30天 AI Agent 实战

每天一小时，从可运行的小例子，到可核验的科研证据助手。

先下载实践包并打开practice目录。每课按15分钟理解、30分钟操作、10分钟验收、5分钟记录推进。忙碌日可用20分钟精简路线；安装与模型下载时间可以单独安排。中断不清零，完成前以实际结果勾选验收。

离线主线是明确标注的规则教学工作流；可选接入真实模型。语料和CSV为教学材料，不承担真实科学结论。课程覆盖原型构建、验证与交付，模型训练和生产规模运维需另行深入。

课程目录

01 · 把第一个 Agent 项目跑起来
02 · 用 JSON 给助手立规矩
03 · 接上模型，也保留可重复的基线
04 · 把模型输出变成可检查的结果
05 · 给 Agent 配一组可信的工具
06 · 做一个会停下来的行动循环
07 · 第一周作品：一张可复核的证据卡
08 · 给材料建立来源与版本档案
09 · 做一个可以解释的检索基线
10 · 让回答带着证据一起出现
11 · 用小评价集检查检索与拒答
12 · 保存状态，让任务能够继续
13 · 让长期记忆有归属、期限与删除
14 · 第二周作品：一个有边界的证据问答助手
15 · 用依赖图组织多步骤任务
16 · 失败以后，怎样重试才合理
17 · 先审阅，再执行写入
18 · 看见 MCP 的真实消息往返
19 · 把现有流程迁移到 LangGraph
20 · 让研究者、质疑者和复核者各司其职
21 · 阶段作品：带质量门的文献研究助手
22 · 用 CSV 工具完成一份可复核的数据摘要
23 · 把外部指令留在数据边界内
24 · 建立开发集与测试集的评价习惯
25 · 记录轨迹、耗时与成本边界
26 · 给研究助手加一个本地网页
27 · 用缓存与并发控制重复工作
28 · 把作品整理成别人能复现的项目
29 · 为未知、冲突与攻击建立失败案例册
30 · 毕业交付：可复现的科研学习 Agent

第01天 · 把第一个 Agent 项目跑起来

先拿到一次完整运行，再认识模型、工具与任务边界。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 在 VS Code 中打开并运行实践包。

- 区分聊天、固定工作流和模型参与决策的 Agent。

- 写出输入、允许动作、交付物和停止条件。

开始之前

- 一台可以安装Python的电脑；主线无需GPU或API密钥。

- 下载实践包并解压，能找到practice目录。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

一个月的目标，是做出可以解释的作品

这一个月围绕科研证据助手展开：输入一个问题，查找指定材料，按需要调用工具，最后留下能够核对来源的结果。每天以一次小实验和一个具体文件衡量进展。30天是学习顺序；忙碌时可以分两次完成，不需要为了连续打卡而跳过理解。

课程主线采用Python标准库和明确标注的离线规则。它让没有GPU或API账号的电脑也能检查接口、数据和控制流程。第3天提供真实模型路径；离线运行成功与真实模型能力是两项不同的验证。

把聊天、工作流和 Agent 分开

让模型将一段话改写成三句话，可以是一次模型调用。读取CSV、检查缺失、输出固定报告，可以预先写成工作流。如果系统需要根据不同问题选择下一步工具、观察结果后调整行动，才涉及模型参与的行动决策。结构越复杂，越需要明确成本与停止条件。

本课的offline模式是教学工作流，不会隐藏调用大模型。你将先学会外围程序如何约束、执行与检查，再接上模型。这能帮助你判断：一个结果来自检索、计算、固定规则还是模型，而不是把所有输出都归功于所谓智能。

让环境准备尽量简单

先安装Python 3.10或以上版本与VS Code，再解压实践包。用VS Code的“文件→打开文件夹”选中能直接看到run.py、agentkit和data的practice目录，然后点“终端→新建终端”。网页负责阅读与记录，Python命令在你电脑的终端运行，不要输入到带有>>>的Python交互窗口。

Windows先试python --version，找不到时试py --version。建议建立.venv环境并直接使用其中的解释器：Windows是.\.venv\Scripts\python.exe，macOS/Linux是.venv/bin/python。这样无需修改PowerShell执行策略。下文统一写python，你可以替换为环境内的完整路径。

先写任务边界，再增加功能

合适的起步任务是“依据给定教学资料解释关联与因果的区别，并列出没有得到支持的部分”。它有有限输入和可检查的终点。“自动完成所有微生物研究”则缺少输入、权限、交付标准和结束条件，不能靠一次运行判断成功。

今天先写四句话：材料从哪里来；允许读取或写入什么；最后得到哪个文件；遇到什么情况停止。后面会把这些文字逐渐转成JSON参数、工具白名单、预算和人工批准。现在只生成任务卡，不宣称已经实现整个行动循环。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 确认目录与解释器

为什么做：先排除路径问题。

在practice目录打开终端；最后一行必须为True。

```
python --version
python -c "from pathlib import Path; print(Path.cwd()); print(Path('run.py').exists())"
```

预期结果：Python版本不低于3.10；当前目录存在run.py。

2. 创建独立环境并执行第一课

为什么做：避免影响已有科研环境。

Windows运行下面两行；macOS/Linux改为python3 -m venv .venv和.venv/bin/python run.py day 1。

```
py -3 -m venv .venv
.\.venv\Scripts\python.exe run.py day 1
```

预期结果：终端输出本课JSON，生成outputs/day01.json。

3. 读懂结果的层次

为什么做：知道本次程序真正完成了什么。

打开输出，找到day、title、mode和result，确认是offline教学结果。

```
python -m json.tool outputs/day01.json
```

预期结果：JSON可正常格式化；你能指出任务卡与模式标记。

4. 把目标写成四句话

为什么做：让后续每个功能都有检查依据。

在学习笔记写输入、允许动作、交付物、停止条件；对照experiments.json的day01配置。

预期结果：留下一张小任务卡；不把任务卡误认成已执行的自主Agent。

改一改 · 改变目标与预算，观察任务卡

- 打开experiments.json，将day01.goal改成“从给定资料整理关联与因果的证据说明”。

- 保持day01.max_steps不变，保存后运行python run.py day 1；这一轮只修改目标文字。

- 对比result.task_card中的goal；恢复原值。若还有时间，再单独将max_steps改为4做第二轮实验。

观察什么：任务卡中的目标随配置变化；单独改变步数时只改变预算说明，第1课不会自动执行模型调用。

机制解释：把任务说明写成配置，是可检查程序的第一步；真正的行动循环在第6课出现。

常见问题

提示找不到run.py

原因：终端没有位于practice目录。

处理：用VS Code重新打开practice文件夹，再开终端；检查Path('run.py').exists()。

PowerShell不允许激活环境

原因：系统阻止执行激活脚本。

处理：直接使用.\.venv\Scripts\python.exe执行命令，无需调整系统策略。

python命令不存在

原因：解释器未安装或未加入路径。

处理：Windows尝试py；macOS/Linux尝试python3。仍失败则从Python官网安装后重开终端。

03 • 验收

第1课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day01.json

☐ 可以从新终端再次运行第1课。

☐ 能解释offline模式及本课任务卡的范围。

☐ 任务说明包含输入、权限、交付物与停止条件。

自测与解析

1. 离线示例运行成功，能够证明什么？

- A. 某个真实模型已能自主科研

- B. 这组输入下教学流程能够运行

- C. 所有未来输入都可靠

- D. 研究结论已经得到验证

答案：B。 运行成功证明当前流程可执行，不能替代模型测试、真实数据分析或科学论证。

2. 第一版助手最适合哪种任务？

- A. 处理一切科研事务

- B. 自动作出临床判断

- C. 从给定三份资料整理可复核证据卡

- D. 无限搜索直到得到肯定结论

答案：C。 有限输入、清楚输出与可检查的来源，使任务能够逐步实现和验收。

小词典

- Agent：模型参与选择下一步动作，外围程序负责约束、执行和检查的系统。

- Workflow：主要步骤与转移规则预先确定的工作流。

- 虚拟环境：隔离一个项目的Python解释器和第三方依赖。

04 · 我的记录

- 今天的困难属于安装、路径、程序还是任务定义?

- 下次能否在两分钟内重新运行这份结果?

忙碌日：20分钟

- 用3分钟读“一个月的目标，是做出可以解释的作品”，写下一个关键区别。

- 用12分钟运行 python run.py day 1，打开 outputs/day01.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 为自己的科研工作写任务卡

- 选择一个真实工作场景，但只用公开或虚构材料。

- 把输入、允许动作、交付物与停止条件分别写成一句话。

- 删去当前无法验收的宽泛目标。

参考资料与阅读范围

Python 官方：虚拟环境：阅读 Creating virtual environments；本课使用解释器完整路径，无需改变系统执行策略。
https://docs.python.org/3/library/venv.html

Anthropic：Building effective agents：阅读 workflows 与 agents 的区别，以及从简单结构开始的讨论。
https://www.anthropic.com/research/building-effective-agents

下一步

下一课把任务中的文字约定转成JSON和程序可以检查的数据结构。

第02天 · 用 JSON 给助手立规矩

把自然语言要求变成类型、范围与必填字段。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 分清JSON语法正确与内容有效。

- 识别字符串、数字、空值和布尔值的差别。

- 用错误输入检验参数边界。

开始之前

- 完成第1课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

JSON是交换格式，不是质量保证

JSON用对象、数组和基本值描述数据。它适合在模型、工具与文件之间传递信息，因为程序能明确读取字段。但能够被JSON解析器读入，只说明文本遵守格式；它不保证question有意义，也不保证top_k位于允许范围。

例如{"top_k":3}和{"top_k":"3"}都可以是合法JSON，前者是数字，后者是字符串。工具要的是整数时，应明确拒绝不合适的类型，而不是在每个位置猜测用户本意。边界越清楚，错误越容易定位。

一个参数同时有名字、类型和范围

本课围绕question与top_k练习。question应是非空问题，top_k表示最多取多少条检索结果。契约应交代哪些字段必需、是否允许额外字段、参数类型以及上下限；缺少其中一项，都可能把错误推迟到真正执行时才暴露。

Python中bool是int的子类，这意味着简单的isinstance(x,int)可能让True通过。对于希望排除布尔值的计数参数，需要更严格的类型检查。课程检验的是已明确列出的约束，不声称自动理解所有问题语义。

拒绝错误，比猜一个结果更容易检查

收到缺字段、空问题、超范围数字或不允许的额外字段时，程序应该返回可读的验证错误。这样使用者知道要修正输入，工具也不会在错误假设下继续执行。不要把错误悄悄替换成默认值后再写“成功”。

默认值当然可以存在，但必须写在契约里，并与缺少必填字段区分。科研脚本尤应保留原参数和最终生效参数，避免报告中的配置与真正运行的设置不一致。

把配置文件当作实验条件

experiments.json是一组可以修改的教学实验条件。每次只改day02中的一个字段，保存后运行第2课，再比较输出。一次改多个条件会让你难以判断是哪项变化导致验证通过或失败。

修改前可以复制原配置到自己的笔记；修改后用python -m json.tool experiments.json检查语法。JSON使用双引号，布尔值与空值写true、false、null；不能直接把Python的True和None当成JSON值。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 先跑合法基线

为什么做：先确认环境和正常输入都没问题。

保留默认配置执行第2课。

```
python run.py day 2
```

预期结果：得到参数与验证结果，能够看到question和top_k的检查。

2. 打开结果与配置

为什么做：建立输入到输出的对应。

同时打开experiments.json和本课结果，定位day02。

```
python -m json.tool outputs/day02.json
```

预期结果：能指出哪些值来自配置，以及验证是否通过。

3. 只改变一种类型

为什么做：观察语法有效但契约不合格的情况。

将day02.top_k从数字3改成字符串"3"，保存后重跑。

```
python run.py day 2
```

预期结果：契约检查拒绝不匹配的类型，或返回明确失败；不得把字符串通过误认成模型理解。

4. 恢复并记录规则

为什么做：把一次错误变成自己的检查表。

恢复top_k为数字3；记录必填、类型、范围和额外字段四条约束，再次运行。

预期结果：正常输入恢复通过；笔记说明为什么合法JSON仍可能不符合工具要求。

改一改 · 同一个字段的四种取值

- 依次给day02.top_k设置3、"3"、null、0，每次只留一个值并运行第2课。

- 记录每次结果，分清语法错误与契约错误。

- 恢复合法整数，并保留一组通过/失败对照。

观察什么：类型或范围不符合契约的值会被拒绝；失败是本课主动制造的可观察结果。

机制解释：数据边界将含糊输入转成明确规则，使错误在执行昂贵工具之前暴露。

常见问题

JSONDecodeError

原因：引号、逗号或括号不成对。

处理：运行python -m json.tool experiments.json，根据行列提示修复；不要在JSON里加注释。

改了文件但结果没变

原因：编辑器未保存，或改错了日课配置。

处理：按Ctrl+S并确认修改位于day02，然后重跑；输出文件会被本次运行更新。

03 · 验收

第2课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day02.json

☐ 能解释3与"3"的类型区别。

☐ 完成至少一个合法值和两个错误值的对照。

☐ 恢复可用配置，记录失败发生在哪条约束。

自测与解析

1. 以下哪项仍需要额外验证?

- A. JSON能成功解析

- B. 文件扩展名是.json

- C. 对象具有某个字段

- D. 以上全部

答案：D。 解析、后缀与字段存在都不保证类型、范围和业务条件符合要求。

2. 实验时每次只改一个参数的主要作用是什么?

- A. 让文件更短

- B. 便于把输出变化归因于该参数

- C. 让模型更聪明

- D. 避免所有错误

答案：B。 控制其他条件，才能解释观察到的变化；它并不能保证程序没有其他问题。

小词典

- Schema：对字段、类型、范围及组合关系的说明。

- Validation：根据明确规则检查一个输入是否可接受。

- 边界值：接近允许范围端点、最容易暴露遗漏的输入。

04 · 我的记录

- 哪些错误应该在进入工具前被拒绝?

- 如果别人使用这份接口，是否能只看契约就正确传参?

忙碌日：20分钟

- 用3分钟读“JSON是交换格式，不是质量保证”，写下一个关键区别。

- 用12分钟运行 python run.py day 2，打开 outputs/day02.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 补一张边界输入表

- 写出空问题、缺字段、额外字段和布尔计数四类情况。

- 预测应接受还是拒绝，再对照代码验证。

- 给每个拒绝案例写一句对使用者有帮助的提示。

参考资料与阅读范围

Python 官方：JSON 编码与解码：只读基本读写、Python/JSON类型对应和格式错误说明。

https://docs.python.org/3/library/json.html

BioAgentLab：工具契约：对照输入、输出、失败与权限四种契约，不必通读全书。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/06_tool_contracts.md

下一步

下一课认识模型适配器，并明确离线基线与真实模型调用的区别。

第03天 · 接上模型，也保留可重复的基线

从接口和环境变量开始，认识真实工具调用的连接方式。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 解释模型适配器为何与工具执行分开。

- 运行离线基线并识别其模式。

- 知道如何选学连接Ollama或OpenAI，并记录未验证的部分。

开始之前

- 完成第2课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

同一个任务，可以有不同决策来源

模型适配器把程序内部请求转成服务商接受的消息格式，再把响应整理成统一结构。离线适配器使用固定规则，可以稳定复现；真实模型根据提示和上下文生成输出，可能提出工具调用，也可能直接回答。二者共享外围工具，但不代表二者的能力相同。

先留下离线基线，再接模型，能帮助定位失败：网络是否可达、模型是否存在、参数是否正确、工具是否拒绝、最终引用是否支持答案。不要把服务连接成功当成整套研究助手已经通过验证。

工具调用是一轮往返，不是执行字符串

真实模型可以提出工具名称和参数，程序随后检查并执行允许的工具，再把工具结果放回下一次模型请求。模型提出调用不等于已经运行工具；只有程序的执行记录才说明发生了什么。

工具结果也应被当成数据。模型不能因为收到一段文本就获得额外权限，程序更不应将它生成的任意字符串作为终端命令。后面的课程会进一步加入参数检查、最大步数、人工批准和来源验证。

Ollama选学：先安装、再检查、再调用

从Ollama官网按自己的系统安装并启动应用；Linux也可遵循官方安装说明。终端执行ollama --version和ollama list确认程序及已下载模型。若服务未运行，可另开终端执行ollama serve；服务已经启动时不需要重复开第二个。

一个可选示例是ollama pull qwen3:4b，然后再次ollama list。官方页面标有工具支持；它只是示例型号，能否流畅运行仍取决于内存、硬件、上下文与服务设置。模型下载和首次启动可单独安排，不计入每天的一小时核心任务。

把费用、凭据和验证范围写清楚

OpenAI路径使用本机环境变量保存API密钥，并选择自己账号可用的模型。请按实践包README设置，不要把密钥放到网页笔记或提交到GitHub。带--allow-network的命令表示你主动选择允许真实服务请求。

课程制作过程中没有替你下载运行大模型，也没有调用你的付费API。连接代码的协议测试与真实推理测试应分别记录。真实调用失败时应明确报错，不应偷偷换回离线结果后继续标成真实模型。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行不需要账号的基线

为什么做：把环境、模式和结果先检查清楚。

先用默认offline运行；打开结果中的mode。

```
python run.py day 3
```

预期结果：得到明确的离线教学输出，无需模型下载或密钥。

2. 查看统一问答入口

为什么做：确认任务可以通过独立命令调用。

用下面的英文教学问题运行ask，并阅读实践包README中的provider说明。

```
python run.py ask "What is a tool contract?"
```

预期结果：离线回答与证据或执行记录能够检查；不声称来自真实LLM。

3. 选学准备本地模型

为什么做：把安装与推理分成可定位的步骤。

安装并启动Ollama后执行下面命令；下载时间另外安排。若本机不适合，继续完成离线主线即可。

```
ollama --version
ollama list
ollama pull qwen3:4b
ollama list
```

预期结果：列表出现qwen3:4b；失败时先处理安装、网络或资源问题。

4. 选学进行一次真实请求

为什么做：观察模型提出工具调用与程序执行之间的区别。

在practice目录运行下列命令；保存完整状态，检查mode与工具轨迹。

```
python run.py ask "What is a tool contract?" --provider ollama --model qwen3:4b --allow-network
```

预期结果：服务可用时得到真实请求结果；输出可能与离线不同。若报错，保留错误并按README排查，不伪造成功。

改一改 · 比较任务变化，明确决策来源

- 先执行python run.py ask "What is a tool contract?"并保存结果。

- 再执行python run.py ask "Can correlation prove causation?"，比较命中的资料与回答。

- 若已连接真实模型，再任选同一问题比较provider；未连接时在记录中写“真实模型尚未执行”。

观察什么：问题变化影响检索或回答；离线规则与真实模型仍有明确模式标识。

机制解释：保持任务与工具边界一致，才有条件比较决策来源；连接成功并不等于答案可信。

常见问题

Ollama连接被拒绝

原因：本地服务未启动或服务地址不匹配。

处理：先确认Ollama应用运行；按README检查地址，必要时另开终端运行ollama serve。

提示模型不存在或内存不足

原因：请求的模型未下载，或资源不能支持当前配置。

处理：用ollama list核对名称；选本机能够运行的工具模型并降低资源负担，或继续离线主线。

提示缺少网络许可或API密钥

原因：真实连接需要显式许可和正确环境变量。

处理：只在愿意发起真实请求时使用--allow-network；API设置参照README，密钥保留在本机。

03 · 验收

第3课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day03.json

☐ 完成一次离线日课和一次ask调用。

☐ 能描述“模型提出调用→程序检查执行→结果送回模型”的顺序。

☐ 如实记录真实模型已执行或尚未执行，不把连接代码当成推理证据。

自测与解析

1. 模型返回一个工具调用，下一步应是什么？

- A. 直接执行任何生成的代码

- B. 程序校验工具名和参数，再执行允许的工具

- C. 把它当作已经完成的计算

- D. 自动扩大权限直到成功

答案：B。 工具调用是请求，执行权属于外围程序；参数、权限和预算均需要检查。

2. 尚未运行真实模型时，哪种记录准确？

- A. 已完成真实LLM能力验证

- B. 所有模型都能调用本工具

- C. 离线基线已运行，真实模型接入尚未实测

- D. 模型连接必然免费且无资源要求

答案：C。 把完成范围与未执行部分分别登记，便于下次继续验证，也避免夸大作品能力。

小词典

- Provider：提供模型决策或生成服务的实现。

- 适配器：把内部请求与外部服务格式互相转换的组件。

- Tool call：模型提出的工具名称及结构化参数请求。

04 · 我的记录

- 这次失败若发生在服务连接层，我应该先检查什么？

- 哪些信息可以公开，哪些应只留在本机环境变量？

忙碌日：20分钟

- 用3分钟读“同一个任务，可以有不同决策来源”，写下一个关键区别。

- 用12分钟运行 python run.py day 3，打开 outputs/day03.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 做一张模型接入记录卡

