程序设计训练（Rust 语言）

第 9 讲：AI Agent 大作业

于新雨

清华大学

2025–2026 学年度夏季学期

本讲内容

9.1 作业说明

9.1.1 AI Agent 大作业

9.1.1.1 这次要做什么

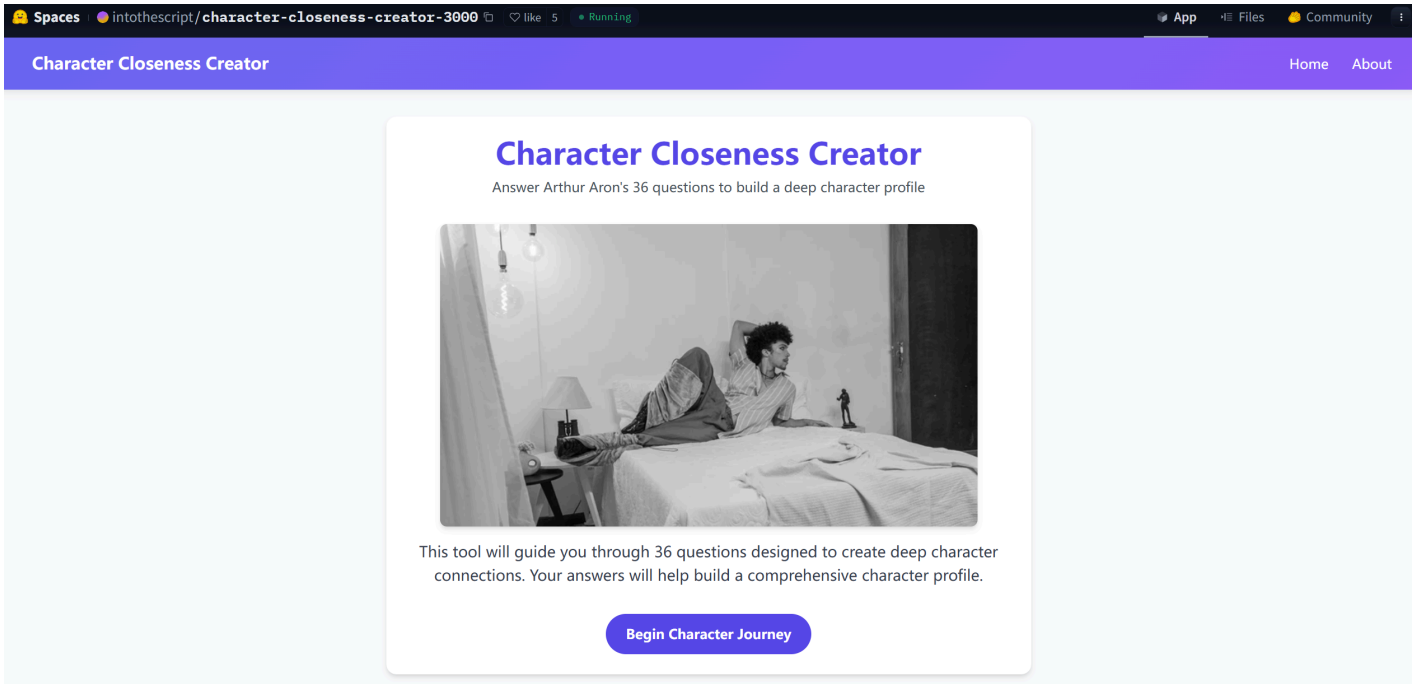
- 这次作业没有固定题目，由你自己发现问题、设计系统，并完成实现。
- 目标：做出一个真正能为你推进具体任务的专属 Agent。
- 交付目标：问题真实、方案具体、过程可运行、结果能解释。

9.1.1.2 先想一个真实的问题

- 从“有没有一件事经常重复、很麻烦、希望有人帮忙做”开始，或者是你觉得足够有趣、且可以由大模型完成的任务。
- 优先选择有明确输入、处理步骤和可验证结果的任务。
- 例子：整理照片、自动记账、跟踪比赛、汇总某类学习资料。

