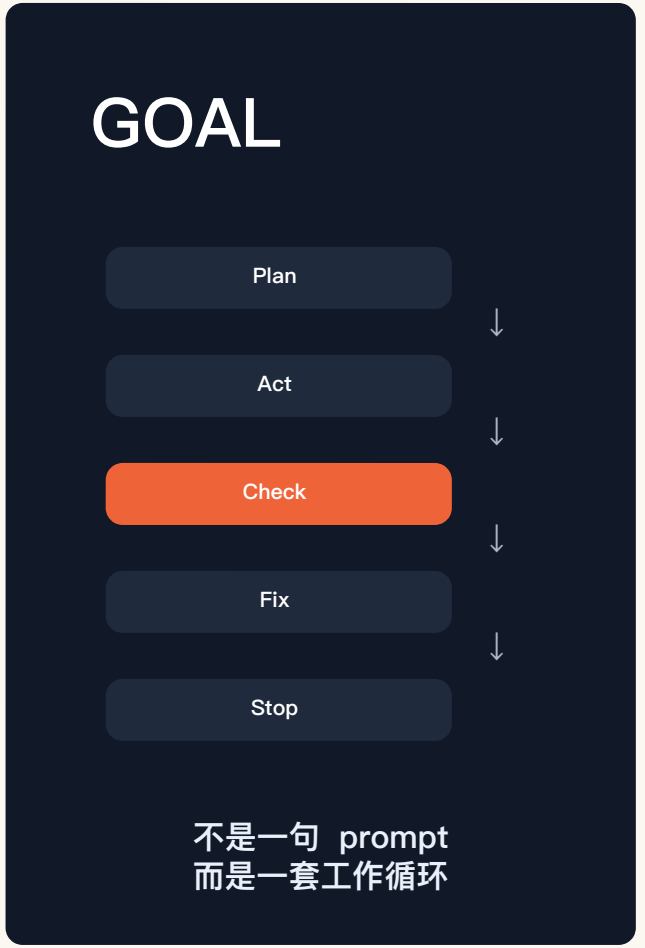
AI AGENT 方法课

Loop Engineering

从会聊天，到会交付

把目标、检查、记忆和停止条件设计成一个循环，让 Codex / Claude 自动推进到成品。

适合：不想只问 AI 要建议，而是想让 AI 产出文档、PPT、网页、SOP、研究报告的人



一句话定义

Loop engineering 是什么?

它不是让你写更长的 prompt，而是把“你不断提醒 AI 的动作”变成一个系统。

Prompt engineering

优化单次对话：这一句话怎么写，才能让 AI 答得更好。

Context engineering

优化输入材料：给 AI 什么背景、文件、规则、范例。

Loop engineering

优化工作循环：让 AI 计划、执行、检查、修正，直到满足目标或交回给你。

关键变化：人不再每一步都催促 AI，人负责定义目标、标准、边界和验收。

因为 agent 最大的问题，
不是不会生成，而是不会收尾。

一次性生成很容易；难的是发现不对、自己重跑、保留上下文、知道什么时候停。

1 **目标**
你要什么成品

2 **动作**
AI 调工具执行

3 **验证**
测试、预览、对照标准

4 **修正**
根据反馈继续改

5 **停止**
通过验收或请求人类

没有验证和停止条件，就不是 loop，只是让 AI 一直输出。

最简单的例子

任务：做一份 PDF 版 PPT

普通聊天方式

你：帮我做 PPT 。
AI：给你一个大纲。
你：第一页不好，改一下。
AI：改了。
你：能不能导出 PDF？
AI：你可以用某软件导出。

结果：人一直在推流程。

Loop engineering 方式

你先把目标聊透：受众、风格、页数、验收标准、禁区。
AI 复述目标后自动：查资料 → 写结构 → 做 PPT → 渲染预览 → 修错 → 导出 PDF 。

结果：人验收成品。

简单说：你不是“叫 AI 写一页”，而是“设计一个能把 PPT 做完的闭环”。

把例子拆开

一个好 loop 长什么样?

Goal	产出 10 页 PDF 版 PPT，讲清 loop engineering，普通人能照做。
Context	受众是非技术用户；中文；适合公开课；不要堆术语。
Plan	先查定义，再定故事线，再做 slides，再视觉检查。
Verification	每页是否可读、是否有例子、是否给出 Codex/Claude 用法、PDF 能打开。
Stop	PDF/PPTX 存在；预览无明显遮挡；最终回复给文件路径。

先聊透目标，再让 agent 复述

不要一开始就催它做。先花时间把“你到底要的成品”讲清楚，直到它能准确复述。

1. 讲业务

这份东西给谁看？用来成交、教学、内部培训，还是交付客户？

2. 讲成品

要 PDF 、 PPT 、网页、表格、 SOP ，还是一套资料包？页数和风格是什么？

3. 讲验收

什么算完成？什么绝对不能出现？要不要自动检查和导出？

核心动作：让 agent 先复述 goal ；复述正确后，再授权它自动分解和执行。

可以直接复制的开场

Goal 模板

我想让你用 goal-driven 的方式完成一个成品。

请先不要立刻制作，先和我聊清楚以下信息：
1. 目标受众是谁
2. 最终交付物是什么格式
3. 内容必须包含什么
4. 风格、语气、长度、禁区是什么
5. 你会如何验收这个成品

当你能完整复述我的 goal 后，请你：
- 自动拆成子任务
- 自己决定是否需要查资料、读文件、生成草稿、做视觉检查
- 中途不要反复问我小问题，除非会影响方向
- 最后给我可打开的成品和一段简短说明

Claude 更像策划编辑，Codex 更像执行工程师

Claude 中的用法

适合先把目标聊透：受众、话术、风格、结构、中文表达。

你可以让它先复述 goal，再生成讲稿、课程结构、网页文案、SOP。

Codex 中的用法

适合把目标变成文件：PPT、PDF、网页、脚本、表格、自动化流程。

你可以给它明确验收：运行、渲染、截图、修错、导出。

最稳的方式：Claude 帮你把 goal 写清楚；Codex 按 goal 生成和验证文件。

一个可落地流程

做任何成品，都按这 6 步

1

聊

聊业务目标和交付对象

2

复述

让 agent 复述 goal

3

授权

允许它自动分配子任务

4

执行

生成文件、调用工具、整理内容

5

验证

截图、测试、预览、对照标准

6

交付

给成品路径和修改说明

不要掉进这几个坑

Loop 越自动，越要有边界

坑 1：目标太虚

“帮我做一个高级的 PPT”没有验收标准。要写清受众、页数、场景、必须包含什么。

坑 2：没有检查

没有预览、测试、来源核对， agent 会把看似合理的错误包装成成品。

坑 3：不会停止

必须写明什么时候算完成，什么时候该停下来交给人。

普通人不需要会写代码，但必须会定义：目标、材料、标准、边界。

回到这次任务

我们现在的 goal

产出一份 PDF 版 PPT，用简单例子讲清 loop engineering，并告诉普通人如何在 Codex / Claude 中使用。

核对概念和来源 → 设计公开课式结构 → 生成可编辑 PPTX → 渲染页面并检查 → 导出 PDF 和文件路径

这就是 loop engineering：把“我要反复催你”的过程，变成 agent 自己执行的闭环。

最后记住一句话

Prompt 是一句话。

Loop 是一套交付系统。

普通人真正要学的，不是更多术语，而是把目标讲清楚，把标准讲清楚，然后让 agent 自己推进到成品。

下一步：把你的课程、服务、交付流程，改写成一个可复用的 goal 模板。