- 记录provider、模型全名、日期、输入问题和是否触发工具。

- 记录实际延迟、失败情况与尚未验证的能力。

- 以同一个问题比较两次结果，区分格式一致与内容可靠。

参考资料与阅读范围

Ollama 官方：Tool calling：只看单次工具调用及把工具结果送回模型的消息顺序。

https://docs.ollama.com/capabilities/tool-calling

Ollama 模型页：qwen3:4b：核对模型名称与工具支持标签；下载大小不等于完整运行内存需求。

https://ollama.com/library/qwen3:4b

OpenAI 官方：Function calling：阅读工具定义、模型提出调用、程序执行及返回结果的流程。

https://platform.openai.com/docs/guides/function-calling

下一步

下一课检查输出结构：能生成JSON，并不等于结果可以直接采用。

第04天 · 把模型输出变成可检查的结果

验证字段、引用和不确定性，而不是只看答案顺不顺。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 区分输出格式检查与事实核验。

- 定位候选结果中的必填字段。

- 主动删除字段，观察验证失败。

开始之前

- 完成第3课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

先定义要收什么结果

科研助手的回答应至少把结论、依据和局限分开。没有来源的漂亮段落，往往难以复查；只含引用编号、没有对应主张的结果，也可能看似严谨却无法核验。结构化输出让后续程序知道每部分该在哪里。

本课生成一个候选结构并执行检查，帮助你观察字段契约。它主要演示外围验证，不宣称模型已经自动理解科学事实，也不宣称失败后有一个已经实现的自动修复器。

结构正确与证据支持是两道检查

格式检查关心字段存在、类型合适、列表和字符串符合约定。证据检查则要问：引用是否存在，摘录是否来自该来源，摘录是否真的支持对应结论。这两类检查可能给出不同答案。

一个JSON可以完全合法，却把关联写成因果；一条引用可以确实存在，却与回答的问题无关。因此本课的通过状态只能按实际检查项解释，不要把“check通过”写成“事实已验证”。

不确定性需要成为结果的一部分

保留uncertainty之类的字段，是要求系统说明哪些部分没有得到材料支持。它可以包含样本、研究设计、适用范围或待复核事项。没有发现错误，不等于已经穷尽所有可能的错误。

人工填写不确定性也有价值。最开始可以写“只有教学材料，未检索完整文献”“没有验证外部数据”；随着系统能力增长，再把这些说明细化成证据缺口和下一步核验任务。

把失败当作教学观察

如果删除一个必填字段后检查失败，说明某条约束确实在运行；若结果仍通过，就需要检查自己删错位置、配置没有保存，或者验证器确实漏了这条规则。用刻意制造的坏例子检验程序，比只运行正常样例更有说服力。

本课每次修改后会重新生成候选结果。请保留失败前后的JSON或把关键差异记到笔记，不要只保留最后一次通过。这样后面重构代码时，也能重新使用这些边界案例。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行原始候选结果

为什么做：建立通过情况的基线。

使用默认day04配置运行。

```
python run.py day 4
```

预期结果：输出包含候选内容与检查结果。

2. 逐项对照字段

为什么做：知道验证器承诺检查哪些内容。

打开输出的candidate与check，列出结论、引用、不确定性等实际字段。

```
python -m json.tool outputs/day04.json
```

预期结果：你能指出一个字段的类型要求以及一个当前未覆盖的事实问题。

3. 删除一项必填内容

为什么做：证明错误输入会触发检查。

将experiments.json的day04.remove_field设为uncertainty，保存后重跑。

```
python run.py day 4
```

预期结果：检查明确指出缺失字段；候选内容不能直接当成通过结果使用。

4. 恢复并解释边界

为什么做：不把验证器夸大成全自动科学审查。

恢复remove_field的默认值；在笔记区分别写“已检查”和“仍需人工复核”。

预期结果：保留一份成功/失败对照，明确本课没有自动模型修复。

改一改 · 引用字段和不确定性字段都能删吗?

- 先用默认配置运行并记录check。

- 分别将remove_field设为uncertainty和citations，每次重新运行第4课。

- 比较错误提示；恢复默认值后再运行，确认正常状态恢复。

观察什么：删除被要求的字段会导致验证失败，而不是得到一个看似完整的成功答案。

机制解释：机器能可靠检查明确写出的规则；没有写出的科学含义不能凭结构验证自动得到保证。

常见问题

删除后仍显示旧结果

原因：没有保存配置或查看了旧文件。

处理：确认day04.remove_field已保存，并重跑后重新打开outputs/day04.json。

误以为程序会自动修复

原因：把验证失败和模型重新生成混为一谈。

处理：本课只观察验证；先人工修正配置，后续再设计有限次数的修复流程。

03 • 验收

第4课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day04.json

☐ 能定位candidate与check。

☐ 完成两种缺字段实验并恢复正常配置。

☐ 能举例说明结构通过但科学结论仍可能不成立。

自测与解析

1. 结构检查通过，是否意味着引用支持结论?

- A. 是，所有字段存在就够了

- B. 否，还要核查来源内容与主张的对应关系

- C. 是，只要有三个引用

- D. 否，因此结构检查没有价值

答案：B。 结构检查有助于后续处理，但不能替代语义证据核验。

2. 为什么要保留uncertainty字段?

- A. 为了让答案更长

- B. 让缺少的证据与适用边界可见

- C. 让所有结论自动变正确

- D. 替代原始来源

答案：B。 明确已知局限能帮助读者决定是否继续验证；它不是免责口号或真实性保证。

小词典

- Candidate：尚待检查的候选输出。

- 结构验证：检查字段、类型、范围和组合约定。

- 证据核验：检查来源内容是否支持对应主张。

04 · 我的记录

- 当前验证器可以检查什么，不能检查什么？

- 我是否保存了足以复现失败的输入？

忙碌日：20分钟

- 用3分钟读“先定义要收什么结果”，写下一个关键区别。

- 用12分钟运行 python run.py day 4，打开 outputs/day04.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 设计一个格式正确但事实错误的例子

- 保留全部字段与合法JSON。

- 把一个观察性关联改写为因果结论。

- 说明结构检查为何可能通过，以及需要增加哪种人工复核。

参考资料与阅读范围

Python 官方：JSON 编码与解码：只读基本读写、Python/JSON类型对应和格式错误说明。

https://docs.python.org/3/library/json.html

BioAgentLab：工具契约：对照输入、输出、失败与权限四种契约，不必通读全书。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/06_tool_contracts.md

下一步

下一课将工具定义成明确的接口，并在执行前检查参数。

第05天 · 给 Agent 配一组可信的工具

把能做什么、怎么调用和失败后怎么解释写清楚。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 阅读一个工具的名称、说明、参数和输出。

- 通过错误top_k观察工具边界。

- 理解工具白名单与最小权限。

开始之前

- 完成第4课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

工具是一个受约束的函数入口

工具将某种动作包装成程序可以执行的接口，例如检索教学资料、读取证据记录或分析指定CSV。一个合格接口应交代输入参数、返回结构、失败状态和允许触及的数据范围。描述过于含糊，会让调用方很难选择正确工具。

从你的科研场景出发，未来可以把CLEAR或MEF等已有流程包装成工具，但包装不会替这些方法补上统计前提。当前课程只使用教学材料与合成数据，不会宣称已经连接或验证那些专业研究流程。

工具说明应帮助正确选择

名称应说明动作，说明文字应写明适合和不适合的任务。search_evidence可用于寻找给定资料中的相关记录；它并不等于搜索全互联网，也不意味着返回内容已经经过全面事实核验。

参数应尽量少而明确。query说明找什么，top_k限制最多返回多少条。路径、字段和算法选项若可任意传入，就会增加误用空间。第一版先支持少量可解释的操作，再根据确实存在的需求扩充。

白名单让执行权留在程序一侧

程序只允许注册过的工具名称，并验证参数后才执行。模型即使提出不存在的工具、额外字段或超范围计数，也应得到清楚的拒绝。不要用eval或任意终端执行把模型文本直接变成权限。

白名单只是其中一层约束。被允许的工具本身仍可能设计有问题，返回的材料也可能包含误导内容。后面会继续学习不可信数据、人工批准和评价，不能把一份白名单理解成安全问题已经全部解决。

失败结果也应有稳定结构

工具失败时应告诉调用者发生在哪一层：参数不符、资料缺失、预算耗尽还是运行错误。将一切失败都转成空列表，会让后续流程误把故障解释为“没有证据”。

本课比较合法调用和主动制造的错误调用。你应能够在输出中找到成功结果与拒绝原因，并说明为什么错误调用没有继续访问工具。保持错误可解释，是后面重试与恢复的基础。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行工具案例

为什么做：观察工具注册与实际执行。

先保留默认配置。

```
python run.py day 5
```

预期结果：看到允许的工具、合法调用结果以及参数边界演示。

2. 读懂一次调用的四个部分

为什么做：把自然语言需求与函数接口对应起来。

打开输出，定位工具名、参数、返回值和失败信息；同时阅读实践代码中的工具定义。

```
python -m json.tool outputs/day05.json
```

预期结果：能解释top_k控制结果数量，不代表证据质量。

3. 改变错误参数

为什么做：看拒绝是否真正发生。

在experiments.json中找到day05.invalid_top_k，设置一个明显超范围的数再运行。

```
python run.py day 5
```

预期结果：非法调用被拒绝，合法调用仍可完成。

4. 写一张科研工具卡

为什么做：将接口方法迁移到熟悉工作。

选择“检查样本表缺失”这样的有限操作，写输入、输出、失败和权限四段说明；本课只写设计，不调用真实研究数据。

预期结果：一张别人能够理解的工具卡，并明确哪些资料可访问。

改一改 · 结果数量与合法范围

- 先记录默认day05.invalid_top_k产生的错误。

- 将它改成0，再改成一个明显过大的数，各运行一次第5课。

- 恢复默认值，写下合法检索与错误调用的不同执行路径。

观察什么：超出契约范围的参数收到明确拒绝，不会因为模型提出请求就直接执行。

机制解释：校验发生在执行之前，使工具权限与资源限制成为代码规则。

常见问题

工具名称不存在

原因：调用方使用了未注册名称。

处理：查看工具列表并使用精确名称，不通过模糊匹配自动选一个危险的相似动作。

输出为空但不知是否失败

原因：把成功的空结果与执行异常混淆。

处理：先查看状态与错误字段，再判断是否真的没有命中材料。

03 · 验收

第5课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day05.json

☐ 能描述一个工具的输入、输出、错误与权限。

☐ 至少运行两种不合法top_k的对照。

☐ 能说明白名单没有自动解决内容可信和科学方法有效性。

自测与解析

1. 模型请求一个不在白名单中的工具，应怎样处理?

- A. 自动安装并运行它

- B. 拒绝并返回明确原因

- C. 把名字当终端命令

- D. 换成任意相近工具

答案：B。 允许执行的动作由程序确定，模型文本不能自动扩大权限。

2. 将专业分析流程包装成工具，能证明什么?

- A. 方法前提自动满足

- B. 研究结论已被独立验证

- C. 获得了可调用入口，仍需核对数据和方法前提

- D. 无需记录参数

答案：C。 包装解决接口问题，科学有效性仍依赖数据、设计、方法和验证。

小词典

- 工具契约：输入、输出、错误状态和权限的共同约定。

- 白名单：程序明确允许调用的有限工具集合。

- 最小权限：只给予当前任务确实需要的动作与数据访问范围。

04 · 我的记录

- 哪一个工具参数最容易被误解?

- 如果工具返回空结果，我如何判断它是正常情况还是故障?

忙碌日：20分钟

- 用3分钟读“工具是一个受约束的函数入口”，写下一个关键区别。

- 用12分钟运行 python run.py day 5，打开 outputs/day05.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 给一个工具补负例

- 写出一个应该使用该工具的任务。

- 再写两个不应该交给它的任务。

- 将这三个例子加入工具说明，帮助调用者理解边界。

参考资料与阅读范围

BioAgentLab：工具契约：对照输入、输出、失败与权限四种契约，不必通读全书。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/06_tool_contracts.md

OpenAI 官方：Function calling：阅读工具定义、模型提出调用、程序执行及返回结果的流程。

https://platform.openai.com/docs/guides/function-calling

下一步

下一课把工具连接成能够观察结果并按预算停止的循环。

第06天 · 做一个会停下来的行动循环

把行动、观察与结束条件连接起来。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 读懂一次检索与回答的执行轨迹。

- 理解最大步数与终止状态。

- 用较小预算观察流程如何提前停止。

开始之前

- 完成第5课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

循环的价值，在于根据观察决定下一步

一个行动循环接收问题，决定一个动作，执行工具，再读取结果。真实模型模式可以让模型参与选择动作；离线主线则用规则安排同样的接口，以便稳定观察每一阶段。理解两种决策来源的差别，比给流程起一个复杂名称更重要。

工具返回内容后，下一步不一定是继续检索。如果已有足够材料，可以生成带来源结果；如果没有支持证据，可以停止并说明不足；如果调用失败，应进入错误分支。不断重复相同动作不等于持续取得进展。

结束条件应该写进程序

至少需要三种结束方式：完成任务、证据不足、预算耗尽。还可以记录不可恢复错误或用户取消。输出中给出明确状态，使用者才能判断这次结果是完整交付还是中间状态。

最大步数限制的是当前实现定义的一次循环迭代，不自动等于模型token数、秒数或工具次数。阅读trace，找出每一步实际发生的动作，再解释预算的含义。未来加入真实模型时，还需要独立考虑请求次数与费用。

轨迹记录应该可检查，但不冒充内部思维

本课trace记录可观察的事件，例如选择了哪个工具、参数是什么、返回什么状态以及何时停止。这样的记录帮助你定位故障与复现行为，属于程序审计信息。

不要把程序生成的解释字段当成模型完整的内部推理，也不要因为有一长串轨迹就认为回答更加正确。轨迹提供行为证据，答案仍须与材料逐项核对。

先比较完整路径，再制造预算不足

第一次使用默认max_steps运行并保存结果。第二次只降低步数，看程序是否出现清晰的停止状态；第三次恢复预算，检查正常路径是否回来。这样可以验证限制真正参与运行，而非只写在说明文档。

一个好的停止结果应告诉你已经完成什么、缺什么以及是否适合重试。对于科研助手，部分完成的材料可以保留为中间产物，但不能直接改名为最终研究结论。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行完整基线

为什么做：确定默认预算下实际执行的阶段。

保留day06默认配置运行。

```
python run.py day 6
```

预期结果：得到answer对象与可检查的trace和状态。

2. 按顺序读轨迹

为什么做：把抽象循环对应到真实事件。

从第一个事件读到结束，记录工具调用与回答在哪一步。

```
python -m json.tool outputs/day06.json
```

预期结果：能够画出本次真实路径，而不是只复述课件示意图。

3. 降低最大步数

为什么做：验证预算能中断流程。

将day06.max_steps设为1，保存并重跑。

```
python run.py day 6
```

预期结果：观察预算限制下的状态与已发生动作；不把部分输出视为完整交付。

4. 恢复并比较

为什么做：证明变化来自步数预算。

恢复max_steps为6，再运行；在笔记列出两次状态、工具动作和引用的差别。

预期结果：保留正常与提前停止两条轨迹，能解释停止条件的作用。

改一改 · 一轮预算能走到哪里?

- 分别以day06.max_steps=1和6运行第6课。

- 比较result.answer的status、trace与citations。

- 只根据轨迹中已发生的事件描述结果，不推断尚未执行的模型能力。

观察什么：较小预算可能在形成完整答案前停止；恢复预算后按主线完成。

机制解释：停止条件是系统行为的一部分；可靠流程既要会执行，也要在条件不足时有边界地结束。

常见问题

预算太小，没有完整答案

原因：流程在生成最终回答前达到限制。

处理：查看trace确认已走到哪步，再恢复允许的预算；不要直接取消全部上限。

有轨迹就认为结果可信

原因：把行为记录与证据核验混为一谈。

处理：逐条检查引用内容是否支持主张，保留尚未确认的部分。

03 · 验收

第6课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day06.json

☐ 能说明一次行动与一次观察各是什么。

☐ 完成max_steps的对照实验。

☐ 把完成、证据不足、预算耗尽与错误分开记录。

自测与解析

1. 达到最大步数后应怎么办？

- A. 无限继续直到满意

- B. 返回明确停止状态并保留已完成信息

- C. 把中间结果自动标为最终成功

- D. 忽略预算字段

答案：B。 预算限制需要真正控制执行，并让使用者看到这次任务完成到什么程度。

2. trace最直接证明哪件事？

- A. 模型完整内部推理

- B. 这次程序记录了哪些可观察动作

- C. 所有引用均支持结论

- D. 未来同类问题永远正确

答案：B。 事件轨迹是行为记录；内部推理、内容真实性与未来泛化需要不同证据。

小词典

- 行动循环：提出动作、执行、接收观察并决定继续或停止的循环。

- 终止状态：告诉使用者任务为何结束的明确标记。

- Trace：可观察步骤与工具事件的记录。

04 · 我的记录

- 如果下一步继续花费资源，怎样判断它有可能带来新信息？

- 预算耗尽后，我愿意交付哪些中间产物？

忙碌日：20分钟

- 用3分钟读“循环的价值，在于根据观察决定下一步”，写下一个关键区别。

- 用12分钟运行 python run.py day 6，打开 outputs/day06.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 给循环增加一个停机理由的设计

- 构造“连续两次得到相同证据”的情况。

- 在纸面设计无进展检测规则。

- 说明何时停止、何时允许人工批准增加预算。

参考资料与阅读范围

Anthropic：Building effective agents：阅读 workflows 与 agents 的区别，以及从简单结构开始的讨论。

https://www.anthropic.com/research/building-effective-agents

BioAgentLab：规划与 ReAct：关注行动、观察、停止条件与规划依赖，不把示意流程当成可靠性证明。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/07_planning_and_react.md

下一步

下一课把第一周的接口、工具与验证整合成一张证据卡。

第07天 · 第一周作品：一张可复核的证据卡

把环境、契约、工具与检查连接成小作品。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 按证据ID整理一张可复核的卡片。

- 区分原文摘录、自己的主张和局限。

- 知道卡片模板与完整论文提取的差别。

开始之前

- 完成第6课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

阶段作品要小到可以认真核对

第一周作品是一张证据卡：你选择一条教学记录，写出它支持的主张，保留来源与摘录，并登记局限。作品的价值在于别人能够沿着卡片回到材料，检查你的解释是否越界。

这与自动总结整篇论文不同。当前程序读取实践包内准备好的教学记录，部分字段是模板或提示，并不代表系统已经阅读全文、提取全部统计数据或独立评价研究质量。

主张、摘录与来源分别承担不同作用

claim是你要表达的判断；本课quote是实践包原创教学记录的文字，source_kind会明确材料性质；source_url提供进一步阅读入口，不是声称quote逐字出自该外部论文。引用了一篇真实文章，不代表文章支持任何你写出的句子。

以“关联不等于因果”为例，卡片可以记录教学说明中的对应文字，再解释为什么观察到两个变量同变还需要考虑混杂、反向因果和研究设计。原文摘录应保持原意，你自己的推断要另写。

局限字段应写具体缺口

局限可以写“这是教学概念说明，没有报告真实受试者结果”“没有核对完整论文的方法与补充资料”“结论适用于所述研究条件”。比泛泛写“仍需进一步研究”更有用。

requires_human_review是一个工作状态，提醒有人继续核验。它不代表程序已经知道所有风险，也不能替代实际审阅。只有完成了什么检查，才在学习笔记中写什么证据。

把已收藏资料变成可读的任务

你的BioAgentLab提供工具契约和证据整理思路；GitHub收藏中的CLEAR-Microbiome与SCUT_Minilab_MEF可以作为未来的真实研究工具案例。当前一小时先学会证据卡结构，避免同时安装几个大型分析流程而无法完成核验。

以后把自己的公开论文加入材料时，要重新登记来源、阅读日期、研究对象和评价范围。不要把课程中的合成或概念数据复制到真实研究结果表里，也不要让同一张卡片承担超出原材料的结论。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 选择第一条证据

为什么做：先完成一张可以核对的小卡片。

保持day07.evidence_id为E01，查看data中的对应教学记录。

```
python run.py day 7
```

预期结果：输出result.evidence_card，包含ID、主张、来源、摘录和局限。

2. 逐项核对来源

为什么做：确认卡片不是只有好看的格式。

打开结果，对照原始教学记录，检查quote来自哪里。

```
python -m json.tool outputs/day07.json
```

预期结果：能区分原始材料文字与自己填写的claim。

3. 修改主张但不改摘录

为什么做：体验证据不随措辞自动改变。

将day07.claim改成“观察到关联仍不足以证明因果”，保存后运行。

```
python run.py day 7
```

预期结果：主张变化，所选证据仍由evidence_id确定；需要人工判断是否支持。

4. 完成一页人工复核

为什么做：把检查结果变成可交付物。

写下三项：主张是否忠于材料、局限是什么、还缺哪一步验证；保留待复核状态，直到自己完成检查。

预期结果：得到一张有来源、有边界、有复核记录的证据卡。

改一改 · 同一证据，能支持相反主张吗?

