二八定律

- 任何人学习编程，哪怕赚不到钱，也可以赚到时间
- 用编程提升你的效率。
- 学会得意忘形：看一遍教程，可以忘记细节，只要脑子里面有个印象就可以，不需要死记硬背。

概念学习

我想学习【X】，清按照二八法则制定一个全面的学习计划，重点关注能让我开始构建项目的20%的核心概念。请将计划按周安排，总计【Y】周，每周涵盖特定的学习主题。
在完成这【Y】周的核心学习后，请推荐5个难度递增的项目（从入门到进阶），帮助我应用和拓展【X】的知识。对于每个项目，请提供简要描述并列出它将帮助强化的关键概念。
请确保计划详细到足以让初学者跟随，同时也要有足够的挑战性来培养独立思考和解决问题的能力。

- 接受一知半解的状态，对于程序员来说一知半解是常态。

过程加深

我正在学习【X】中的【Y】，请提供：
1.简明扼要的解释【Y】，包括用途和常见使用场景
2.一个演示【Y】的简单代码示例。
3.初学者关于【Y】常见的三个错误或误解，以及如何避免它们。
4.两个【Y】特别有用的实际应用场景或用例。
5.三个难度递进的练习题，帮助我练习使用【Y】。请只提供题目描述，不要提供解答。
在提供以上信息后，请向我提出一个关于【Y】的发人深省的问题，引导我更深入的思考它的应用或影响。

- 基础学习完之后，要达到的状态：能够看懂一个实战教程
- 了解一个项目是怎么糊到一起的。
- 学会测试/调试/各种工具

实战演练

- 这里主要扮演两个角色，planner和debugger，而不是coder
- 尽可能运用AI弄懂AI写的代码
  我准备开始制作【X】项目，这个项目涉及【Y】。我目前在【Z】方面的水平属于【高级/中级/初级】。
  请提供：

1.这个项目应该具备的组件和功能，按层次叙述。
2.建议的项目结构，包括需要创建的主要文件或模块。
3.在构建这个项目时可能遇到的三个关键挑战，以及克服这些挑战的总体策略（不需要具体的代码解决方案）。
4.在完成基本功能后，可以尝试实现的两个进阶目标，以提升项目的复杂度。
5.这个项目将帮助我强化或学习的三个【Z】相关的重要概念或技能。
请以引导思考的方式来回答，而不是提供明确的解决方案。我希望在实现细节方面受到挑战，自己去寻找解决方案。

- 循序渐进，从简单到复杂，不要一上来就上强度！
- 慢慢摸到他的能力上限。在他的能力范围内，找到能赚钱的套利空间。
- 分两分钟上线：单次对话的上限，完成一个项目的复杂度上限。
- 工具还在发展，上限还在提高，每一两个月可以关注一下AI工具的更新内容。