**数据与模板的角色分工**

1. YAML —— 数据存储

特点：

支持字典、列表结构。

语法简洁、层次清晰，易读易改。

读取后在 Python 中变为 dict/list，方便代码调用。

用途：

存储参数配置、提示词片段、摘要信息等。

适合做「动态内容的数据源」。

2. Jinja2 —— 模板渲染

核心原则：

不会碰普通文本（保持缩进、换行、Markdown 等格式）。

只处理 3 种语法：

{{ ... }} → 表达式求值 → 输出字符串

{% ... %} → 控制语句（for/if/宏/include …）

{# ... #} → 注释，不输出

其它文本（比如 Markdown、XML、自定义标签）都被视为普通字符串。

灵活性：

Jinja2 语法可以放在任意位置。

例如自定义 DSL 组件：

<system role>

{{ system\_prompt }}

</system role>

渲染后 <system role> 仍然是普通文本，不会被模板引擎改动。

用途：

定义提示词模版（系统提示、用户输入、输出格式）。

通过传入 YAML/数据库中的变量，实现动态填充。

3. Markdown —— 文本与代码片段

特点：

自然语言友好，LLM 训练语料多。

内置 fenced code block (python ... )，适合存放示例代码。

用途：

存储示例代码（便于人类阅读和 LLM 消化）。

用 front-matter（YAML 头部）存元数据（id、title、tags、libs…）。

4. 数据库 —— 索引与查询

作用：

数据库（SQLite / 向量库）只存 索引信息和路径：

id / title / tags / libs / path / embedding

查询时返回路径，再从本地加载对应 md 文件 → 得到代码/文本。

优势：

版本管理：代码片段还是放在文件系统（Git 管理）。

查询效率：数据库只管索引、过滤、检索，不承载大段文本。

可扩展：既支持关键字/标签过滤，也可接入向量检索。

**✅ 一句话总结：**

YAML → 存储结构化数据，读取后是字典/列表，方便程序使用。

Jinja2 → 模板引擎，只渲染 {{}}、{% %}、{# #}，其他一切保持原样。

Markdown → 最适合存放代码和文档，人/LLM都好读。

数据库 → 用于索引和查询，存文件路径/元数据，真正的代码片段仍保存在本地 .md 文件中。