- 先用E01与忠于材料的claim生成卡片。

- 仅将claim改成“只要相关就能证明因果”，再运行第7课。

- 观察程序是否仍生成卡片；在笔记解释为什么“生成成功”不能证明主张正确。最后恢复合理主张。

观察什么：模板可以容纳不被来源支持的主张；你需要通过原文对照发现并纠正它。

机制解释：卡片组织信息，科学判断需要核对主张与证据之间的关系；字段完整不能替代审阅。

常见问题

找不到证据ID

原因：ID输入错误或不在当前教学库。

处理：打开data/evidence.json核对现有ID，恢复E01后再次运行。

把卡片当成完整论文分析

原因：当前程序读取预先准备的教学记录。

处理：在交付说明标注材料范围；若要分析论文，应另行阅读全文并登记研究设计与数据。

03 · 验收

第7课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day07.json

☐ 证据卡具有可定位的ID与来源。

☐ 人工比较过quote与claim，完成一次不支持主张的负例。

☐ 局限和待复核事项具体，不把教学说明写成真实实验结果。

自测与解析

1. 生成了一张字段完整的证据卡，能否证明claim成立？

- A. 能，字段完整就是充分证据

- B. 不能，还需核对来源是否支持该主张

- C. 能，只要URL真实

- D. 不能，因此不需要整理卡片

答案：B。 卡片使审阅成为可能，不能代替审阅本身。

2. 哪一条局限最有用？

- A. 可能有局限

- B. 未来还有机会

- C. 本卡来自教学概念记录，未核对真实队列数据与完整论文方法

- D. 助手很强但偶尔出错

答案：C。 具体说明材料范围与未完成检查，读者才能判断还需要什么工作。

小词典

- 证据卡：把主张、来源、摘录、局限和复核状态放在一起的记录。

- Claim：需要证据支持的明确主张。

- 人工复核：由人对实际来源与结果进行检查，而不是只点击完成。

04 · 我的记录

- 哪一个字段最容易让读者误以为程序做了更多工作?

- 如果把这张卡交给同学，对方能否复现我的核对过程?

忙碌日：20分钟

- 用3分钟读“阶段作品要小到可以认真核对”，写下一个关键区别。

- 用12分钟运行 python run.py day 7，打开 outputs/day07.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 将卡片模板迁移到公开论文

- 挑选一篇已公开且能读到全文的论文。

- 人工补研究对象、设计、主要结果和局限，不填写猜测数字。

- 保留网页链接与阅读日期，注明哪些内容尚未读完。

参考资料与阅读范围

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

BioAgentLab：工具契约：对照输入、输出、失败与权限四种契约，不必通读全书。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/06_tool_contracts.md

下一步

第二周从材料导入开始，让每份证据拥有稳定的来源与版本。

第08天 · 给材料建立来源与版本档案

从导入记录开始，让每次实验能够追溯材料变化。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 识别导入记录的文字与元数据。

- 观察内容摘要随记录变化。

- 区分本次语料副本与永久修改原资料库。

开始之前

- 完成第7课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

先登记材料，再讨论检索效果

没有稳定来源，回答很难复查。导入一条材料时，至少应知道它的标题、内容、标识与来源范围；有条件时还要记录作者、日期、许可和读取时间。材料越杂，越需要明确哪些字段来自原文，哪些是自己的注释。

本课导入的是一个自写教学记录，并追加到本次运行的语料副本。它不会自动改写data/evidence.json，也没有实现通用PDF解析或全网抓取。先把一个可控例子做对，再考虑复杂格式。

摘要用于识别版本，不负责评价真假

程序为记录计算SHA-256摘要。相同规范化输入通常得到相同摘要，输入改变时摘要也会改变，因此可以用来检查本次实验是否仍使用同一版材料。摘要长度固定，不能从摘要还原全部原文。

摘要不告诉你材料是否真实、是否来自权威来源或是否符合研究伦理。它解决“是否发生变化”的识别问题；来源质量、内容准确和许可仍需另外检查。

内容与元数据都可能影响实验

只改文字会影响检索内容，只改标题或日期也可能改变材料解释与版本标记。当前实现对整条imported_record做稳定JSON序列化，再计算SHA-256；键按名称排序、使用紧凑分隔符与UTF-8编码。因此只改title或只改text都会改变摘要。你将分别修改并观察，而不是只凭字段名字猜测。

科研工程中还需要把材料版本与配置、代码版本一起记录。只保存一个最终回答，往往无法知道后来不同结果究竟来自模型变化、材料变化还是参数变化。

用本次副本练习，减少不可逆操作

本课运行时构建语料副本，输出导入记录与前后条数，便于检查新增是否发生。重跑相同命令不意味着原始资料文件被永久扩充；你应查看records_before和records_after，并核对data文件是否改变。

以后实现真正持久导入时，还要考虑重复ID、重复内容、版本覆盖、错误回滚和删除。不要在这一课混入未经整理的个人或临床资料；用自写的短说明就足以观察机制。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行一次教学导入

为什么做：认识导入结果与语料副本。

使用默认record_title和record_text运行。

```
python run.py day 8
```

预期结果：得到imported_record、content_sha256和前后记录数。

2. 核对输入与新增记录

为什么做：证明输出确实来自当前配置。

打开结果，核对标题、文字与记录数差异。

```
python -m json.tool outputs/day08.json
```

预期结果：本次副本包含新增记录，原始资料文件并未被永久改写。

3. 只修改文字

为什么做：观察版本标记随材料变化。

把day08.record_text末尾加上“Review the original source before reuse.”，保存并重跑。

```
python run.py day 8
```

预期结果：导入文字更新；比较content_sha256是否改变。

4. 保留版本对照

为什么做：让后续检索实验知道自己读的是哪版资料。

将两次标题、文本改动、摘要和日期记录到笔记；恢复或明确保留修改后的配置。

预期结果：留下可解释的材料版本差异，而不是只记录“运行成功”。

改一改 · 标题和正文，分别会带来什么变化?

- 先运行默认第8课并保存摘要。

- 只改day08.record_title，运行并记录变化。

- 恢复标题，只改record_text再运行；结合源代码说明哪些字段参与当前摘要。

观察什么：只改变标题或只改变正文，都会改变content_sha256；当前摘要覆盖整条导入记录。

机制解释：版本追踪需要明确摘要覆盖的输入范围，不能把一个哈希当成来源可信度评分。

常见问题

重跑后原资料没有变多

原因：误把内存副本导入当成永久保存。

处理：查看records_before/records_after与输出记录；本课不修改data/evidence.json。

认为哈希相同就能证明内容可信

原因：混淆版本一致和事实准确。

处理：分别记录一致性检查与人工来源审查，两者不能互相替代。

03 · 验收

第8课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day08.json

☐ 能指出导入记录的标题、文字与摘要。

☐ 完成标题/正文的单变量对照。

☐ 能说明导入只影响本次语料副本，未自动改写原资料库。

自测与解析

1. SHA-256摘要最适合帮助检查什么？

- A. 这段文字是否属于科学共识

- B. 当前输入是否与已记录版本一致

- C. 作者是否有足够资历

- D. 研究结论是否因果成立

答案：B。 摘要可识别输入版本，不能判断内容的科学质量。

2. 本课导入新记录后，原始data/evidence.json会怎样？

- A. 自动永久追加

- B. 自动删除旧版本

- C. 保持原文件，本次运行使用新增后的副本

- D. 自动上传到网络

答案：C。 课程把导入范围限定在本次语料副本，避免把教学步骤与持久数据管理混为一谈。

小词典

- 元数据：描述材料来源、标题、时间与用途等信息的字段。

- 摘要/Hash：由输入计算的固定长度版本标记。

- Provenance：描述数据从何而来、经过何种处理的来源记录。

04 · 我的记录

- 如果两次回答不同，我是否能检查它们使用的材料版本?

- 哪些来源字段在我的科研工作中不可省略?

忙碌日：20分钟

- 用3分钟读“先登记材料，再讨论检索效果”，写下一个关键区别。

- 用12分钟运行 python run.py day 8，打开 outputs/day08.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 设计一张材料登记表

- 列出ID、来源URL、版本、读取日期与许可。

- 增加“原文”和“人工注释”的区别。

- 写出重复导入时应新增版本还是拒绝的规则。

参考资料与阅读范围

Python 官方：hashlib：查看sha256的输入与hexdigest输出；理解摘要用于识别内容版本。

https://docs.python.org/3/library/hashlib.html

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

下一步

下一课把材料切成块，建立可以解释的词项检索基线。

第09天 · 做一个可以解释的检索基线

比较分块、重叠与词项排序，不急着追逐复杂框架。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 理解分块如何影响证据单元。

- 调整chunk_size和top_k并观察结果。

- 区分词项检索与向量检索。

开始之前

- 完成第8课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

分块决定检索返回什么大小的材料

整篇材料可能太长，一句话又可能缺少上下文。分块将材料切成可以检索和引用的片段。块太小容易拆开限定条件；块太大可能混入很多无关内容。选择块大小应根据材料结构和问题类型做对照。

本课以字符长度与重叠数量演示分块，便于看见参数如何影响片段。它不是按模型token精确切分，也不自动理解句子、表格或方法段落的语义边界。

重叠有作用，也有代价

相邻片段保留一些重叠文字，可以减少边界附近信息被切断的机会。但重叠太大，会带来重复结果、更多存储与处理开销；也可能让评价表里看起来找到很多证据，实际上来自同一段文字。

因此除了看命中数量，还要读结果是否互相重复、是否包含完整限定条件，以及能否回到原始记录。片段ID与原记录ID都值得保留，便于合并和追踪。

先理解一个词项基线

本课按词项匹配给片段排序，查询词与材料中的词有重合时容易命中。优点是过程直观、容易调试，缺点是同义词、跨语言和语义改写可能漏检。它不是已经接入embedding模型的向量数据库。

向量检索可以作为后续扩展，但也需要独立评价。不要因为换成更复杂的表示，就假定召回和答案正确性一定提高。一个解释清楚的简单基线，是比较改进是否有效的起点。

top_k控制返回数量，不自动控制质量

top_k决定最多保留多少个高分结果。较小值可能漏掉必要资料；较大值可能引入无关或冲突材料。排序靠前只能说明当前评分较高，不意味着该片段具有更高科学可信度。

本课每次改一个条件，记录块数、命中片段、重复情况和是否包含完整依据。先看内容，再看分数；后面的评价课会用预先标注的问题检查检索是否找到了该找的证据。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行分块与检索

为什么做：取得可以复查的基线。

保留默认query、chunk_size、overlap和top_k。

```
python run.py day 9
```

预期结果：得到chunks和retrieved，能够看见原记录与片段内容。

2. 手工检查一个片段

为什么做：确认分块没有神奇理解语义。

打开结果，选一个片段找出其原记录，查看开头与结尾是否切断关键限定条件。

```
python -m json.tool outputs/day09.json
```

预期结果：能说明一个片段为什么有用或为什么需要上下文。

3. 改变块大小

为什么做：观察片段数量与信息完整性的变化。

保持query、overlap和top_k不变，将day09.chunk_size从80改为120并重跑。

```
python run.py day 9
```

预期结果：块数和片段范围随长度变化；不要求分数一定提高。

4. 恢复并比较返回数量

为什么做：把检索参数与可见结果对应。

恢复块大小，再分别用top_k=1和3运行，记录多出的片段是否有帮助。

预期结果：留下参数、结果和人工判断三列对照。

改一改 · 同义改写能被词项基线找到吗?

- 先用默认day09.query运行并保存retrieved。

- 将query改成含同义表达或中文的问题，保留其他条件再运行。

- 比较词项命中，说明遗漏是否可能来自语言形式；恢复能复现的查询。

观察什么：词面变化可能改变排序甚至导致漏检；这不表示原材料不存在。

机制解释：检索表示有自己的能力边界，理解漏检原因比只看一个分数更能指导改进。

常见问题

重叠参数导致分块异常

原因：overlap接近或超过chunk_size。

处理：保持0≤overlap<chunk_size；恢复默认80/16后再逐项修改。

问题明显相关却找不到

原因：查询与材料没有足够词项重合。

处理：先用材料中的关键词验证基线，再记录同义词/跨语言的局限，而不是假称语料没有证据。

03 · 验收

第9课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day09.json

☐ 能指出字符分块与token分块的差别。

☐ 完成块大小或top_k的真实参数对照。

☐ 能给出一个词项检索可能漏检的同义改写例子。

自测与解析

1. top_k从1增加到3，最直接改变什么?

- A. 材料的真实性

- B. 允许返回的最大结果数量

- C. 模型参数数量

- D. 研究设计质量

答案：B。 top_k改变选择数量，是否更好仍要检查新增材料的相关性和支持作用。

2. 本课的chunk_size按什么计算?

- A. 模型token

- B. 研究样本数

- C. 字符长度

- D. 嵌入向量维度

答案：C。 字符切分便于观察机制，但与模型token计数或语义段落并不等价。

小词典

- Chunk：从原始材料切出的可检索片段。

- Overlap：相邻片段共享的文字范围。

- 词项检索：根据查询和材料中的词项匹配进行选择与排序。

04 • 我的记录

- 哪一个片段虽然命中关键词，却没有真正回答问题?

- 如果改用向量检索，我会用哪些相同问题做比较?

忙碌日：20分钟

- 用3分钟读“分块决定检索返回什么大小的材料”，写下一个关键区别。

- 用12分钟运行 python run.py day 9，打开 outputs/day09.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 • 为检索建立三条人工判断

- 选择一条应命中的问题、一条同义改写、一条无答案问题。

- 为每条写期望来源与理由。

- 用相同问题比较两组参数，避免只展示最好看的案例。

参考资料与阅读范围

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

RAG 原始论文：只读摘要与方法概览；本课词项检索教学基线不复现论文模型。

https://arxiv.org/abs/2005.11401

下一步

下一课把检索结果组织成带引用的回答，并检查引用能否回到材料。

第10天 · 让回答带着证据一起出现

把检索、引用与无证据时停止连接起来。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 解释RAG中检索与生成分别负责什么。

- 核对回答中的证据ID与摘录。

- 用无支持问题检查系统是否克制。

开始之前

- 完成第9课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

RAG给回答补充可定位的材料

RAG把检索得到的材料交给回答环节，使结果可以依据当前资料，而不只依赖模型已有知识。本课先检索教学记录，再组织回答与引用；离线模式由规则完成，真实模型路径可在已配置时另做比较。

RAG不是把任意文件塞进提示后就保证正确。导入可能漏内容，检索可能选错，回答可能夸大，引用也可能张冠李戴。必须把这些环节拆开检查，才能知道应该改哪一部分。

引用需要一个能够回到原文的锚点

每条引用应尽量带原记录ID、来源和对应文字。拿到结果后，先检查引用ID是否存在，再检查quote是否来自该记录，最后检查引用是否支持答案中的具体主张。三步缺一不可。

程序可以较容易检查ID和文字匹配，但“材料是否真正支持结论”往往仍需人工判断。摘录存在，不等于科学推断成立；例如材料讨论相关性，回答却下因果结论，仍然需要纠正。

没有支持材料时，停止也是结果

当问题超出当前资料范围，系统应明确说明证据不足。拒答不代表系统一无是处，它让使用者知道需要补材料、换工具或把问题转交给人。用猜测填补空白会让后续研究更难复查。

拒答策略也需要评价：过于保守可能漏答本来有依据的问题，过于宽松可能给出没有支持的答案。因此要同时准备有答案和无答案的例子，并分别记录。

区分来源支持与材料本身的质量

即使回答忠于某份材料，那份材料仍可能过时、错误或只适用于特定条件。课程中的教学记录用于演示接口，不能代替系统检索完整文献、评价研究设计或形成真实研究结论。

在自己的工作中，可以把结果分成“材料明确支持”“合理但待核验的推断”“当前无证据”三类。先保持边界，再逐步增加高质量来源与更可靠的检查。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行带引用问答

为什么做：先得到可检查的完整结果。

用默认day10.question运行。

```
python run.py day 10
```

预期结果：得到answer与引用，保留模式和终止状态。

2. 从引用回到材料

为什么做：检验可追溯性。

打开输出，并根据evidence_id在data/evidence.json找到对应记录。

```
python -m json.tool outputs/day10.json
```

预期结果：能核对一条引用的原文，并用自己的话说明它支持哪一句。

3. 改变问题

为什么做：检验材料范围，而不是只展示成功例子。

将day10.question改为当前教学库没有依据的具体问题，保存后重跑。

```
python run.py day 10
```

预期结果：观察是否拒答或给出证据不足；如错误回答，应如实记录为失败。

4. 恢复并写边界

为什么做：给结果增加可使用的解释。

恢复默认问题，写出检索命中、引用匹配与科学支持这三层检查的不同。

预期结果：保留一份可回答和一份无支持问题的结果。

改一改 · 固定检索参数，只改变问题

- 保留day10.top_k=3，使用默认question运行第10课并记录status和citations。

- 仅将question改为zephyrquartz gondola 92851，再运行python run.py day 10，检查是否出现abstained与空引用。

- 恢复默认问题。这个人为无匹配串只验证无命中分支；另选自然语言问题检查，若词面误命中造成错误回答，就如实登记失败。

观察什么：默认问题有教学资料支持；人为无匹配串进入明确拒答路径。自然语言难题仍需逐项人工检查，不能因这个简单负例通过就认为拒答可靠。

机制解释：一次只改变问题可以检验材料范围与无命中行为；引用是否支持更强主张仍需另外核对。

常见问题

引用存在但与结论无关

原因：把检索相关性当成语义支持。

处理：按句子核对材料实际说了什么，删除或改写没有依据的结论。

无答案问题仍出现肯定答案

原因：拒答策略或检索阈值存在不足。

处理：保存失败输入和输出，纳入后续评价；不能只删除不理想的案例。

03 • 验收

第10课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day10.json

☐ 至少核对一条引用ID、摘录与对应主张。

☐ 运行一次材料外的问题并如实记录状态。

☐ 能说明RAG没有自动消除事实错误和科学推断风险。

自测与解析

1. 检查引用时，哪一步不能省？

- A. 只看URL能否打开

- B. 只看引用数量

- C. 核对来源内容是否支持具体主张

- D. 只看回答长度

答案：C。 可访问来源与引用数量都不能替代主张层面的证据核验。

2. 当前资料没有问题所需证据时，合理结果是什么？

- A. 编一个最可能的答案

- B. 无限重复同样检索

- C. 说明不足并指出需要补充的材料

- D. 隐藏引用字段

答案：C。 清楚暴露证据缺口，比生成无法复查的猜测更有助于后续工作。

小词典

- RAG：检索相关材料，并让回答环节基于这些材料组织结果的方法。

- Grounding：把输出中的主张与可定位的材料连接起来。

- 拒答/Abstention：在支持条件不足时明确不提供肯定结论。

04 · 我的记录

- 哪句话来自原文，哪句话来自我的推断?

