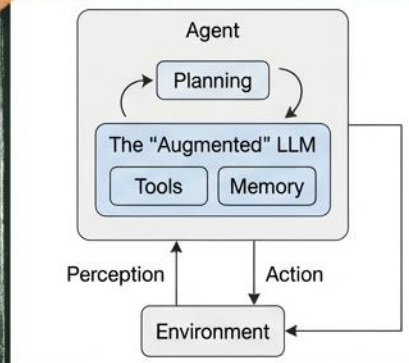
LLM agent 与 agentic coding tool：最小但准确的定义

LLM agent：以 LLM 作为“brain”，能推理/规划并执行多步任务，不是单轮回答的 chatbot。
agentic coding tool：面向 coding domain 的 LLM agent（例如 Claude Code / Codex）。

3 个关键组件：

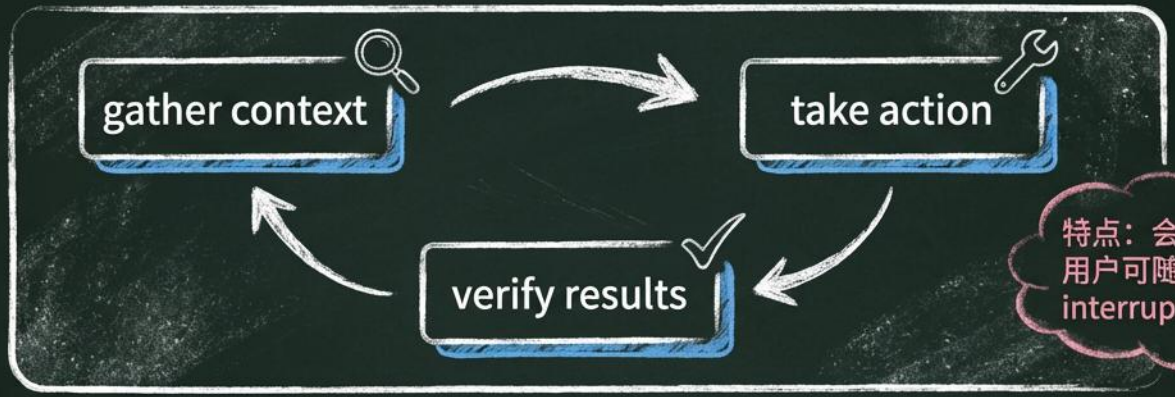
- (🧠) **Reasoning & Planning**：拆解目标 → 子任务 → 决定下一步动作
- (🛠️) **Tool Use**：调用外部工具（读/改文件、执行命令、搜索等）
- (📂) **Memory**：短期（当前上下文）/长期（跨任务沉淀）

为什么重要：把“读代码→改代码→跑命令→验证”串成闭环。



示意：brain + tools + memory

Claude Code 的 agentic loop：3 阶段 + tools



Tools 四类（每类 1 行）：

- File operations：读/改文件、重构
- Search：找文件、正则检索、定位引用
- Execution：跑命令、测试、git
- Web：搜文档、查报错、获取最新信息

个人体验：Claude Code vs Codex

Copilot(tab completion) → Cursor(heavy review) → Claude Code(placeholders 风险) → Codex(更干活但慢)

Claude Code（更 Vibe）

- 更“说人话”，更 Vibe/情绪价值
- 适合讨论需求/方向
- 风险：可能“完成了但代码是 placeholders”→ 需要 human review

Codex（更干活）

- 更像“真的干活”
- 慢；解释不一定好懂
- 需要更多人类 context switch 来把控节奏

结论：真实科研/项目里不能完全 vibe coding；需要掌握 feature/architecture/example。
Claude Code 更适合讨论需求；Codex 更适合按既定方案干重活。

方法论：Critical Rules + ralph-loop

（不丢信息、不半途而废）

File Purposes（3 个文件）：

File	Purpose	When to Update
`task_plan.md`	Phases / progress / decisions	After each phase
`findings.md`	Research / discoveries	After ANY discovery
`progress.md`	Session log / test results	Throughout session

Critical Rules（4 条）：

✅ **Create Plan First**：复杂任务先写 `task_plan.md`（non-negotiable）

⏱️ **The 2-Action Rule**：每 2 次 view/browser/search 后立刻写入文件

📖 **Read Before Decide**：重大决策前读 plan，避免偏航

📝 **Update After Act**：完成 phase 后更新状态 + 记录错误

```
while :; do
  cat PROMPT.md | claude-code
done
```

核心：prompt 不变、文件持续累积 → 循环迭代直到完成。

"Perfect combination"：`/ralph-loop ... --completion-promise "DONE"`（点到为止）