصى وميزانية الرموز معا. راقب التكلفة لكل جلسة في الإنتاج.
يستمرون في استدعاء الأدوات بدون إحراز تقدم. نفذ دائما حدود التكرار
رؤية: أشهر عطل إنتاجي هو الحلقات اللانهائية — وكلاء

7. Building Your First Agent Loop

٧. بناء أول حلقة وكيل لك

The complete agent loop in pseudocode is surprisingly simple. Here's what every agent — from Claude Code to Hermes to ml-intern — runs under the hood:

في الكود الزائف بسيطة بشكل مفاجئ. هذا ما يشغله كل وكيل — من Claude Code إلى Hermes إلى ml-intern — تحت الغطاء: حلقة الوكيل الكاملة

The Universal Agent Loop (Pseudocode)

```
messages = [system_prompt, user_input] while not done: response = call_llm(messages) if response has tool_calls: for tool in response.tool_calls: result = execute_tool(tool.name, tool.args) messages.append(tool_result(result)) else: done = True return response.text
```

That's it. Every agent framework — LangChain, smolagents, CrewAI, AutoGen — is a variation of this pattern. The differences are in how they handle memory, tool routing, error recovery, multi-agent coordination, and stopping conditions. But the core loop is always: call LLM → if tools needed, execute and loop → if no tools, return result.

الوكلاء المتعددين وشروط التوقف. لكن الحلقة الأساسية دائما: استدعاء LLM → إذا احتاج أدوات نفذ وكرر → إذا لم يحتج أعد النتيجة. وAutoGen — هو تنويع على هذا النمط. الاختلافات في كيفية التعامل مع الذاكرة وتوجيه الأدوات واسترداد الأخطاء وتنسيق هذا كل شيء. كل إطار عمل وكيل — LangChain وsmolagents وCrewAI

Need help building AI agents for your business?
Conneqt designs and deploys agent systems for GCC businesses.
Visit conneqtme.com or contact us for a free consultation.

هل تحتاج مساعدة في بناء وكلاء ذكاء اصطناعي لعملك؟
كونكت تصمم وتنشر أنظمة وكلاء لشركات الخليج.
قم بزيارة conneqtme.com أو تواصل معنا لاستشارة مجانية.