- 如果资料是错的，即使忠实引用，我还需要哪一道检查?

忙碌日：20分钟

- 用3分钟读“RAG给回答补充可定位的材料”，写下一个关键区别。

- 用12分钟运行 python run.py day 10，打开 outputs/day10.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 给同一回答做一次红笔复核

- 把每句主张旁边写上证据ID。

- 找出没有明确支持的句子。

- 将其删除、弱化或标为待验证推断，并记录原因。

参考资料与阅读范围

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

RAG 原始论文：只读摘要与方法概览；本课词项检索教学基线不复现论文模型。

https://arxiv.org/abs/2005.11401

下一步

下一课用标注问题评价检索与拒答，不再只看单个成功演示。

第11天 · 用小评价集检查检索与拒答

把“看起来不错”转成有分母、能定位失败的观察。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 计算并解释证据召回与两类通过率。

- 用同一组问题比较top_k=1和3。

- 区分公开开发集表现与真实场景可靠性。

开始之前

- 完成第10课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

评价首先要说明题目与标准

评价集是一组问题及其预先写明的期望。今天使用data/eval_dev.json公开教学开发集，其中可回答题具有应出现的证据ID，不可回答题没有支持证据。先读一条题目和标签，再看系统表现，避免事后根据输出修改正确答案。

这些案例数量很小、全部公开，适合发现接口与流程问题。它们不是隐藏测试，更不是对真实科研能力的独立验证。开发集上得分提高，说明在这套明确规则下变好；是否能推广到新问题，需要另做评价。

检索召回回答的是“找回来多少”

一条可回答题的证据召回等于“检索到的期望证据数量÷全部期望证据数量”。例如需要E03和E07，只找到E07，这条题的召回是1/2。当前程序对可回答题逐题计算，再求平均；无支持证据题的召回记为null，不把它塞进相同分母。

召回不惩罚所有无关材料。找回正确证据的同时夹入很多噪声，召回仍可能很高。因此要打开items检查具体结果，并考虑回答是否引用全部必要材料、是否引入不支持的主张。不要用一个数字替代全部质量判断。

可回答与不可回答分别检查

当前answerable_accuracy的通过条件是状态为answered，且引用集合包含该题全部期望证据ID。abstention_accuracy则统计本来没有支持证据的题，系统是否返回abstained。它们是教学程序的规则指标，不会自动判断自然语言中的每个科学结论。

这个区别能发现一种常见失败：系统给出了答案，引用也真实，却遗漏了题目必须同时考虑的另一条证据。另一种失败是任何问题都回答；如果没有单独设置不可回答题，漂亮的总体结果可能掩盖这种问题。

让检索与回答使用同一个实验条件

今天的top_k同时传给检索和回答路径，避免一边评价取1条、另一边回答偷偷取3条。先用1，再用3，其他条件保持不变。比较时记录配置、每个指标的分母，以及哪一题发生变化。

当前教学集D6要求同时覆盖E03和E07，top_k=1无法同时取回两条；top_k=3能够展示完整覆盖的区别。这个案例说明了指标如何定位缺口，不代表以后top_k越大越好：更多材料也会增加噪声和上下文成本。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行公开开发集

为什么做：建立与后续修改可比较的基线。

保持day11.split为dev、top_k为3，运行本课。

```
python run.py day 11
```

预期结果：得到evaluation中的retrieval_recall、answerable_accuracy、abstention_accuracy和items。

2. 直接比较两个检索数量

为什么做：确保参数真实传入评价程序。

依次运行下面两条命令，把两次终端结果复制到学习笔记；同名evaluation_dev.json会更新。

```
python run.py evaluate --split dev --top-k 1
python run.py evaluate --split dev --top-k 3
```

预期结果：top_k=1时D6缺少部分期望证据；top_k=3时该教学案例覆盖完整。

3. 回到失败题核对分母

为什么做：从指标追到具体材料。

打开开发集与结果，找到D6的expected和retrieved，再人工算一次该题召回。

```
python -m json.tool data/eval_dev.json
python -m json.tool outputs/evaluation_dev.json
```

预期结果：能够解释这条题的1/2与整体平均值为什么不同。

4. 写一段评价结论

为什么做：把观察与推广范围分开。

用三句话记录配置、发生变化的案例、仍未检验的能力；不要把公开小题集得分写成科学结论正确率。

预期结果：结论包含分母与适用范围，指出仍需人工核对语义支持。

改一改 · 让D6从部分证据变成完整证据

- 在experiments.json中只把day11.top_k改为1，运行python run.py day 11。

- 记录D6的expected、retrieved、recall和passed，再将top_k改回3运行。

- 对照其余题是否变化；解释为什么当前测试的改善不意味着所有问题都应增加检索数量。

观察什么：当前公开教学集中D6从部分覆盖变为完整覆盖；指标计算使用同一个top_k驱动检索与回答。

机制解释：逐题记录让数字变化具有可解释的原因；结果仅反映本套公开开发题的规则检查。

常见问题

只看到最终一次评价结果

原因：两次命令写入同名输出文件。

处理：运行下一次前复制终端结果到笔记，或给自己的记录注明配置；不要把两次混成一份。

达到满分就认为可以交付真实研究

原因：当前题目很少且已公开，指标按证据ID和状态计算。

处理：追加人工检查与新问题，并把公开开发集范围写入报告；不要修改标签迎合输出。

03 • 验收

第11课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day11.json

☐ 手工计算过一条题的证据召回。

☐ 保留top_k=1与3两次对照并指出D6的具体变化。

☐ 说明拒答率、可回答题通过率与科学语义正确性不是同一件事。

自测与解析

1. 一题需要E03和E07，检索只找到E07，该题召回是多少?

- A. 0

- B. 1/2

- C. 1

- D. 无法计算

答案：B。 已找到1条期望证据，共需2条，因此召回是1/2；无关材料多少不改变这个分母。

2. 公开开发集上的高分最直接说明什么?

- A. 所有科研问题都能正确回答

- B. 模型拥有真实科研能力

- C. 当前实现通过了这套已知题目的规定检查

- D. 无需再检查拒答

答案：C。 它证明的是明确题集与指标下的表现，不能替代未知问题测试或科学语义验证。

小词典

- 召回：检索到的期望证据占全部期望证据的比例。

- 开发集：用于观察和改进系统的已知案例集合。

- 拒答评价：检查没有支持材料时是否明确停止作答。

04 · 我的记录

- 哪一条失败最值得先修：没检索到、引用不完整，还是该拒答却回答？

- 自己的真实问题应如何写出可检查的期望，而不只写“答案很好”？

忙碌日：20分钟

- 用3分钟读“评价首先要说明题目与标准”，写下一个关键区别。

- 用12分钟运行 python run.py day 11，打开 outputs/day11.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 手工扩展一条评价案例

- 选择一个必须同时考虑两条教学证据的问题。

- 写出期望证据ID和为什么两条都需要，不先改程序。

- 记录当前输出，再判断问题、标签和评价规则是否一致。

参考资料与阅读范围

BioAgentLab：评价与校准：关注评价单位、失败分类、拒答和人工复核。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/18_evaluation_and_calibration.md

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

下一步

下一课把中间状态保存下来，让任务可以从已完成阶段继续。

第12天 · 保存状态，让任务能够继续

检查点记录做到哪里，恢复时先确认还是同一项任务。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 区分当前状态、检查点与长期记忆。

- 实际观察首次运行与重复恢复的不同。

- 验证问题或语料改变时不会误用旧检查点。

开始之前

- 完成第11课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

状态描述的是这一次任务

状态包含当前问题、已取得的证据、执行阶段和待完成步骤。它像实验记录中的当前进度，告诉程序接下来可以做什么。它不等于用户长期偏好，也不意味着模型已经通过这些记录重新训练。

本课先检索教学记录，把证据ID保存到outputs/checkpoint.json，再从保存的ID读取材料并组装答案。恢复时继续使用已保存的检索结果，而不是只打印一句“已恢复”却重新执行所有阶段。

检查点要有身份与版本

一个可用检查点至少需要说明属于哪个问题、语料是什么版本、当前阶段和已完成结果。课程文件记录schema_version、question、corpus_sha256、phase与evidence_ids，使程序能判断它是否适用于当前任务。

仅仅发现一个同名文件是不够的。如果用户换了问题，或资料内容已经变化，旧证据可能不再适合。本课只有问题和语料摘要同时一致，且resume允许恢复，才复用已有记录；否则建立本次的新检查点。

恢复成功与最终正确分别验收

loaded_existing表示是否读取了适用的已有检查点。checkpoint.phase描述保存时的阶段，state.phase描述本次继续执行后的阶段。二者不同是正常情况：从retrieved阶段恢复后，可以到completed阶段。

恢复说明控制流程能够继续，不自动证明证据正确、引用充足或研究结论可靠。还要按前面课程检查材料与回答。对于会发送消息、扣费或写外部系统的动作，恢复还涉及重复执行问题，第17课再学习批准与幂等。

用主动重跑理解恢复条件

先将resume设为false运行，得到明确的新起点。再设回true，用完全相同的问题运行，观察loaded_existing成为true。最后只改问题再运行，验证旧检查点没有被错误复用。

不需要删除整套outputs，也不需要强行中断电脑。这个小实验就能演示保存、匹配与恢复；如果检查点文件损坏，应报告格式问题并从新任务重新建立，而不是悄悄接受半份状态。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 建立新检查点

为什么做：让首次与复用状态可区分。

把day12.resume设为false，保留默认question并执行；这会重新写本课检查点。

```
python run.py day 12
```

预期结果：result.loaded_existing为false，生成outputs/checkpoint.json并完成本次组装。

2. 查看真正保存的内容

为什么做：核对恢复所依据的数据。

检查文件中的question、corpus_sha256、phase与evidence_ids。

```
python -m json.tool outputs/checkpoint.json
```

预期结果：保存的是检索阶段证据ID及任务身份，不是模型训练参数。

3. 恢复相同任务

为什么做：验证程序确实复用已完成阶段。

将day12.resume改回true，不改变question，再运行。

```
python run.py day 12
```

预期结果：loaded_existing为true；state.resumed_from为retrieved，state.phase为completed。

4. 给旧检查点一个新问题

为什么做：确认任务身份检查生效。

只把day12.question改成What does relative abundance measure?，再运行并比较；最后恢复原问题，按需重建检查点。

```
python run.py day 12
```

预期结果：新问题不复用旧问题的检查点；loaded_existing为false，证据随任务更新。

改一改 · 只改变resume，观察是否复用

- 固定question和语料，设置day12.resume=false运行，记录loaded_existing。

- 仅将resume改为true再运行，比较loaded_existing与保存的evidence_ids。

- 说明第一次是建立检查点，第二次是从适用的检查点继续；保留两次结果。

观察什么：同一问题与语料下，关闭恢复时建立新记录，开启恢复后可以复用；答案相似不等于两次走了完全相同路径。

机制解释：恢复机制需要可以观察的身份匹配与状态转移，不能只依靠最终答案判断。

常见问题

第一次就显示loaded_existing=true

原因：此前运行留下了适用于同一问题与语料的检查点。

处理：先用resume=false运行一次，再开启恢复；不要把已有记录误认为程序状态错误。

恢复时JSON解析失败

原因：检查点被手工编辑成了无效JSON或文件不完整。

处理：先保留错误记录供排查，将resume设为false重新建立；不要在不知道字段含义时随意修改状态。

03 • 验收

第12课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day12.json

☐ 能够找到并解释检查点的任务身份和语料摘要。

☐ 完成resume=false与true的可复现对照。

☐ 通过修改问题确认旧检查点不会被误用。

自测与解析

1. 只要outputs/checkpoint.json存在，是否就应该使用?

- A. 是，文件名相同即可

- B. 是，任何问题都能共享证据

- C. 否，还要核对问题、语料版本与恢复条件

- D. 否，状态永远不能持久保存

答案：C。 存在文件仅说明曾经保存过；任务身份与输入版本匹配才能合理复用。

2. 保存检查点后，模型发生了什么变化?

- A. 自动完成了再训练

- B. 获得了永久记忆能力

- C. 模型权重必然改变

- D. 程序保存了任务状态，不能据此推断模型被训练

答案：D。 检查点是应用层记录；是否训练模型是另一套过程。

小词典

- State：当前任务输入、阶段和中间结果的集合。

- Checkpoint：保存下来供后续匹配和恢复的状态。

- 任务身份：区分当前任务与其他任务的输入及版本信息。

04 · 我的记录

- 当前检查点足以解释为什么恢复到这个阶段吗?

- 如果未来工具具有外部写入动作，重跑时需要额外防止什么?

忙碌日：20分钟

- 用3分钟读“状态描述的是这一次任务”，写下一个关键区别。

- 用12分钟运行 python run.py day 12，打开 outputs/day12.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 设计一个检查点升级方案

- 假设下一版增加required_evidence字段，写出旧检查点缺字段时如何处理。

- 说明何时可以迁移，何时应重新运行。

- 给恢复失败写一条清楚的提示，并保留原始错误便于排查。

参考资料与阅读范围

BioAgentLab：状态与记忆：只读状态、会话和长期记忆的不同用途，以及生命周期。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/05_state_and_memory.md

Python 官方：JSON 编码与解码：只读基本读写、Python/JSON类型对应和格式错误说明。

https://docs.python.org/3/library/json.html

下一步

下一课处理跨任务记忆，加入用户归属、过期和删除。

第13天 · 让长期记忆有归属、期限与删除

只记住有用途的信息，并能说明谁可以读取、何时失效。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 区分任务状态与跨任务偏好。

- 观察用户归属对读取结果的影响。

- 用逻辑时间验证过期与删除，不等待真实时间流逝。

开始之前

- 完成第12课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

长期记忆服务于下一次任务

用户偏好“每天学习一小时”可以影响之后的计划；当前问题检索到了E03则通常只属于这次任务。把两者混在一个无限增长的聊天记录里，会让后续任务难以判断哪些信息仍然适用。

本课用一条合成偏好演示记忆生命周期。程序写入本地JSON并检查归属、期限与删除；它不是生产环境账号系统，也没有把这条记录训练进模型权重。理解这个范围，有助于准确设计下一步。

读取时先核对记录归属

一条记录拥有owner字段，查找时需要指定读取者。默认创建者alice应能看到未过期记录，而读取者bob应得到空列表。owner_visible与other_user_visible都是记录数组，不是单个布尔值。

如果把lookup_user也设成alice，两次读取自然都属于同一个用户，不能据此认为隔离失效。真正的服务还要从经过认证的会话确定用户身份；不能接受任意人随便填写alice就获得权限。

有效期与物理删除解决不同问题

TTL表示有效期限。本课基于固定教学起点，用elapsed_minutes模拟已经过去多久；ttl_minutes=60、elapsed_minutes=30时未过期，elapsed_minutes=90时已过期。这样可以在一分钟内重复观察边界，不必真的等90分钟。

过期检查让记录不再出现在可见结果中，并不意味着所有副本都已物理删除。delete=true则会从本次示例记录集合移除该用户条目并写入memory.json。课程每次重新构造演示数据，不能把重跑生成新记录误认为删除功能恢复了旧记录。

记忆需要纠正和最小化

如果用户从每天一小时改成每周三次，系统应能更新偏好与来源，而不是永久坚持旧假设。真正有用的记忆应有明确用途、可靠归属、可修改范围和退出方式，避免为了“更聪明”而保存无关私人内容。

今天只用合成用户标识和一条学习偏好。验收时分别检查未过期读取、其他用户读取、过期与主动删除四种情形；不要仅看文件存在就说生命周期已经完成。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行未过期基线

为什么做：先看正常用户与其他用户的差别。

保持user_id=alice、lookup_user=bob、ttl_minutes=60、elapsed_minutes=30、delete=false。

```
python run.py day 13
```

预期结果：owner_visible含一条记录，other_user_visible为空数组；expired和deleted都为false。

2. 查看保存内容与逻辑时间

为什么做：理解文件存在和是否可见的区别。

打开输出与记忆文件，比较owner、expires_at和logical_now。

```
python -m json.tool outputs/day13.json
python -m json.tool outputs/memory.json
```

预期结果：可以指出归属与到期时间；不会把固定教学时间误认为真实服务器时钟。

3. 模拟时间已经过去

为什么做：验证过期规则。

只把day13.elapsed_minutes改成90，其他条件保持基线，再运行。

```
python run.py day 13
```

预期结果：expired为true，owner_visible和other_user_visible均为空；无需真实等待。

4. 验证主动删除

为什么做：区分删除与过期。

将elapsed_minutes恢复30，再把delete改为true运行；记录结果后恢复delete=false。

```
python run.py day 13
```

预期结果：deleted为true，memory.json是空记录列表；此时无需靠过期来阻止读取。

改一改 · 只改变读取者，核对隔离

- 固定user_id=alice、ttl_minutes=60、elapsed_minutes=30、delete=false，用lookup_user=bob运行一次。

- 仅把lookup_user改成alice再运行，比较other_user_visible。

- 恢复bob，并解释这只是过滤规则演示，真正服务还需要验证请求者身份。

观察什么：不同用户读取不到alice记录；同一用户查询能够得到未过期记录。

机制解释：归属过滤是记忆隔离的一部分；请求身份可信与访问规则生效需要分别验证。

常见问题

两个读取结果都非空

原因：lookup_user与user_id相同，或查看的是此前输出。

处理：检查配置是否仍是alice与bob，并保存后重新运行。

删除后重跑记录又出现

原因：本课每次运行都会重新构造一条合成记录。

处理：把每次运行看成独立教学实验；观察delete=true的那次输出和文件，别把演示初始化当成持久业务流程。

03 • 验收

第13课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day13.json

☐ 解释owner_visible与other_user_visible是数组，并完成不同读取者对照。

☐ 观察elapsed_minutes=90时expired=true且记录不可见。

☐ 完成delete=true实验，解释过期与删除的区别。

自测与解析

1. ttl_minutes=60、elapsed_minutes=90时，应出现什么?

- A. 记录仍应对所有人可见

- B. 记录因过期不再出现在可见结果中

- C. 自动把用户账号删除

- D. 模型忘记所有知识

答案：B。 期限控制当前记录的可见性；它不等于删除账号或修改模型知识。

2. 怎样理解本课的owner过滤?