9.1.2 案例：人物设定生成器

- 用户回答 Arthur Aron 的三十六问，Agent 根据回答生成完整人物设定。



9.1.3 案例：算命 Agent

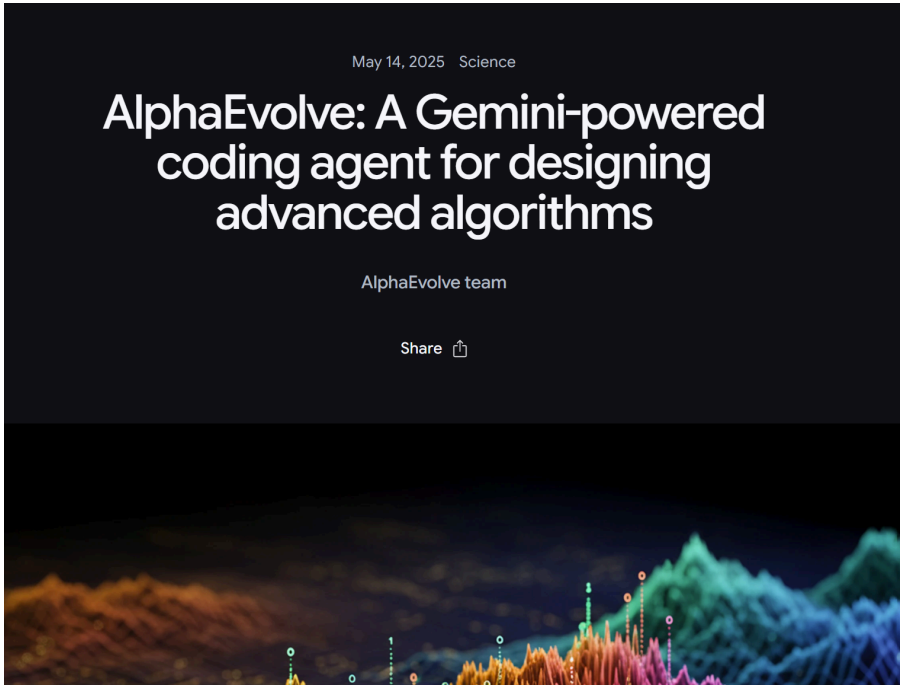
- 用户输入自己的“生辰八字”，Agent 挑一种命理（算命方法），给出运势等信息。

9.1.4 案例：Minecraft Agent

- 用一个 Agent 代替玩家完成 Minecraft 中的日常任务，例如收集资源、建造房屋等。

9.1.5 案例：算法 Agent

- 用 Agent 优化现实中的算法，如加速 TPU 的矩阵乘等。



9.1.6 为什么今年做 Agent

- 往年的大作业如 Wordle、OJ，都是写一个规范写得很细的固定程序，AI 照课程文档就能写出完整代码。
- 只会“把代码写出来”不够。真正要紧的是：发现值得解决的问题、针对它做设计，并保证做出来的东西可靠。
- 过去：实现固定程序；现在：围绕目标设计自己的工具和工作流。

9.2 快速入门

9.2.1 什么是 AI Agent

9.2.1.1 Agent 由什么组成

- Agent = 大模型 + 工具 + 状态 + 能自主推进任务的程序。
- 大模型负责理解目标、判断下一步和生成结构化决策。
- 程序负责执行工具、保存状态、处理错误，并控制权限和边界。

9.2.1.2 和聊天机器人有什么区别

- 聊天机器人（比如没开“智能搜索”的 DeepSeek 网页端）主要针对当前问题生成回答。
- Agent 可以查询数据、读写文件、调用 API，并完成多步任务。
- 对比：“帮我识别照片上的人”是回答；“帮我按日期和地点整理照片”是执行任务。

9.2.2 好选题的三个问题

- 这个问题经常出现、值得投入时间解决吗？还是你纯粹觉得有趣，别的人也会想用？
- 通用 Agent 做不好吗？里面有没有领域知识或专用数据的门槛？
- 有没有可以自动化的步骤？结果又是否能够检查？

9.2.3 一个具体例子：MOBA 比赛 Ban-Pick 分析助手

- 通用模型通常不知道最新版本、比赛数据和英雄强度。
- 专用 Agent 接入版本信息、历史对局和英雄统计等数据源。
- Agent 根据用户的问题调用数据工具，给出有证据的阵容分析和建议。

9.2.4 什么样的选题合格

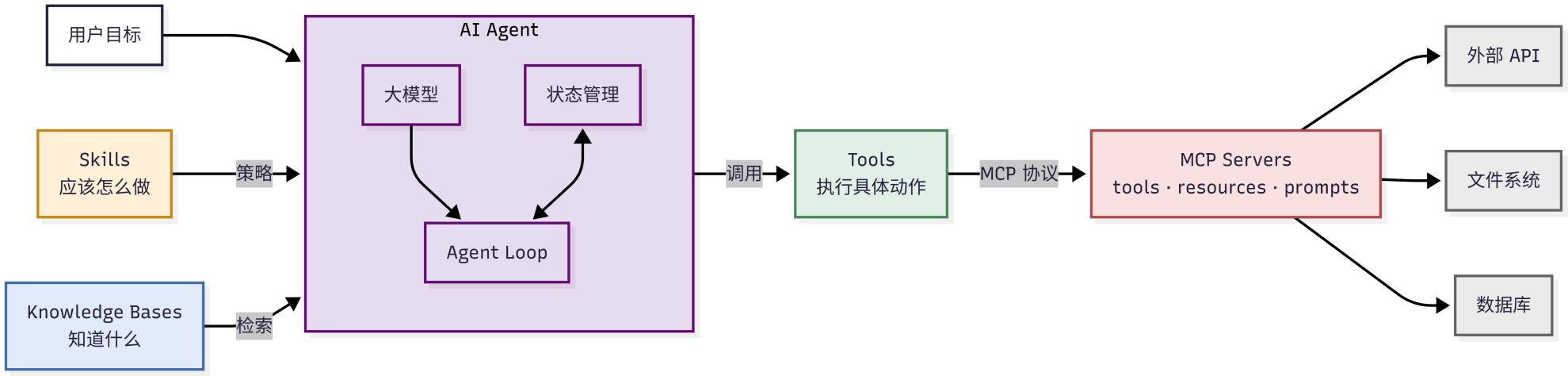
- 真实：来自个人、团队或具体领域的实际需求或者应用场景。
- 具体：有清晰边界、输入、输出和成功标准，不能只是“做一个助手”这样一句话。
- 可实现：有可获取的数据或工具，有一条能运行和演示的工作流。
- 例子：“学习助手”过于宽泛；“论文复现助手”更容易落地和评估。