- A. 已经构成完整登录与鉴权系统

- B. 只要随便填一个用户名就安全

- C. 展示按归属过滤记录；真实请求还需可靠身份认证

- D. 能替代所有数据治理

答案：C。 过滤规则需要可信的用户身份配合；本地教学示例没有完整认证服务。

小词典

- TTL：记录保持有效的时长。

- 用户隔离：读取时限制只能访问当前用户有权取得的记录。

- 逻辑时间：用于可重复实验的人为时间条件，不必等待真实时间。

04 · 我的记录

- 什么信息有必要跨任务保留，什么只属于当前任务？

- 如果一条记忆已经过时，用户如何发现并纠正它？

忙碌日：20分钟

- 用3分钟读“长期记忆服务于下一次任务”，写下一个关键区别。

- 用12分钟运行 python run.py day 13，打开 outputs/day13.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 写一张记忆说明卡

- 为“每日学习时长”写用途、来源、有效期和更新方式。

- 补上“用户要求忘记时”的处理。

- 列出两种不值得保存的无关信息，说明为什么没有必要收集。

参考资料与阅读范围

BioAgentLab：状态与记忆：只读状态、会话和长期记忆的不同用途，以及生命周期。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/05_state_and_memory.md

Python 官方：JSON 编码与解码：只读基本读写、Python/JSON类型对应和格式错误说明。

https://docs.python.org/3/library/json.html

下一步

下一课把检索、证据、预算与检查整合成第二周作品。

第14天 · 第二周作品：一个有边界的证据问答助手

交付可重复的案例、引用与失败说明，检验两周学习成果。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 用统一命令运行证据问答小项目。

- 核对答案、引用、预算和人工复核标记。

- 用一个正常问题和一个失败条件验收作品。

开始之前

- 完成第13课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

作品围绕一个有限科研概念展开

本次问题是What does relative abundance measure?，教学资料说明相对丰度是样本内的比例。你需要交付可以追溯到教学记录的解释，以及材料没有支持的部分；不需要在这一小时内训练模型或安装整套微生物分析流程。

相对比例变化不能直接当作绝对数量变化；实际研究还需要合适测量与方法前提。这与已有CLEAR-Microbiome和SCUT_Minilab_MEF资料中的方法阅读方向相关，但课程没有执行这些仓库，也不声称复现任何真实研究结果。

把已有模块连起来，再检查每个边界

本课复用问题输入、检索工具、有限步骤、引用结构与答案检查。result.answer记录状态、答案、引用和轨迹；project_checks汇总结构检查、预算检查与人工复核要求。集成的意义在于让这些模块围绕同一任务合作。

project_checks.answer_contract本身是一个带valid和errors的对象，不应只检查它存在就说通过。bounded反映本实现的轨迹上限检查；human_review_required提醒继续核对材料。结构通过、执行结束和科学解释成立仍是三种不同判断。

清楚标注教学材料与外部来源

offline答案由给定教学记录片段组装，citations中的quote来自这些原创教学说明。source_url提供进一步阅读的真实外部资料入口，source_kind会说明它不是外部论文逐字引文。查看链接时，应实际阅读再判断它支持哪些内容。

交付中可以写“这次运行检索到E02并引用了教学说明”；不能写“系统自动阅读全文并验证了论文”。准确说明证据来源，比包装成更强的自动化能力更有助于后续改进。

给作品设置正常路径和失败路径

正常路径使用默认问题与足够预算；失败路径可以只把max_steps设为1，观察系统是否清楚说明预算耗尽。还可以在后续练习中加入没有材料支持的问题，核对拒答。

周作品完成的标志是你能解释这些结果并知道还缺什么，而不是每一次都得到一大段答案。疲惫时先完成正常案例与一项复核，把失败实验留在下一次；记录精简进度，不把未做事项勾成完成。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行第二周项目

为什么做：获得集成后的完整结果。

保持day14默认问题、top_k=3、max_steps=6，执行本课。

```
python run.py day 14
```

预期结果：得到answer与project_checks；默认教学问题返回E02相关说明。

2. 沿着引用回到材料

为什么做：检查来源性质与支持范围。

查看citation的evidence_id、quote、source_url和source_kind，再对照data/evidence.json中E02。

```
python -m json.tool outputs/day14.json
```

预期结果：能解释quote来自教学记录，外部URL是进一步阅读入口。

3. 主动缩小运行预算

为什么做：让失败行为也成为交付的一部分。

只把day14.max_steps改为1，再次运行；记录status与trace，随后恢复为6。

```
python run.py day 14
```

预期结果：低预算运行出现budget_exhausted等明确停止结果，不应冒充完整答案。

4. 完成一页作品说明

为什么做：让另一位学习者能复跑并检查。

写下运行命令、材料范围、一个正确解释、一次失败条件、剩余人工复核事项；恢复默认配置后运行一次作为交付。

```
python run.py day 14
```

预期结果：留下正常结果和失败记录，说明没有完成真实论文分析或专业流程复现。

改一改 · 固定问题，只改变最大步数

- 使用相同question与top_k，以max_steps=6运行并保存状态、引用数量和轨迹。

- 仅将max_steps设为1重跑，比较状态与实际执行步骤。

- 恢复6；在作品说明写清完整运行与预算停止各自可以交付什么。

观察什么：预算足够时可以组装教学引用答案；预算过小时必须显式停止，不能把部分结果当作完成。

机制解释：集成项目不仅验收成功结果，也检验约束在整条路径中是否真实生效。

常见问题

answer_contract对象存在就判定通过

原因：把对象存在与对象内valid=true混淆。

处理：打开valid与errors字段逐项检查，并额外核对语义支持。

把教学摘录当作论文逐字引用

原因：忽略了source_kind对材料性质的说明。

处理：将引用标为教学记录；需要引用论文时自己打开原文核对准确措辞与上下文。

03 • 验收

第14课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day14.json

☐ 能用一条命令复跑正常问题并找到E02来源记录。

☐ 逐项检查answer_contract.valid、bounded和human_review_required的含义。

☐ 交付包含正常案例、预算失败案例和明确材料范围。

自测与解析

1. 相对丰度比例上升，能否直接断定绝对数量增加?

- A. 能，比例与数量相同

- B. 不能，需要额外测量或有根据的换算

- C. 能，只要答案带引用

- D. 能，只要Agent运行成功

答案：B。 相对丰度由组成中的比例决定，单看比例不能直接推出绝对数量变化。

2. 本课哪种交付说明最准确?

- A. 自动复现了CLEAR和MEF全部研究

- B. 已证明所有微生物学结论正确

- C. 基于给定教学材料完成有限问答，保留引用与待复核事项

- D. 无需原文复核

答案：C。 交付应与实际材料、程序行为和已完成验证一致。

小词典

- 集成验收：围绕同一任务检查多个模块之间的输入、输出与边界。

- 相对丰度：在一个样本组成中某部分所占的比例。

- 失败路径：条件不足或运行出错时程序采取的明确行为。

04 · 我的记录

- 哪一个看似顺畅的输出最容易让我过度相信?

- 如果明天把新资料加入语料，哪几项验证需要重做?

忙碌日：20分钟

- 用3分钟读“作品围绕一个有限科研概念展开”，写下一个关键区别。

- 用12分钟运行 python run.py day 14，打开 outputs/day14.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 · 制作两分钟作品演示

- 第一分钟展示问题、命令与引用记录。

- 第二分钟展示预算失败以及为什么需要人工复核。

- 请一位同学指出一处说明可能被误解的地方，再修正措辞。

参考资料与阅读范围

BioAgentLab：Agentic RAG 与证据：关注证据台账、来源与回答之间的对应关系。
https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/10_agentic_rag_and_evidence.md

BioAgentLab：评价与校准：关注评价单位、失败分类、拒答和人工复核。
https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/18_evaluation_and_calibration.md

BioAgentLab：工具契约：对照输入、输出、失败与权限四种契约，不必通读全书。
https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/06_tool_contracts.md

下一步

第三周开始处理多步骤规划、恢复和人工批准，让任务协作更可靠。

第15天 · 用依赖图组织多步骤任务

先知道哪些步骤必须等待，再讨论自动执行。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 把任务写成步骤和依赖关系。

- 阅读拓扑排序给出的可行顺序。

- 通过真实循环依赖观察计划验证失败。

开始之前

- 完成第14课，或能在practice目录运行日课命令。

- 只使用教学材料；修改配置前保留原值，实验结束后恢复或记下修改。

一小时安排

- 15分钟 · 理解概念：阅读四节讲解，记下今天最关键的一个区别；外部资料按指定小节选读。

- 30分钟 · 操作与修改：先完成四步带练，再改变一个参数，保留修改前后的结果。

- 10分钟 · 验收与自测：对照三项验收，完成两道自测并阅读解析。

- 5分钟 · 记录与收尾：写下结果、未解决的问题和下次打开项目后的第一步。

01 · 理解概念

计划是带前提的动作集合

一个证据报告可以分成retrieve、analyze、draft与review。分析需要先拿到材料，草稿需要分析结果，审阅需要草稿。这些前提比单纯列出四个动词更重要，因为它们决定某一步是否已经具备输入。

今天用固定教学图演示依赖检查，不调用模型生成计划，也不真正运行四个科研节点。程序会检查图并给出可行顺序；接下来几课才逐步讨论执行失败、重试、批准和状态管理。

有向无环图表达可以完成的依赖

图的节点是任务，依赖表示执行某节点之前必须完成什么。本课graph用“节点→其前置节点列表”的JSON表示，例如analyze:[retrieve]表示分析依赖检索。不要把这个存储方向误读成分析要先于检索。

当依赖中没有环，可以按前提依次完成。若检索又等待最后的审阅，审阅等待草稿，草稿等待分析，分析等待检索，就没有任何步骤能够先开始。实际程序应拒绝这样的计划，而不是无限等待。

拓扑排序给顺序，不证明方法正确

拓扑排序寻找满足全部依赖的执行次序。当前实现会拒绝未知依赖和循环；无依赖的节点可以先安排。对于存在独立分支的图，可能有多个合理顺序，顺序不同不一定代表错误。

即使图有效，仍要检查每一步的方法、输入、权限与验收条件。例如“根据相关性宣布因果”的节点可以排列得很整齐，却没有科学依据。结构正确只是可执行计划的一项条件。

给每个节点写出完成标准

retrieve完成的标准可以是留下可定位来源；analyze完成的标准可以是记录支持与不支持的结论；draft需要引用与局限；review需要记录哪些项已核对。完成标准越具体，程序越容易判断是否可以进入下一步。

真实任务中还会有等待用户批准、可重试失败、预算不足与取消。这些状态不宜硬塞进“成功/失败”两个词。今天先把依赖讲清楚，之后再给节点增加适合的执行控制。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 运行无环图

为什么做：从一个有效计划建立基线。

保持day15.force_cycle为false，执行本课。

```
python run.py day 15
```

预期结果：valid为true，execution_order满足retrieve、analyze、draft、review的依赖。

2. 逐项核对依赖方向

为什么做：确保读懂图的实际表示。

打开graph，对每个节点检查其前置节点是否出现在execution_order更前面。

```
python -m json.tool outputs/day15.json
```

预期结果：能用自己的话解释analyze依赖retrieve，而不是反过来。

3. 制造一个真正的环

为什么做：验证无效计划被拒绝。

只将day15.force_cycle改为true，再运行；程序会让retrieve反过来依赖review。

```
python run.py day 15
```

预期结果：valid为false、execution_order为空，并出现循环依赖错误说明。

4. 恢复并写节点验收

为什么做：把图结构与任务标准对应。

恢复force_cycle=false并重跑；为四个节点各写一个可检查的完成条件。

```
python run.py day 15
```

预期结果：图重新有效；笔记说明今天验证的是依赖结构，尚未执行这些专业任务。

改一改 · 一条反向依赖为什么让整个计划停住?

- 用force_cycle=false保存graph和execution_order。

- 仅将force_cycle设为true，比较graph中retrieve的依赖变化与error。

- 用一句话解释为什么任何节点都无法首先执行，然后恢复false。

观察什么：增加retrieve依赖review后形成闭环，拓扑排序拒绝计划并返回明确错误。

机制解释：依赖检查能在实际工具执行之前发现无法调度的计划，避免把逻辑问题推迟到运行中。

常见问题

把依赖表当成执行顺序

原因：JSON对象中的节点排列与前置依赖是两回事。

处理：查看execution_order，再逐项验证每个依赖是否在前。

计划有环却想靠多重试解决

原因：不存在满足全部前提的开始节点，重复运行不会消除逻辑矛盾。

处理：修改错误依赖或拆出阶段，恢复force_cycle=false验证；不要通过无限等待绕过。

03 • 验收

第15课运行结果、一次参数对照，以及自己能解释的学习笔记。

成果文件：outputs/day15.json

☐ 能正确解释graph中的前置依赖方向。

☐ 运行过正常图与force_cycle=true的实际失败对照。

☐ 为每个节点写出完成标准，并说明有效计划不等于科学方法有效。

自测与解析

1. analyze:[retrieve]在本课中表示什么?

- A. 先分析再检索

- B. 分析必须等待检索完成

- C. 二者必须同时执行

- D. 二者没有关系

答案：B。 列表记录该节点的前置依赖，因此retrieve应在analyze之前。

2. 拓扑排序成功能够说明什么?

- A. 研究设计已经正确

- B. 所有工具都执行成功

- C. 依赖结构允许至少一种执行顺序

- D. 无需人工复核

答案：C。 拓扑排序验证依赖与顺序，不能代替节点实际执行、方法审查或结果核验。

小词典

- DAG：有向无环图，用于表示没有循环等待的依赖。

- 拓扑排序：按前置依赖安排节点的可行顺序。

- 循环依赖：多个节点互相等待，形成无法首先开始的闭环。

04 • 我的记录

- 自己的任务中，哪一步最容易在输入还没准备好时就开始?

- 如果一个步骤失败，哪些下游任务需要等待，哪些仍能继续?

忙碌日：20分钟

- 用3分钟读“计划是带前提的动作集合”，写下一个关键区别。

- 用12分钟运行 python run.py day 15，打开 outputs/day15.json 并核对第一项预期结果。

- 用5分钟完成第一道自测并写一句解释；完整参数实验和其余验收留待下次。

留下本课运行结果与一句自己的解释；精简完成单独记录，不等于完整课程已经验收。

有余力再做：20分钟选学 • 把自己的小任务画成依赖图

- 选择一个只需要四至六步的工作。

- 写出每一步的输入、交付物与前置步骤。

- 找出可以独立开展的节点，并标出需要人工批准的动作。

参考资料与阅读范围

BioAgentLab：规划与 ReAct：关注行动、观察、停止条件与规划依赖，不把示意流程当成可靠性证明。

https://github.com/zbyloveyj/BioAgentLab/blob/main/book/zh/07_planning_and_react.md

Anthropic：Building effective agents：阅读 workflows 与 agents 的区别，以及从简单结构开始的讨论。

https://www.anthropic.com/research/building-effective-agents

下一步

下一课处理临时失败、重试次数与超时，让可行计划能够有边界地执行。

第16天 · 失败以后，怎样重试才合理

区分暂时故障、参数错误与时间预算，避免无限重试。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 解释总尝试次数与成功前失败次数的关系。

- 运行一个会成功和一个会耗尽预算的重试案例。

- 区分超时返回、取消工作与重复副作用。

开始之前

- 已经完成第 15 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

先判断失败属于哪一类

读取服务偶尔超时，与把字符串传给整数参数不是同一种错误。暂时故障可能在下一次调用时消失；参数错误通常会原样重复。把所有异常都放进无限重试循环，会消耗时间，也会掩盖真正的错误。

本课用可控制的失败次数制造暂时故障。failures_before_success 表示成功前会失败几次，max_attempts 是总尝试次数，包含第一次。先比较这两个数，再预测程序应该成功还是耗尽预算。

重试次数必须有明确含义

当成功前失败2次，总尝试上限为3时，第三次才有机会成功。若把总尝试数改成2，程序应报告失败而停止。这里训练的是边界意识，不是猜一个“足够大”的数字。

生产服务还要考虑退避、抖动、服务端提示和哪些错误可重试。本课使用小规模受控延迟便于观察，不把这个教学策略包装成适用于所有网络服务的通用配置。

时间预算与尝试次数分开控制

尝试次数限制防止不停调用；截止时间限制一次工作等待多久。work_delay_ms 与 deadline_ms 用毫秒表示，所以改变时间参数前先核对单位。操作没有报错，也可能因超时而无法交付可用结果。

某些线程超时只意味着调用方停止等待，并不自动杀死底层工作。阅读程序实现时要区分“返回超时状态”和“工作已被彻底取消”；真实进程或远端请求需要相应取消机制。

重试可能重复产生副作用

检索同一份材料通常可重复执行；发送邮件、写入报告或提交任务若被重复执行，可能产生两份结果。不要因为第一次请求没有返回，就假定第一次一定没有发生。

下一课会给写操作加入内容摘要、人工批准与幂等检查。本课先在笔记标出自己的流程中哪些动作只读、哪些会改变状态，并说明哪些允许直接重试。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day16。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day16'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day16.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 16
```

预期结果：结果写入 outputs/day16.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查retry.attempts、retry.status 和 deadline.status。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day16.json
```

预期结果：能够定位retry.attempts、retry.status 和 deadline.status，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 让成功发生之前，重试预算先耗尽

- 先执行 python run.py day 16，将 outputs/day16.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day16.max_attempts 为 2，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 16。

- 对比两份结果，重点观察retry.attempts、retry.status 和 deadline.status；解释改变原因后恢复原设置或明确记录保留值。

观察什么：默认成功前失败2次；将总尝试上限降为2后，应停止并给出耗尽或失败状态，而不是无限等待。

机制解释：重试上限是总尝试次数，第一次也计入。明确计数规则能避免越界与错误的可用性预期。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day16，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 · 验收

区分暂时故障、参数错误与时间预算，避免无限重试。 保留基线、修改结果与自己的解释。

成果文件：outputs/day16.json

☐ 能用参数预测成功或耗尽。

☐ 保留成功与失败两份输出，并指出停止原因。

☐ 写明一个不能盲目重试的有副作用动作。

自测与解析

1. 成功前失败2次，总尝试上限2次，应该怎样结束？

- A. 第三次再试一次

- B. 报告失败并停止

- C. 把错误忽略后报告成功

- D. 无限重试直到成功

答案：B。 第三次超出了明确预算。受控失败是正确行为，不能为了成功率悄悄增加次数。

2. 等待超时是否必然意味着底层工作已停止？

- A. 必然停止

- B. 一定回滚所有副作用

- C. 取决于执行与取消机制

- D. 意味着输入必定错误

答案：C。 超时、取消与回滚是不同机制，需要分别设计和验证。

小词典

- Retry：在明确条件与次数限制内重新尝试操作。

- Deadline：某次工作可消耗的时间边界。

- Side effect：写文件、发送请求等会改变外部状态的效果。

04 · 我的记录

- 我的任务遇到哪类错误应该立即停止？

- 如果没有收到响应，我怎样判断写入是否已经发生？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 16，打开 outputs/day16.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：concurrent.futures：阅读 Future.result(timeout) 与 Executor 的关闭行为。

https://docs.python.org/3/library/concurrent.futures.html

BioAgentLab：科研 Agent 教材：查阅运行失败、恢复与可靠性相关章节，按本课失败类型对照阅读。

https://github.com/zbyloveyj/BioAgentLab

下一步

下一课把写入前审阅与幂等执行落实到一个可检查的文件。

第17天 · 先审阅，再执行写入

让批准绑定到具体内容，用幂等记录避免重复执行。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 复现 needs_approval、written 与 already_applied 三种状态。

- 验证改变草稿后旧批准失效。

- 解释内容摘要、身份授权与科学复核的不同作用。

开始之前

- 已经完成第 16 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

批准的是具体动作与具体内容

“可以写报告”仍不够具体。一次可检查的批准应当说明写在哪里、准备写什么，以及执行会改变什么。本课固定写入项目输出目录中的 approved_report.txt，避免把演示扩展成任意路径写入工具。

第一次运行时不要填批准值。先阅读 preview 和 digest；程序应停在 needs_approval。只有你看过将写入的内容，才把相应摘要复制到 approved_digest 中，再运行同一任务。

摘要把批准和内容联系起来

本课摘要由固定目标路径与待写内容共同计算。相同目标和内容得到相同摘要；内容改变后，之前的批准不应继续适用。这能减少“审阅的是旧稿，执行的是新稿”的错误。

摘要不是加密，也不是完整的权限系统。本课的摘要用于把一次演示批准与文件内容联系起来。真实产品还需要验证批准者身份、权限、有效期与操作目标。

幂等让重复请求不重复做事

网络中断后用户可能再点一次提交。幂等设计让同一批准、同一内容重复提交时，报告 already_applied，而不是重复追加或制造多个副本。这里判断的是相同操作是否已经执行。

幂等不等于所有状态永远不变。新内容应产生新的摘要与新的审阅要求；不同动作还需要不同的操作标识。把全局“已批准”布尔值当作永久授权，会失去这层边界。

在科研流程中使用审阅节点

你可以把写入前预览迁移到分析报告、候选文献清单或批量文件整理。自动化负责准备结果，人负责确认具体内容，再让程序执行有限操作。

本课只写教学报告，不提交论文、不发送邮件，也不修改原始科研数据。完成后要能指向输出文件和批准记录，而不能把“程序生成了一份文本”当成内容已经经过科学审核。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 先查看草稿，保持未批准状态

为什么做：先知道程序读什么，后面才能解释输出为何改变。

确认 day17.approved_digest 为空字符串，再运行第17课。检查 preview 是否是你愿意写入的内容。

```
python run.py day 17
```

预期结果：status 为 needs_approval，输出待写预览与 digest。

2. 批准当前草稿后执行

为什么做：保持默认设置的一次运行，是后续比较的参照。

在输出中复制完整 digest，粘贴到 experiments.json 的 day17.approved_digest；不要改 draft_text，保存后再运行。

```
python run.py day 17
```

预期结果：status 为 written，固定输出文件包含刚才审阅过的内容。

3. 重复请求，检查幂等

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

保持草稿和批准摘要完全不变，再次运行并查看结果。

```
python run.py day 17
python -m json.tool outputs/day17.json
```

预期结果：status 为 already_applied，重复执行没有再次追加内容。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 修改草稿，观察旧批准为什么失效

- 先执行 python run.py day 17，将 outputs/day17.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day17.draft_text 为 "Teaching report: this is a revised draft."，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 17。

- 对比两份结果，重点观察status、digest、preview 和 path；解释改变原因后恢复原设置或明确记录保留值。

观察什么：先完成一次批准写入，再只修改 draft_text 并保留旧 approved_digest；程序应拒绝把旧批准用于新内容。

机制解释：审批必须绑定到当前具体内容，不能因为用户曾经批准过一次就持续执行变化后的动作。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day17，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 · 验收

让批准绑定到具体内容，用幂等记录避免重复执行。 保留基线、修改结果与自己的解释。

成果文件：outputs/day17.json

☐ 未批准时不发生正式写入。

☐ 批准后写入成功，重复执行得到 already_applied。

☐ 修改草稿后旧摘要被拒绝，已记录审阅边界。

自测与解析

1. 修改待写内容后，之前的 approved_digest 应如何处理?

- A. 永久有效

- B. 自动替换为新摘要

- C. 需要重新审阅并批准

- D. 忽略摘要继续执行

答案：C。 旧批准只对应旧内容，新内容需要新的审阅。

2. already_applied 主要说明什么?

- A. 科学结论已被验证

- B. 相同操作已执行，重复请求没有再次写入

- C. 所有用户都拥有写权限

- D. 报告内容永远不能改

答案：B。 这是执行层面的幂等状态，不代表科学正确性或普遍授权。

小词典

- Digest：由内容计算的摘要，可用于检验内容是否改变。

- Idempotency：重复执行同一操作时避免重复产生效果的性质。

- Approval：用户对明确对象、内容与动作作出的批准。

04 · 我的记录

- 我的科研流程里哪一步值得在写入前显示预览？

- 如果批准后文件内容又改变，应该由哪一层阻止？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 17，打开 outputs/day17.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：hashlib：阅读 hashlib.sha256 的输入、digest 与 hexdigest。

https://docs.python.org/3/library/hashlib.html

MCP：2025-11-25 协议规范：阅读 Security and Trust & Safety 中的用户控制与工具安全。

https://modelcontextprotocol.io/specification/2025-11-25

下一步

下一课把工具连接放进真实的 stdio 消息往返中。

第18天 · 看见 MCP 的真实消息往返

观察本地进程如何初始化、列出工具并调用工具。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 运行真实 stdio 子进程通信。

- 区分初始化请求、通知、工具发现与工具调用。

- 说明教学子集与完整协议实现的区别。

开始之前

- 已经完成第 17 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

MCP统一连接方式，不替代工具实现

MCP把工具与数据接口组织成可协商的协议。主机应用、客户端连接器和服务端各有职责；服务端仍需实际执行检索或计算，协议不会自动让工具更正确。

本课启动真实的本地子进程，用标准输入输出传递 JSON-RPC 消息。你会看到 initialize、初始化完成通知、tools/list 与 tools/call。这比只打印一张流程图更接近协议的实际工作方式。

初始化先确认彼此支持什么

客户端不能一上来就假定对端支持所有功能。初始化会交换协议版本和能力，再通知对方初始化完成。本课手写教学子集使用2025-06-18协议版本，只覆盖课程需要的一条路径；可选官方SDK示例另按其支持版本协商，当前验收记录为2025-11-25。

不要把本课脚本称为完整 MCP SDK。协议还涉及错误、取消、传输、授权和不同版本的行为。阅读规范时先把实际出现的字段认出来，其余内容留作扩展清单。

标准输出是协议通道

在 stdio 连接中，stdout 不是普通的调试打印窗口。若服务端把“我启动了”这种日志写进协议通道，客户端可能把它当作 JSON 解析而失败。调试信息应与协议数据分开。

观察 requests、responses 和 stderr。请求ID帮助匹配响应；通知与请求不同，通常不要求对应响应。你应能够解释为什么消息数量不一定一一相等。

官方SDK适合后续扩展

主线使用标准库展示最小通信，便于在普通电脑复现。可选官方Python SDK示例能帮助理解正式客户端和服务端的生命周期。实践包锁定mcp==1.30.0，可运行python optional/mcp_sdk_demo.py；它与手写教学子集是两条独立练习路径。

SDK 的主分支文档可能已经迁移到新主版本。安装前先看 optional 说明，不能把新文档的类名和旧版本示例混用；是否实际运行成功要以自己的日志为准。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day18。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day18'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day18.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 18
```

预期结果：结果写入 outputs/day18.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查requests、responses 与 stderr。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day18.json
```

预期结果：能够定位requests、responses 与 stderr，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 通过工具调用参数控制返回数量

- 先执行 python run.py day 18，将 outputs/day18.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day18.top_k 为 1，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 18。

- 对比两份结果，重点观察requests、responses 与 stderr；解释改变原因后恢复原设置或明确记录保留值。

观察什么：重新运行后，tools/call 的参数和返回检索结果规模应反映 top_k=1；初始化和工具发现仍按原先顺序执行。

机制解释：传输协议与业务参数是不同层面，改变检索规模不需要改变整个连接流程。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day18，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

观察本地进程如何初始化、列出工具并调用工具。 保留基线、修改结果与自己的解释。

成果文件：outputs/day18.json

☐ 能在输出中识别初始化、工具列表与调用消息。

☐ 解释请求ID、通知和响应数量。

☐ 改变 top_k 后能指出对应的请求与结果变化。

自测与解析

1. stdio服务端应把普通调试日志写到哪里?

- A. 协议stdout中随意插入

- B. stderr等与协议消息分离的通道

- C. 藏进请求ID

- D. 覆盖客户端配置

答案：B。 stdout承载机器可解析消息，混入普通文字会破坏协议读取。

2. 运行本课最小示例成功能证明什么?

- A. 兼容所有MCP功能与版本

- B. 本课覆盖的有限消息流程可以运行

- C. 所有外部工具都安全

- D. 任意网络服务都不需授权

答案：B。 测试结论只能覆盖实际实现和执行的路径。

小词典

- stdio：通过进程标准输入输出交换消息的本地传输方式。

- JSON-RPC：带方法、参数和请求标识的远程过程调用消息格式。

- Capability：连接初始化时声明的功能支持情况。

04 · 我的记录

- 工具没有出现在列表里时，我先检查初始化还是模型提示词？

- 哪些教学实现的缺口在接入外部服务前必须补上？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 18，打开 outputs/day18.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

MCP：2025-11-25 协议规范：先读 Overview、Base Protocol 与消息生命周期，再查 tools/list、tools/call。
https://modelcontextprotocol.io/specification/2025-11-25

MCP：官方 Python SDK：对照官方SDK的服务器、客户端与版本迁移说明；按实践包锁定版本运行。
https://github.com/modelcontextprotocol/python-sdk

下一步

下一课将相同业务步骤表达为显式的状态图。

第19天 · 把现有流程迁移到 LangGraph

先理解状态、节点与边，再选择是否使用框架。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 把流程解释成状态、节点与边。

- 检查标准库主线的节点轨迹。

- 按选学说明运行真实LangGraph示例并分清覆盖范围。

开始之前

- 已经完成第 18 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

先认出流程，再引入框架

把“接收问题、检索材料、检查证据、输出回答”写成图，有助于看清每个阶段的输入与输出。节点负责局部工作，边决定下一步，状态负责在节点之间传递必要信息。

本课主线仍使用标准库运行同样的业务步骤。先对照 state 与 node_trace 理解流程，再看可选 LangGraph 示例；框架名不能代替对业务逻辑的解释。

状态应该包含完成任务所需的信息

状态中可以包含问题、检索结果、引用和错误状态。不要把所有日志、完整文件与无限聊天历史都堆进去；状态越大，复制、保存和排查问题越困难。

节点更新状态时应有明确合同。检索节点不应顺手完成批准写入；验证节点也不该静默修改来源。职责清楚才能单独测试，也方便以后替换模型或工具。

持久化与恢复需要明确定义

图框架可以提供检查点与持久化机制，但你仍要决定什么时候保存、恢复到哪里，以及重复运行可能产生什么副作用。第12课的恢复状态和第17课的幂等问题在这里依然存在。

本课选学示例的默认用途是展示真实 StateGraph API，不代表已经部署耐久服务或数据库。内存中的流程运行成功，与进程崩溃后仍可恢复，是不同的验收要求。

框架迁移也要保持评价方式

迁移前后使用同一问题与同一材料，比较引用、停止条件和输出结构。如果结果变了，先找状态流转或参数差异，而不是简单归因于框架更聪明。

optional/langgraph_demo.py 是独立示例，使用自己的固定查询，不自动读取 day19 的配置。主线参数实验只作用于 python run.py day 19，避免修改一处配置却期待另一个脚本改变。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day19。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day19'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day19.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 19
```

预期结果：结果写入 outputs/day19.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查state 与 node_trace。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day19.json
```

预期结果：能够定位state 与 node_trace，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 缩小检索结果，观察状态在节点间的变化

- 先执行 python run.py day 19，将 outputs/day19.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day19.top_k 为 1，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 19。

- 对比两份结果，重点观察state 与 node_trace；解释改变原因后恢复原设置或明确记录保留值。

观察什么：state.top_k反映新的上限；如果问题只命中一条材料，top_k从3降为1时证据数量可能不变。检查真实结果，并用node_trace追踪执行顺序。

机制解释：图结构控制步骤之间的关系，节点内部参数控制具体业务行为，二者应分开比较。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day19，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 · 验收

先理解状态、节点与边，再选择是否使用框架。 保留基线、修改结果与自己的解释。

成果文件：outputs/day19.json

☐ 能按 node_trace 还原执行顺序。

☐ 指出每个节点读写的状态字段。

☐ 明确区分主线实验、可选框架示例与未实现持久化。

自测与解析

1. 引入LangGraph后，谁负责定义工具与状态的业务含义?

- A. 框架自动推断全部含义

- B. 项目开发者

- C. 操作系统

- D. 任何网页文本

答案：B。 框架提供运行结构，业务合同仍由开发者设计与验证。

2. 修改day19配置后，独立的可选示例一定变化吗?

- A. 一定

- B. 只有它明确读取此配置才会变化

- C. 因为同一文件夹所以变化

- D. 只要安装依赖就变化

答案：B。 必须理解实际的数据读取路径，不能凭文件位置假定共享配置。

小词典

- State：任务执行过程中明确传递和更新的数据。

- Node：承担某一阶段工作的函数。

- Edge：决定节点之间执行关系的连接。

04 · 我的记录

- 我的流程哪里需要条件分支，哪里只需要固定顺序？

- 引入框架后，哪个行为必须保持不变才能公平比较？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 19，打开 outputs/day19.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 运行独立 LangGraph 示例

- 阅读 practice/optional 中的安装说明，在虚拟环境安装锁定的选学依赖。

- 运行 python optional/langgraph_demo.py，检查节点与最终状态。

- 说明它使用独立固定问题，没有读取 day19 配置；不要把这次结果算作真实模型推理。

参考资料与阅读范围

LangGraph：官方概览：阅读最小StateGraph示例与状态、节点、边的关系。

https://docs.langchain.com/oss/python/langgraph/overview

LangGraph：持久化：选读检查点、线程标识与恢复的基本概念。

https://docs.langchain.com/oss/python/langgraph/persistence

下一步

下一课把研究、质疑与复核拆成可以检查的角色职责。

第20天 · 让研究者、质疑者和复核者各司其职

用不同职责发现问题，避免把角色数量当作可靠性。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 解释研究者、质疑者、复核者的输入输出。

- 观察引用要求如何影响规则复核。

- 区分角色分工、独立证据与真实多模型调用。

开始之前

- 已经完成第 19 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

先按职责拆分，而不是按人数拆分

研究者整理候选回答与来源，质疑者寻找缺少依据的句子，复核者决定结果能否通过。每个角色应有可检查的输入与输出；三个角色重复同一句话并没有增加独立证据。

本课使用规则实现角色分工，结果中会明确标识。它展示责任划分与信息交接，不是同时运行三个真实大语言模型，也不代表模型之间已经完成独立评审。

质疑要指向具体证据缺口

有效的质疑是“这句话没有对应来源”或“引用只支持关联，不能支持因果”。只说“请更严谨”通常难以转化为下一步动作。把问题写成可复核条目，才能让研究者有针对性地补充。

机器规则只能发现预先定义的一部分错误，例如缺少引用；它不一定能判断引文是否真正支持结论。这里要保留人工复核入口，不把规则未报错解释成科学论证没有漏洞。

复核者需要公开的通过标准

require_citations 表示是否要求引用。这个参数应影响复核行为，而不是只改变显示文案。你可以比较开启和关闭时的反馈，并说明为什么科研任务通常应保留来源要求。

通过标准越宽松，通过率可能越高，但结果不一定更可靠。评估时应同时查看错误漏检与错误拒绝，不能把通过率当成唯一优化目标。

多角色会带来额外成本

真实模型版本中的多角色通常需要更多调用、更多等待和更多上下文。只有当职责分工能减少具体错误时，这些额外开销才有理由。简单任务可以使用单个明确流程。

同一模型反复扮演不同角色，仍可能共享相同偏差。后面评价课程会要求用外部可检查的案例与标准衡量表现，而不是让系统自己给自己打满分。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day20。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day20'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day20.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 20
```

预期结果：结果写入 outputs/day20.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查researcher、critic、reviewer 与 mode。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day20.json
```

预期结果：能够定位researcher、critic、reviewer 与 mode，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 关闭引用要求，审视通过标准是否变松

- 先执行 python run.py day 20，将 outputs/day20.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day20.require_citations 为 false，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 20。

- 对比两份结果，重点观察researcher、critic、reviewer 与 mode；解释改变原因后恢复原设置或明确记录保留值。

观察什么：对比 reviewer 的标准或检查项；部分基线本身已有引用，最终通过状态可能不变，这也是应记录的结果。

机制解释：参数改变不保证每个样例都改变最终标签；需要同时查看规则配置与中间信息，避免只盯通过或失败。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day20，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

用不同职责发现问题，避免把角色数量当作可靠性。 保留基线、修改结果与自己的解释。

成果文件：outputs/day20.json

☐ 能指出三种角色各自负责的检查。

☐ 比较修改前后要求与反馈，允许最终标签相同。

☐ 明确本课为规则角色示范，并保留科学人工复核。

自测与解析

1. 三个角色都同意一个结论，是否相当于三份独立证据?

- A. 是

- B. 只有用了不同名字才是

- C. 不是，可能共享来源和偏差

- D. 同意次数越多越确定

答案：C。 角色一致不能替代独立资料、方法或研究证据。

2. 参数变化后最终通过状态不变，首先应做什么?

- A. 认定参数没生效

- B. 查看检查条件、中间结果与样例是否触发差异

- C. 随意改其他参数

- D. 删除原始结果

答案：B。 某个样例可能同时满足两套标准，应检查执行路径再判断。

小词典

- Critic：按明确标准寻找缺口与反例的角色。

- Reviewer：依据验收标准决定是否进入下一步的角色。

- Correlated error：多个角色因共享模型或材料而产生的相关错误。

04 · 我的记录

- 哪类错误需要另一个角色，哪类错误用一个断言更直接？

- 我的复核标准会不会因为追求高通过率而被放松？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 20，打开 outputs/day20.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Anthropic：Building effective agents：阅读并行、评估者—优化者及何时使用复杂工作流。

https://www.anthropic.com/research/building-effective-agents

BioAgentLab：科研 Agent 教材：围绕评价与文献Agent章节，对照角色输出和证据台账。

https://github.com/zbyloveyj/BioAgentLab

下一步

下一课把已有的研究流程整合成带质量门的阶段作品。

第21天 · 阶段作品：带质量门的文献研究助手

把检索、报告与验收接起来，让缺少引用的结果停在发布之前。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 形成带引用和验收结果的阶段报告。

- 主动删除引用，验证质量门返回失败原因。

- 完成包含正常、异常和适用边界的短演示。

开始之前

- 已经完成第 20 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

阶段作品要有一条完整路径

一个可演示的作品至少要接收问题、检索资料、形成报告并返回可检查的状态。先把这些步骤接通，再考虑美化界面。你的演示对象需要知道问题是什么、材料来自哪里、什么条件下会停止。

本课把前面练习的证据检索与检查组合起来，产出 report 和 gate。主线使用固定小型教学材料与规则回答；报告能说明流程，但不能充当对真实论文全文的系统综述。

质量门应阻止不合格产物继续流转

质量门不是在报告末尾加一句“请核实”，而是把验收结果作为程序中的判断依据。缺少必要引用时，gate.valid 应当为 false，并列出具体错误。

单独检查引用存在仍然不够：存在的引用可能与论断不匹配，来源也可能过时。本课先验证一个能执行的最低标准，把语义支持、研究设计与偏差评估保留给人工审阅和后续扩展。

故意破坏输出，才能检查防线

如果每次都只运行正确样例，就不知道质量门遇到缺陷时是否真的工作。remove_citations 是一个受控开关，用于把候选报告中的引用移除，制造可重复的失败。

实验只改这个开关，其他问题与材料保持一致。对比报告正文和 gate.errors，你应当能解释为什么内容看起来流畅，程序仍应拒绝通过。失败案例是作品交付的一部分。

演示要包含成功、失败与边界

用两分钟讲清楚正常结果，再展示缺失引用导致的拒绝，最后说明它只能处理当前材料与规则覆盖的情况。这样的演示能让使用者知道何时可以信任流程，何时需要接管。

第14课作品验证最小闭环，第21课再增加编排与质量门。不要为了让演示一直成功而隐藏失败状态；记录一次明确拒绝，比只展示漂亮的最终段落更能说明实现。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day21。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day21'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day21.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 21
```

预期结果：结果写入 outputs/day21.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查report 与 gate.valid、gate.errors。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day21.json
```

预期结果：能够定位report 与 gate.valid、gate.errors，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 去掉引用，检验质量门是否真的拒绝

- 先执行 python run.py day 21，将 outputs/day21.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day21.remove_citations 为 true，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 21。

- 对比两份结果，重点观察report 与 gate.valid、gate.errors；解释改变原因后恢复原设置或明确记录保留值。

观察什么：候选报告的引用被移除，gate.valid 变为 false，errors 指出引用缺失；程序仍应保存可检查的实验结果。

机制解释：这是对质量门的反例测试。报告能生成与报告满足交付要求，是两项不同的判断。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day21，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

把检索、报告与验收接起来，让缺少引用的结果停在发布之前。 保留基线、修改结果与自己的解释。

成果文件：outputs/day21.json

☐ 基线报告保留能追溯的引用。

☐ 移除引用后出现明确失败结果与原因。

☐ 演示记录说明当前规则检查无法替代全文与科学审阅。

自测与解析

1. 质量门发现缺少引用后，最合理的行为是什么？

- A. 照常交付并隐藏错误

- B. 返回失败与具体缺口

- C. 自动捏造引用

- D. 只改报告标题

答案：B。 质量门要影响流转并给出可修复原因，不能只装饰输出。

2. 引用字段非空能够单独证明什么？

- A. 结论一定正确

- B. 论文没有偏差

- C. 只证明通过了引用存在性这一项检查

- D. 所有结论都得到来源支持

答案：C。 存在性检查的范围有限，语义支持与研究质量需要额外评价。

小词典

- Quality gate：决定产物能否进入下一步的明确检查。

- Negative case：故意不满足标准、用于验证系统拒绝行为的样例。

- Traceability：从结论追溯到材料、步骤与参数的能力。

04 · 我的记录

- 我的演示是否包含系统应该拒绝的情况?