9.2.5 可以做哪些方向

- 学习科研辅助：论文阅读、实验记录、资料检索、错题归纳。
- 生活资料管理：照片整理、账单分类、文件归档、日程规划。
- 特定领域工作流：比赛分析、表格审核、日志排查、报告生成。
- 娱乐体验增强：人物设定、游戏探索、剧情互动、内容推荐。

9.2.6 Agent 是怎样工作的

- 用户提出目标后，Agent 将任务交给模型进行理解和规划。
- 模型决定是否调用工具；Rust 程序执行调用并把结果写回状态。
- 数据、知识库和历史状态共同支持下一步判断，直到任务完成或停止。



9.2.7 工具调用与 Agent 循环

9.2.7.1 工具调用（Tool Call）

- 模型不会直接访问文件或 API，只会提出“请调用某个工具”的请求。
- 请求包含工具名和结构化参数，例如 `get_match_data(987)`。
- Rust 程序校验参数、执行实际操作、处理错误，再把结果返回给模型。

9.2.7.2 Agent 循环（Agent Loop）

- 循环执行：模型判断 -> 调用工具 -> 返回结果 -> 再次判断。
- 每一轮都更新任务状态，模型据此决定继续、修改计划或直接回答。
- 必须设置最大步数、取消机制、超时和费用上限，避免失控。

9.2.8 几种常见定制能力

- 工具解决“能做什么”：连接 API、文件、数据库或外部程序。
- 知识库解决“知道什么”：提供领域资料、规则、历史数据和引用依据。
- Skills 解决“应该怎么做”：把经验固化为步骤、策略、检查项和输出格式。
- MCP server 通过统一协议，让 AI Agent 安全、标准地发现并调用外部工具、数据和服务。

9.2.9 作业必须有场景定制

- 每个项目至少做两项专门优化，不能只调通用聊天接口。
- 可选方向：领域知识库、专用工具链、定制 Prompt、UI 设计。
- 说明每项定制解决了什么问题，并用示例或实验比较改进前后的效果。

9.2.10 推荐开发路线

- 先用手工准备的数据验证想法，再完成一次最小模型调用。
- 接入一个工具，打通“模型请求->程序执行->结果返回”流程。
- 加入多步循环、知识库和必要的状态管理。
- 最后再完善 UI、错误处理、历史记录和整体体验。

9.3 作业要求

9.3.1 六项固定要求

- R1：核心逻辑用 Rust 实现。
- R2：提供用户交互界面（Web / CLI / 桌面 / 手机）。
- R3：模型与参数可配置，不写死。
- R4：实时展示进度，支持打断。
- R5：上下文历史管理，可保存/加载。
- R6：统计 Token 用量与费用。

9.3.2 要提交什么

- 设计文档：痛点分析、场景定制方案、系统架构图、技术选型。
- 源代码与 README：能够安装、运行和复现主要流程。
- 完整 AI 对话历史：保留真实的设计、开发和调试过程。
- AI 开发开销明细表：按阶段记录模型、调用量、Token 和费用。

9.3.3 时间节点与评分

- 选题确认与互相评论：
  - 8.30 23:59:59 之前，在网络学堂讨论区发布选题。
  - 9.1 23:59:59 之前，给至少其他 3 位同学的选题发布评论。
- 公开展示与互相试用：
  - 9.6 23:59:59 之前，在网络学堂讨论区发布设计文档摘要和项目链接。
  - 9.8 23:59:59 之前，至少试用 3 位其他同学的作品，并提交反馈。
- 最终课堂展示：
  - 9.10 上午第二大节，进行展示与互评。

9.3.4 使用 AI 的原则

- 可以用 AI 辅助写代码，但必须理解、检查，并能够解释提交内容。
- 保留真实对话记录，展示需求分析、设计、实现、调试、修正的过程。
- 不复制他人的设计或实现；引用外部资料时要注明来源。
- 留意 API 费用、敏感数据和隐私，别把不应外传的内容提交给模型。

9.3.5 最后检查清单

- 项目是否解决一个真实而具体的问题？
- 是否至少有两项场景定制，而不是套用通用聊天？
- 是否能稳定运行、现场展示，并解释应用场景和关键设计？
- 现在就写下：用户是谁、需求和应用场景是什么，Agent 怎么替他完成？