- 当前质量门会漏掉哪一种看起来很可信的错误?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 21，打开 outputs/day21.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

BioAgentLab：科研 Agent 教材：定点阅读 book/zh/20_literature_agent.md 与 book/zh/18_evaluation_and_calibration.md，对照报告字段与验收。
https://github.com/zbyloveyj/BioAgentLab

Anthropic：Building effective agents：阅读 evaluator-optimizer 工作流，比较反馈如何影响下一步。
https://www.anthropic.com/research/building-effective-agents

下一步

下一课把数字计算交给可复核的工具，用合成数据练习科研分析。

第22天 · 用 CSV 工具完成一份可复核的数据摘要

把数值计算、缺失值规则与文字解释分开，避免凭语言猜数字。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 区分缺失值、零值与有效样本量。

- 运行真实CSV读取和确定性摘要计算。

- 用另一种缺失策略制造可解释的失败。

开始之前

- 已经完成第 21 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

先确认数据含义，再计算

本课读取 data/participants.csv，它是教学用合成数据，不能用于诊断或真实研究结论。先查看列名、单位和每一行，再确认 abundance 指的是本练习约定的数值。

同名列在不同项目中可能代表计数、比例或经过转换的特征。不能仅因为列名像相对丰度，就把任何数字解释为生物学效果；迁移到真实数据时必须补上数据字典与采集背景。

缺失不是零

缺失表示没有可用观测，零表示已经观测到数值为零。把空白默默当作零会改变平均值；排除缺失也会改变有效样本量，可能影响结果的代表性。

missing_policy=exclude 时，程序从统计中排除缺失值，同时报告 rows_total、rows_valid 与 rows_missing。error 策略则要求遇到缺失时停止。选择哪种策略，应当在分析之前说明理由。

让程序算，让人核对

均值、最小值与最大值由确定性代码计算。你可以手工抽查小数据：先求有效数值之和，再除以有效数量。工具输出要能沿着原始行复核，而不是只保留一段没有依据的解释。

语言模型可以帮助解释变量与写摘要，但不应凭记忆补算或填入未观察的数字。即使后续接入真实模型，也应把工具产出的精确数值与解释过程分开保存。

从教学样本迁移到自己的项目

你收藏的 CLEAR-Microbiome 与 SCUT_Minilab_MEF 可以提供研究问题和数据流程的参考，但本课没有下载、训练或复现它们的完整实验。先找一个公开的字段说明，尝试写出相应的工具合同。

真实科研分析还需要考虑样本独立性、批次、归一化、统计检验与隐私。一天一小时的目标是完成可追溯的描述统计，知道尚未覆盖什么，避免把小练习误称为完整分析平台。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day22。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day22'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day22.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 22
```

预期结果：结果写入 outputs/day22.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查analysis.rows_total、rows_valid、rows_missing、mean、min、max 和 synthetic_data。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day22.json
```

预期结果：能够定位analysis.rows_total、rows_valid、rows_missing、mean、min、max 和 synthetic_data，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 让缺失值成为必须处理的错误

- 先以missing_policy=exclude运行第22课，备份outputs/day22.json作为基线。

- 只把experiments.json中day22.missing_policy改为error并保存。

- 运行python run.py day 22 2> outputs/day22-error.json。程序遇到缺失或无效值后以退出码2结束；这是预期的拒绝。

- 打开outputs/day22-error.json检查具体原因。失败时day22.json不会重写，不能把旧文件误认成新结果；比较基线与错误记录，最后恢复exclude。

观察什么：默认合成表中abundance含缺失或无效值。error策略会停止统计，错误写到stderr且退出码为2；既有outputs/day22.json保持旧基线。用单独错误记录完成比较。

机制解释：排除和拒绝是不同的分析政策；改变政策前必须了解缺失如何影响样本量与解释。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day22，保存后重新运行；检查实际执行目录。

程序已报错，day22.json却仍显示旧平均值

原因：失败分支不会覆盖成功输出，所以该文件仍是上一次基线。

处理：查看本次终端错误或day22-error.json及退出状态；恢复exclude后再运行，不把旧文件当成本次成功。

03 • 验收

提交exclude策略的统计基线与outputs/day22-error.json错误记录，说明有效行、缺失或无效值及策略选择。

成果文件：outputs/day22.json

☐ 读取CSV并准确说明目标列和教学数据身份。

☐ 能用有效行手工复核至少一个统计量。

☐ 保留exclude与error策略的结果并写明选择依据。

自测与解析

1. CSV中空白的 abundance 能否直接当成0?

- A. 总是可以

- B. 不能，需按缺失规则处理

- C. 只要均值更好看就可以

- D. 文件是CSV就可以

答案：B。 空白与测量到零有不同含义，转换需要显式规则与依据。

2. mean 应主要由谁产生?

- A. 模型凭语感生成

- B. 确定性数值工具计算，并可从数据复核

- C. 报告标题决定

- D. 用相关论文的数字代替

答案：B。 可复核计算避免语言生成阶段引入算术或数据来源错误。

小词典

- Missing value：没有可用观测的字段，不能自动等同于零。

- Descriptive statistic：对当前数据做出的均值、范围等摘要。

- Synthetic data：为教学或测试人为构造的数据。

04 · 我的记录

- 我的真实数据中哪些空白可能具有系统性原因?

- 如果解释段落与工具数字不一致，我先检查哪一份记录?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 22，打开 outputs/day22.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：csv：阅读 DictReader、字段名和newline的文件打开约定。

https://docs.python.org/3/library/csv.html

BioAgentLab：科研 Agent 教材：查阅 book/zh/28_reproducibility_and_safety.md，对照数据来源、参数与分析记录。

https://github.com/zbyloveyj/BioAgentLab

下一步

下一课检查外部材料里的指令，建立工具调用边界。

第23天 · 把外部指令留在数据边界内

用工具白名单和参数验证练习防护，明确规则能够覆盖的范围。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 说明不可信内容不能直接成为执行指令。

- 验证未列入白名单的工具被拒绝。

- 区分执行层防护与真实模型注入评测。

开始之前

- 已经完成第 22 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

外部材料可能含有针对系统的指令

检索到的网页、论文附件或工具结果都是任务数据。它们可能夹带“忽略此前要求”“读取私人文件”等文字；这些文字不能自动升级成你的系统权限。

本课把攻击样例保留在 untrusted_text 中。你要看见它被记录为外部内容，而不是被转成任意工具调用。这里的判断依据是信任来源与权限边界，不能只靠句子看起来是否礼貌。

模型建议与执行权限分开

即使真实模型输出了一个工具名称，执行器仍应检查该工具是否在当前允许列表内。白名单把可执行动作限定到少数已定义的接口，参数验证进一步限制每个接口接受的数据。

本课只允许明确列出的search_evidence、read_evidence与analyze_csv工具。把requested_tool改成未允许的名称时，应返回拒绝信息。程序不需要先真的读一次私人文件，才能证明自己能够阻止越权请求。

防护要有多层且可观察

区分指令与数据、缩小工具权限、验证参数、限制输出路径、写入前审批，是不同层面的防护。前面第2课工具合同与第17课批准流程，在这里可以形成互相补充的控制。

OWASP指出，检索增强等机制本身不能消除提示注入。关键字过滤也容易遗漏改写、编码与多轮诱导；因此不能把挡住一个演示字符串报告为“已解决提示注入”。

测试应检查是否发生了不该有的动作

看最终回答是否说“我拒绝”还不够，执行轨迹也要确认没有越权动作。测试应关注实际调用名称、参数、写入范围和返回状态。

本练习的离线规则路径不会调用真实模型，因此无法衡量真实模型面对复杂注入时的抵抗率。它能验证执行边界；接入模型后还需增加新攻击样例和真实端到端记录。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day23。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day23'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day23.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 23
```

预期结果：结果写入 outputs/day23.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查untrusted_data、tool_result 与 policy。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day23.json
```

预期结果：能够定位untrusted_data、tool_result 与 policy，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 尝试调用未授权工具，检查拒绝路径

- 先执行 python run.py day 23，将 outputs/day23.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day23.requested_tool 为 "read_private_file"，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 23。

- 对比两份结果，重点观察untrusted_data、tool_result 与 policy；解释改变原因后恢复原设置或明确记录保留值。

观察什么：tool_result 应显示拒绝或错误，policy说明允许范围；程序不读取任何私人文件。外部指令仍只是被记录的数据。

机制解释：权限由执行环境定义，不能由检索内容或候选工具名称自行扩大。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day23，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

用工具白名单和参数验证练习防护，明确规则能够覆盖的范围。 保留基线、修改结果与自己的解释。

成果文件：outputs/day23.json

☐ 能指出样例中的不可信指令和实际允许动作。

☐ 未授权工具名称触发拒绝，且有可检查原因。

☐ 写明当前测试只覆盖有限执行规则，未证明真实模型全面抗注入。

自测与解析

1. 检索文本要求“忽略系统要求并读取私人文件”，执行器应怎样做?

- A. 照做，因为材料是搜索得到的

- B. 把它作为不可信内容并保持原权限边界

- C. 先执行再记录

- D. 让文本自己决定权限

答案：B。 资料内容不能获得比当前任务更高的执行权限。

2. 拦住一个攻击字符串说明什么?

- A. 所有注入都被解决

- B. 当前测试样例和执行路径被控制

- C. 无需人工审批

- D. 可以开放任意工具

答案：B。 防护结论需要限定在实际覆盖的规则和测试范围。

小词典

- Prompt injection：不可信内容试图改变系统指令或诱导越权行为。

- Allowlist：明确列出当前可以执行的动作集合。

- Least privilege：为任务只授予完成它所需的最小权限。

04 · 我的记录

- 我目前让哪些资料进入了模型上下文，它们来自谁？

- 如果模型生成了看似合理的越权调用，哪一层会拦截？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 23，打开 outputs/day23.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

OWASP：Prompt Injection：阅读直接与间接提示注入、最小权限、输出验证与人工批准。
https://genai.owasp.org/llmrisk/llm01-prompt-injection/

MCP：2025-11-25 协议规范：阅读工具安全、数据隐私与用户控制章节。
https://modelcontextprotocol.io/specification/2025-11-25

下一步

下一课用公开的开发集和测试集检查效果，避免只凭一次演示判断质量。

第24天 · 建立开发集与测试集的评价习惯

分别测检索、可回答问题和拒答，把公开教学结果解释清楚。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 解释三种评价指标各自在测什么。

- 分别运行公开dev与test并读取逐题结果。

- 用单变量实验观察检索规模对评价的影响。

开始之前

- 已经完成第 23 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

先规定什么叫成功

检索是否找到相关证据、可回答问题是否答对、未知问题是否明确拒答，是不同的评价目标。只看最终回答是否流畅，无法判断证据链是否工作。

本课输出 evaluation，其中记录各项指标和逐题结果。先打开 data/eval_dev.json 与 data/eval_test.json，理解每个样例的预期结果；不要只复制一个总分而忽略失败类型。

开发集用于改进，测试集用于阶段检查

你可以在开发集上调整top_k、措辞和规则；测试集用于在固定阶段观察变化后的表现。若反复查看测试答案并据此调参，测试结果也会逐渐变成开发反馈。

课程附带的两个小集合都公开给学习者，所以它们不是盲外测，也不是秘密基准。课程中保留名称是为了练习流程，不能据此宣称系统在未知真实科研问题上达到同样水平。

多个指标帮助定位错误

retrieval_recall关注相关证据是否被找到；answerable_accuracy关注有依据的问题；abstention_accuracy关注系统是否会对材料外问题保留不确定。三个数可能朝不同方向变化。

扩大top_k可能找到更多相关材料，也可能加入噪声；当前词法检索与规则回答有自己的特征。用逐题结果说明变化，而不是假设参数越大越好或平均分越高就全面改进。

小数据集适合发现问题，不适合夸大能力

几道教学题能快速发现回归错误，但分数对单题变化很敏感。还要考虑样例重复、资料泄漏和问题覆盖范围，才能判断评估是否代表真实任务。

本课的产物应包括本次参数、数据集合、逐题失败原因和下一步补充的样例类型。未来加入真正保留的外部题目时，应冻结题目与标准，并控制根据测试结果继续调参的次数。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day24。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day24'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day24.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 24
```

预期结果：结果写入 outputs/day24.json；主线 mode 明确为 offline，代表规则教学流程。

3. 分别运行开发集与测试集

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

运行两条评价命令，保存终端结果，并分别检查split、top_k、三项指标与items。它们都是公开教学题；找到至少一项逐题判断并解释其依据。

```
python run.py evaluate --split dev --top-k 3
python run.py evaluate --split test --top-k 3
```

预期结果：两份结果标注各自集合与参数，可逐题检查；分数不能作为盲外测结论。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 缩小检索规模，比较哪些题受到影响

- 先执行 python run.py day 24，将 outputs/day24.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day24.top_k 为 1，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 24。

- 对比两份结果，重点观察evaluation 的指标、样例数与逐题 items；解释改变原因后恢复原设置或明确记录保留值。

观察什么：在相同split上比较top_k=3与1的逐题结果；可能有题变化，也可能整体分数不变，必须按真实输出记录。

机制解释：小样本和规则流程可能使某些参数变化不影响最终标签。逐题观察比预设“分数一定下降”更可靠。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day24，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

分别测检索、可回答问题和拒答，把公开教学结果解释清楚。 保留基线、修改结果与自己的解释。

成果文件：outputs/day24.json

☐ 分别保存开发集与测试集结果，标注参数。

☐ 至少解释一题成功或失败的原因。

☐ 报告明确写出公开小样本、非盲外测的限制。

自测与解析

1. 不断查看test答案并据此调参后，test结果应怎样解释？

- A. 仍然完全独立

- B. 已受到开发过程影响，独立性降低

- C. 题目越少越独立

- D. 可以称为盲外测

答案：B。 测试反馈参与了优化，它不再提供同等独立的验证。

2. 所有问题都回答一段话，可能损害哪个指标？

- A. 未知问题的拒答准确性

- B. 只会提高所有指标

- C. 与评价无关

- D. 自动提升证据召回

答案：A。 对材料外问题也给出确定回答，可能牺牲拒答能力。

小词典

- Development set：用于调试和改进系统的样例集合。

- Test set：用于阶段性检查的样例集合；独立程度取决于使用方式。

- Abstention：证据不足时明确停止推断或拒绝给出确定结论。

04 · 我的记录

- 我更在意漏掉可回答问题，还是对未知问题乱答，为什么？

- 现有题目遗漏了我的真实任务中的哪一种失败？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 24，打开 outputs/day24.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

BioAgentLab：科研 Agent 教材：阅读 book/zh/18_evaluation_and_calibration.md 与 book/zh/31_benchmark_and_failure_cases.md，重点看指标与失败分类。
https://github.com/zbyloveyj/BioAgentLab

Lewis 等：Retrieval-Augmented Generation：阅读检索增强方法的任务设置与评价部分，对比本课词法检索的范围。
https://arxiv.org/abs/2005.11401

下一步

下一课把耗时、调用与成本记录下来，让运行情况可解释。

第25天 · 记录轨迹、耗时与成本边界

用可观察字段排查流程，把未知用量和费用明确记为未知。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 沿可观察事件解释一次执行路径。

- 区分真实耗时、token报告与费用估算。

- 让步骤预算改变运行，并解释停止行为。

开始之前

- 已经完成第 24 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

轨迹记录动作，不展示隐藏思维链

本课程的trace包含可观察的节点、工具调用与结果状态，帮助回答“检索了什么、在哪一步停止”。这些是程序事件，不是模型内部完整的推理过程。

排查错误时从输入、参数、工具返回和输出验证逐层定位。不要为了得到看起来详细的解释而要求模型编造内部思维；执行日志应当来自真实发生的操作。

耗时需要测量，不能预先写死

observability中的耗时来自当前实际运行，受机器负载、输入规模与执行路径影响。两次运行数字不同并不一定是错误，重要的是测量范围与单位一致。

离线词法检索的耗时无法代表真实模型或网络服务性能。迁移到真实provider后，还要区分等待、模型生成、工具执行和重试耗时；比较时保持任务与环境尽量一致。

未知用量不应该伪装成零

离线规则流程没有提供模型token报告时，相关字段应为null；价格不可确定时cost_usd也应为null。零费用是一个有含义的数值判断，不能用它代替“未测量”或“未定价”。

真实服务返回的usage也需检查定义，包括输入、输出、缓存等分类。估算费用必须同时记录适用模型、单价来源、计价日期和公式；本课不把可能变化的价格写成永久常量。

日志足够排错，也要控制信息范围

记录问题、引用ID和状态能帮助复现，但真实项目还要考虑数据最小化与保留时间。凭证、私人原文和敏感数据不能因为“方便排错”就自动进入可分享日志。

本课使用教学材料。你可以学习如何将调用预算max_steps与实际停止原因对照；后续接入真实模型时，先检查哪些字段会被发送和保存，再扩展日志。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day25。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day25'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day25.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 25
```

预期结果：结果写入 outputs/day25.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查answer.status、answer.trace 与 observability。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day25.json
```

预期结果：能够定位answer.status、answer.trace 与 observability，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 缩小调用步骤预算，观察停止记录

- 先执行 python run.py day 25，将 outputs/day25.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day25.max_steps 为 1，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 25。

- 对比两份结果，重点观察answer.status、answer.trace 与 observability；解释改变原因后恢复原设置或明确记录保留值。

观察什么：步骤预算降低后，应在有限步骤内停止，并在answer状态或轨迹中反映预算限制；耗时按实际结果记录，不预设一定更快。

机制解释：步骤预算控制流程是否能走完。只有观察到真实执行路径，才能解释输出与耗时的变化。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day25，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 · 验收

用可观察字段排查流程，把未知用量和费用明确记为未知。 保留基线、修改结果与自己的解释。

成果文件：outputs/day25.json

☐ 能够指出一次实际执行的输入、动作和停止原因。

☐ 未知token或cost保持null，并解释null与0的区别。

☐ 用基线与小预算结果说明过程变化，不宣称真实模型已经测通。

自测与解析

1. 没有可确认的模型单价时，cost_usd最合适的值是什么？

- A. 0，代表免费

- B. 随便估一个数

- C. null，并说明原因

- D. 上一家服务的价格

答案：C。 null表示未知或不适用，避免把未测量结果误写成确定费用。

2. 课程trace表示什么？

- A. 模型内部完整思维链

- B. 真实记录的可观察执行事件

- C. 模型所有隐藏状态

- D. 答案正确性的充分证明

答案：B。 程序事件能帮助复现和排错，但不等于内部思维，也不能单独证明结论正确。

小词典

- Observability：通过事件、指标与日志了解系统实际运行状态的能力。

- Usage：服务返回的资源用量记录，其字段定义随接口而异。

- Null：没有可用值或不适用的显式表示，不等同于数值零。

04 · 我的记录

- 如果一次回答很慢，我需要哪些记录才能定位原因?

- 我能否说明当前报告里的每一个费用数字来自哪里?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 25，打开 outputs/day25.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：logging：阅读记录级别、格式与处理器，思考哪些字段值得记录。
https://docs.python.org/3/library/logging.html

OpenAI：Responses API：按需阅读usage字段；以当前服务实际返回值为准。
https://platform.openai.com/docs/api-reference/responses

下一步

下一课给同一流程加上本地网页，保持输入输出合同。

第26天 · 给研究助手加一个本地网页

把网页请求接到同一套工具合同，练习从界面到结果的完整链路。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 区分day命令生成协议样例与serve启动真实网页。

- 打开loopback网页并提交一个课程内问题。

- 解释请求验证、本地范围与离线能力边界。

开始之前

- 已经完成第 25 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

界面负责收集输入，服务负责执行

浏览器提供问题输入框与结果展示，Python服务接收请求并调用已有流程。这样界面变化时，检索、预算和验收规则仍可以在同一个后端复用。

python run.py day 26 只演示请求与响应的结构，生成可检查JSON；它不会持续启动网页服务器。要亲手打开网页，必须另运行serve命令，并让该终端保持运行。

本地地址有明确的范围

本课服务绑定127.0.0.1，也称loopback，默认只能由本机访问。打开http://127.0.0.1:8765后，网页调用的是这台电脑上的教学服务。

这不是公网部署、多人账户系统或手机跨设备访问方案。Python标准库HTTP服务适合本地练习；真实上线需要另外处理认证、并发、配置、监控和托管，不能把本地启动成功等同于生产发布。

前端检查不能代替服务端验证

输入框可以提示用户补全问题，但请求也可能绕过网页直接发送。服务端仍要检查字段、类型和长度，并给出可理解的错误。两层检查分别服务于体验和执行边界。

本课沿用问题与工具参数的合同，不让网页随意指定文件路径或工具名称。看到错误时先区分端口连接失败、请求格式错误和业务没有证据，避免把三种情况都写成“网络问题”。

保持离线模式与凭证边界清楚

本地网页走课程的离线教学流程，不在浏览器里配置云模型API key。用户看见的答案应保留来源和边界提示，不能因为界面更完整就让能力看起来超出后端实现。

未来接入真实模型时，凭证应由受控服务端管理，并明确告知哪些内容会发送给模型服务。本课程的网页验收只证明本地输入、执行与结果显示可以连通。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day26。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day26'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day26.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 26
```

预期结果：结果写入 outputs/day26.json；主线 mode 明确为 offline，代表规则教学流程。

3. 启动真实本地服务

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

打开第二个终端，进入practice目录后运行下方命令。保留终端，在浏览器打开 http://127.0.0.1:8765 ，输入课程内问题并提交；记录回答、引用与状态。完成后在服务终端按Ctrl+C停止。

```
python run.py serve --port 8765
```

预期结果：本地网页可以提交问题并显示后端结果；服务器运行期间终端保持占用是正常现象。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 更换问题，观察请求和引用如何对应

- 先执行 python run.py day 26，将 outputs/day26.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day26.question 为 "What does relative abundance measure?"，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 26。

- 对比两份结果，重点观察request、response 与 local_url；解释改变原因后恢复原设置或明确记录保留值。

观察什么：request包含新问题，response中回答与引用按新的检索结果生成；本地服务地址保持不变。以实际证据ID核对，避免只看页面标题。

机制解释：输入经过同一套后端流程，正确的接口应让问题变化影响业务结果，而不是只替换界面文字。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day26，保存后重新运行；检查实际执行目录。

浏览器提示无法连接或端口被占用

原因：serve没有启动、终端已结束，或8765正在被其他程序使用。

处理：在独立终端运行serve并保持开启；若端口占用，可指定其他端口，并打开相应的本机地址。

03 • 验收

把网页请求接到同一套工具合同，练习从界面到结果的完整链路。 保留基线、修改结果与自己的解释。

成果文件：outputs/day26.json

☐ 保存第26课请求响应JSON，并指出关键字段。

☐ 单独启动serve，在浏览器实际提交一次问题。

☐ 说明服务仅限本地教学、离线流程，未宣称公网或真实模型能力。

自测与解析

1. 执行python run.py day 26以后，为什么网页未必能打开?

- A. 课程缺少全部代码

- B. 该命令只生成样例，需另启动serve

- C. 必须把API key粘到网页

- D. 必须关闭本地网络

答案：B。 一次性命令和持续服务是两种运行方式，需明确区分。

2. 前端已经检查问题非空，后端是否还要验证?

- A. 不需要

- B. 需要，客户端请求可能绕过网页

- C. 只要页面漂亮就不需要

- D. 只有使用模型才需要

答案：B。 执行边界必须由服务端守住，界面验证不能覆盖所有请求来源。

小词典

- Loopback：指向本机的网络接口，常见地址为127.0.0.1。

- Request/response：客户端发送输入，服务端返回结果的交互方式。

- Backend validation：在实际执行请求的服务端检查参数与权限。

04 · 我的记录

- 遇到页面打不开，我如何区分服务未启动和业务出错?

- 哪个界面提示能帮助使用者理解当前结果的边界?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 26，打开 outputs/day26.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：http.server：阅读HTTPServer与BaseHTTPRequestHandler，并注意标准库服务的适用范围。
https://docs.python.org/3/library/http.server.html

BioAgentLab：科研 Agent 教材：对照工具合同与可复现章节，检查网页是否改变原有输入输出约束。
https://github.com/zbyloveyj/BioAgentLab

下一步

下一课比较缓存、并发与调用预算，观察实际调用次数。

第27天 · 用缓存与并发控制重复工作

通过实际调用数观察优化效果，同时保留预算和结果正确性。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 比较重复查询在缓存开关前后的实际调用数。

- 解释并发数与调用总预算的不同含义。

- 指出缓存正确性与多用户隔离的后续要求。

开始之前

- 已经完成第 26 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

缓存保存的是特定条件下的结果

同一个查询重复出现时，可以复用已计算的检索结果，减少重复工作。缓存键应能代表影响结果的条件，例如查询、检索参数和材料版本。

本课的缓存是一次教学批处理中的本地机制。它不是跨机器长期缓存，也没有实现多用户敏感数据的完整隔离。未来扩展时，材料更新、用户身份和过期策略都可能成为缓存键或失效规则的一部分。

并发适合相互独立的工作

多个互不依赖的查询可以同时检索，max_workers限制并发工作者数量。若后一步依赖前一步结果，就不能只为了加速而把它们并行执行。

并发会增加共享状态的复杂度，例如两个相同查询同时到达时，是否重复调用或争用缓存。应通过actual_calls与cache_hits观察当前实现的行为，而不凭代码里出现线程池就断言更快。

调用预算约束资源消耗

max_calls限制实际计算或调用次数。缓存命中通常可减少实际调用，但预算耗尽后必须返回明确状态，不能偷偷突破上限来凑齐结果。

本练习是小型词法检索，单次工作很轻，线程开销可能抵消并发收益。课程重点是理解资源控制与可观察行为，不保证某台电脑上的耗时一定下降。

优化之后仍要检查结果

优化应保留查询与结果的对应关系，不能因为并发完成顺序不同就把答案配错问题。对重复查询还要检查引用和证据ID的一致性。

现实应用中，缓存可能保存过时资料；不同用户的相同问题也可能有不同权限。将这些限制写进扩展计划，再决定是否需要持久缓存、租户隔离、过期与失效机制。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day27。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day27'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day27.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 27
```

预期结果：结果写入 outputs/day27.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查results、actual_calls、cache_hits 与 budget_exhausted。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day27.json
```

预期结果：能够定位results、actual_calls、cache_hits 与 budget_exhausted，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 关闭缓存，观察重复查询是否重复执行

- 先执行 python run.py day 27，将 outputs/day27.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day27.cache 为 false，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 27。

- 对比两份结果，重点观察results、actual_calls、cache_hits 与 budget_exhausted；解释改变原因后恢复原设置或明确记录保留值。

观察什么：保持默认的三个查询和max_calls=3不变；关闭缓存后重复问题不再命中缓存，实际调用次数应相应增加，结果仍应对应原查询。

机制解释：单变量比较把重复计算带来的差异与并发数量、调用预算分开，避免混淆优化原因。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day27，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 · 验收

通过实际调用数观察优化效果，同时保留预算和结果正确性。 保留基线、修改结果与自己的解释。

成果文件：outputs/day27.json

☐ 能从查询列表找出重复输入。

☐ 保存缓存开启与关闭的调用数及命中数。

☐ 检查结果对应关系，并写出一个缓存失效或隔离风险。

自测与解析

1. max_workers=2与max_calls=3分别约束什么？

- A. 都表示总任务数

- B. 前者是并发数量，后者是实际调用预算

- C. 前者是缓存天数，后者是模型参数

- D. 都表示费用

答案：B。 同时做多少与累计做多少是两项不同控制。

2. 缓存命中率更高是否总能说明结果更好？

- A. 总是

- B. 不一定，还要检查过期、权限与正确性

- C. 只要快就好

- D. 命中就不需要来源

答案：B。 缓存可能返回过时或权限不合适的结果，效率指标不能替代正确性检查。

小词典

- Cache hit：请求复用了已有缓存结果。

- Concurrency：多个任务在时间上重叠执行。

- Call budget：一次任务允许产生的实际调用上限。

04 · 我的记录

- 我的任务里哪些步骤独立，哪些必须等待上一步？

- 资料更新以后，我怎样避免继续使用旧缓存？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 27，打开 outputs/day27.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：concurrent.futures：阅读ThreadPoolExecutor、submit与结果收集，观察独立工作如何并发。
https://docs.python.org/3/library/concurrent.futures.html

BioAgentLab：科研 Agent 教材：结合状态与记忆章节，思考数据生命周期和隔离。
https://github.com/zbyloveyj/BioAgentLab

下一步

下一课为源码、数据与依赖建立可复现的交付清单。

第28天 · 把作品整理成别人能复现的项目

记录真实文件摘要、运行入口与验证步骤，让交付有据可查。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 生成真实项目文件与摘要清单。

- 解释可复现材料、运行产物和凭证的区别。

- 用明确命令说明如何复现与验证当前能力。

开始之前

- 已经完成第 27 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

可复现从明确输入开始

别人需要知道使用了哪份代码、哪份数据、哪些参数和怎样运行。只发送最终截图，会丢失这些条件；只发送源码，也可能缺少数据与使用说明。

本课生成manifest，列出当前实践包中被纳入清单的实际文件及摘要。先检查清单，再把README中的环境、命令与产物位置和实际目录对照。

摘要帮助发现文件变化

SHA-256摘要由文件字节计算。文件内容变化通常会改变摘要，因此可用于检查两份交付材料是否一致。路径和摘要一起记录，能帮助定位发生变化的文件。

摘要不代表文件安全、科学正确或来源可信；它也不能代替版本控制的历史说明。把它当作一致性检查工具，与来源、许可和审阅记录配合使用。

把凭证和运行产物排除在交付之外

本课清单应针对源码、数据与必要说明，不把outputs、.env和本地凭证一并打包。运行产物可能包含用户输入，也可能随每次运行变化，不适合作为所有人共享的固定项目输入。

需要展示样例结果时，应选择明确可分享的教学样例并解释其来源。不能为了让压缩包“看起来完整”而把整个工作目录不加筛选地复制出去。

验证应对应实际承诺

include_tests决定清单是否包括测试文件。测试能覆盖输入验证、状态分支和失败行为，但通过测试不代表模型质量或科研结论已被充分评估。

完成交付前，按README在新的终端里运行一个主线案例，并检查依赖是否为必需或选学。真实模型、官方SDK与LangGraph选学路径要各自记录是否真正执行过，不能用离线测试代替它们的验证。

02 · 一步一步做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day28。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day28'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day28.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 28
```

预期结果：结果写入 outputs/day28.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查manifest.files、manifest.sha256 与 instructions。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day28.json
```

预期结果：能够定位manifest.files、manifest.sha256 与 instructions，并用自己的话解释该结果。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 移除测试文件，观察交付清单发生什么变化

- 先执行 python run.py day 28，将 outputs/day28.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day28.include_tests 为 false，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 28。

- 对比两份结果，重点观察manifest.files、manifest.sha256 与 instructions；解释改变原因后恢复原设置或明确记录保留值。

观察什么：清单不再包含测试文件，而核心源码与数据保持在清单中；对应集合摘要随纳入文件集合而变化。以实际输出字段为准。

机制解释：交付内容取决于显式纳入规则。改变范围应可被清单与摘要反映，不能只改变说明文字。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day28，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

记录真实文件摘要、运行入口与验证步骤，让交付有据可查。 保留基线、修改结果与自己的解释。

成果文件：outputs/day28.json

☐ 能找到至少一个源码和一个数据文件的真实摘要。

☐ 确认清单没有outputs、.env或凭证文件。

☐ 运行说明包含环境、入口和验证边界，选学状态如实记录。

自测与解析

1. 文件的SHA-256摘要主要帮助检查什么?

- A. 科学结论是否正确

- B. 文件内容是否一致或改变

- C. 软件一定无漏洞

- D. 论文一定原创

答案：B。 摘要是内容一致性工具，不是质量、来源或安全保证。

2. 为什么不应把整个outputs默认并入通用交付?

- A. 它永远无用

- B. 可能含运行输入、变化结果或敏感内容，需要筛选

- C. 文件名不够漂亮

- D. Python不能读取它

答案：B。 交付应根据用途与内容选择，避免把不必要的运行记录传播出去。

小词典

- Manifest：记录纳入交付的文件、版本或摘要的清单。

- Reproducibility：在明确条件下复现同一流程与结果的能力。

- Artifact：运行或创作产生的文件、报告等产物。

04 · 我的记录

- 如果另一个人只有这份项目，最可能卡在哪一步？

- 我说“已测试”时，是否能给出具体命令与覆盖范围？

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 28，打开 outputs/day28.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

Python：hashlib：阅读sha256的字节输入和hexdigest，用小文件理解摘要。

https://docs.python.org/3/library/hashlib.html

Python：unittest：阅读测试发现与断言，核对测试覆盖了哪些真实行为。

https://docs.python.org/3/library/unittest.html

BioAgentLab：科研 Agent 教材：阅读 book/zh/28_reproducibility_and_safety.md，列出复现需要的条件。

https://github.com/zbyloveyj/BioAgentLab

下一步

下一课集中检查未知问题、矛盾证据与攻击样例，形成失败案例册。

第29天 · 为未知、冲突与攻击建立失败案例册

让系统显露盲区，把应当拒绝和需要人工判断的情况留作交付证据。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 识别材料外问题被无关证据误匹配的真实失败。

- 复现有限冲突规则的触发与未触发。

- 记录一个当前规则不能识别的反例。

开始之前

- 已经完成第 28 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

未知问题应该有明确出口

课程材料没有涵盖某个问题时，系统应说明证据不足。默认问题询问未知疾病的治疗，这不在教学语料能够支持的范围内，因此不应产生确定的治疗建议。

默认运行会暴露一个真实失败：unknown一词命中E08中“未知token与成本”的笔记，status可能是answered，却没有提供关于疾病治疗的相关证据。应把它记为未知识别失败，不能将非空回答当成回答正确。再用完全无匹配的问题对照，才能看见当前拒答仅覆盖部分情况。

证据冲突需要先识别再解释

两份材料可能给出不同结论，原因可能是研究对象、方法、时间或定义不同。不能只保留排在第一位的来源，也不能简单投票决定哪个结论正确。

本课conflict_review使用有限关键字与规则触发提醒，帮助你看见冲突检查的位置。它不是语义矛盾检测器，不能可靠识别隐含否定、数值差异、跨句推断或研究设计差异。

没有告警不等于没有冲突

include_conflict控制是否加入课程预设的冲突案例。关闭后告警可能消失，但你只能得出预设触发条件没有出现，不能推断所有资料在科学意义上已经一致。

写一个当前规则会漏掉的反例，例如“改善10%”与“未达到预设阈值”可能需要上下文才知道是否矛盾。把反例留在案例册里，明确它尚未被自动化覆盖。

失败案例比一句总评更有用

将未知问题误匹配、冲突提醒和攻击检查分别记录输入、预期、实际结果与是否需要人工接管。失败类型越具体，后续改进越容易对应到检索、工具、提示或审核机制。

本课与第23课同样使用离线规则，不声称完成真实模型红队测试。演示时要同时说明挡住了什么、没测什么、当前规则会漏什么，让使用者理解接管条件。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day29。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day29'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day29.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 29
```

预期结果：结果写入 outputs/day29.json；主线 mode 明确为 offline，代表规则教学流程。

3. 识别误匹配，再对照零命中问题

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

检查默认unknown_answer：unknown一词可能命中E08，但该笔记没有疾病治疗证据，应记为失败。随后运行下方无匹配问题，比较状态与引用；这只展示零命中时的拒答，不能修复默认样例的误匹配。

```
python -m json.tool outputs/day29.json
python run.py ask "zxqv nebula recipe"
```

预期结果：默认样例出现无关E08摘录；对照问题为abstained。把两者记录成当前检索与拒答规则的边界。

4. 改变一个条件并比较

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

完成下面“改一改”实验。对比基线与新结果，记录哪些字段改变、哪些保持稳定；不要同时改模型、材料和参数。

预期结果：留下一组前后结果和一句解释；失败状态也可以是正确的实验产物。

改一改 · 移除预设冲突样例，比较告警变化

- 先执行 python run.py day 29，将 outputs/day29.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day29.include_conflict 为 false，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 29。

- 对比两份结果，重点观察unknown_answer、conflict_review 与 attack_check；解释改变原因后恢复原设置或明确记录保留值。

观察什么：预设冲突触发条件不再出现时，conflict_review的结果相应变化；unknown_answer与独立攻击检查的职责仍分开。不要将无告警解释为语义上已无冲突。

机制解释：单变量实验只能说明当前规则对预设样例的反应，无法外推出对所有矛盾表达的识别能力。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day29，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

让系统显露盲区，把应当拒绝和需要人工判断的情况留作交付证据。 保留基线、修改结果与自己的解释。

成果文件：outputs/day29.json

☐ 记录默认问题误匹配E08的失败，不把answered标签当作治疗证据。

☐ 记录冲突检查前后结果及触发范围。

☐ 案例册包含一个未覆盖反例和清楚的人工接管条件。

自测与解析

1. 有限规则没有发现冲突，可以得出什么结论?

- A. 所有证据一致

- B. 本次输入没有触发当前规则

- C. 研究设计完全相同

- D. 无需阅读原文

答案：B。 规则覆盖面有限，无告警只能按实际检测条件解释。

2. 面对材料外治疗问题，最合理的输出是什么?

- A. 猜测一种疗法

- B. 说明证据不足并停止确定推断

- C. 引用任意无关材料

- D. 为了完整回答编造研究

答案：B。 证据边界应优先于回答完整度，尤其不能从教学语料推导不存在的医疗证据。

小词典

- Failure case：输入、预期和实际行为被明确记录的失败样例。

- Conflict review：对可能不一致的证据进行标记与复核的过程。

- Blind spot：当前实现或评估没有覆盖到的情况。

04 · 我的记录

- 哪个看起来很自然的问题最可能绕过我的拒答规则?

- 没有告警时，我还需要人工检查哪些证据关系?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 29，打开 outputs/day29.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把本课机制迁移到一个自己的任务

- 从自己的文献整理或数据分析流程中选一个相似环节。

- 写清输入、允许动作、失败出口与验收方法。

- 提出一个会让当前实现失败的反例，并标明它是待实现扩展。

参考资料与阅读范围

BioAgentLab：科研 Agent 教材：阅读 book/zh/31_benchmark_and_failure_cases.md，按未知、冲突、工具错误和攻击分类记录。

https://github.com/zbyloveyj/BioAgentLab

OWASP：Prompt Injection：对照攻击测试和防护局限，区分规则演示与真实模型评估。

https://genai.owasp.org/llmrisk/llm01-prompt-injection/

下一步

最后一课完成可复现演示、能力卡与下一阶段学习计划。

第30天 · 毕业交付：可复现的科研学习 Agent

交付作品、运行证据和诚实的能力卡，把一个月的学习变成可继续改进的项目。

标准路线：60分钟；忙碌日：20分钟；扩展练习另计。

今天的目标

- 运行结业入口并解释各项结果。

- 整理三个可复现案例与真实能力边界。

- 确定一个有数据、工具合同和验收标准的后续任务。

开始之前

- 已经完成第 29 课，或能解释其产物。

- 打开实践包 practice 目录；后文 python 可替换为虚拟环境内解释器的完整路径。

一小时安排

- 15分钟 · 理解概念：阅读本课四节讲解，弄清一个关键机制与适用边界。

- 30分钟 · 亲手操作：先跑基线，再做一次单变量实验，保留前后结果。

- 10分钟 · 检查结果：逐项验收，完成两道自测并阅读解析。

- 5分钟 · 留下记录：写出本课发现、未解决问题和下次第一步。

01 · 理解概念

毕业作品是一条可解释的完整流程

用一个材料内问题启动流程，展示检索依据、回答、引用、停止条件与评价结果。别人应当能够按README复现，而不只是观看你的截图。

本课把answer、evaluation、capability_card和project_checks集中到一个结果里。它是教学作品的结业入口，主要证明离线规则链路与工程机制；真实模型路径需要单独运行并记录结果。

能力卡必须说明实现与边界

列出当前支持的输入、证据来源、工具、运行模式、失败行为及未覆盖内容。规则词法检索不能写成向量语义检索；规则角色分工不能写成多个模型独立研究；本地网页不能写成公网部署。

可选Ollama、OpenAI、MCP官方SDK与LangGraph分别有自己的依赖、配置和验证范围。没有实际执行过的路径，应标为未验证；不能因为示例文件存在或接口形状正确就声称已经端到端跑通。

用三个案例说明作品质量

第一个案例展示材料内的正常回答，第二个展示未知或缺失引用时的拒绝，第三个展示一次预算限制、缺失值或批准失效的受控失败。每个案例保留输入、参数、结果和一句解释。

评价集合是公开的小型教学题。报告分数时附上样例规模和设置，不宣传为真实科研外测；错误仍然存在时，明确说明由谁复核、在什么条件下停止使用自动结论。

下一阶段只选一个真实问题

从BioAgentLab或你收藏的CLEAR-Microbiome、SCUT_Minilab_MEF中选一个具体任务，例如公开文献证据整理或公开CSV摘要。先写数据字典、工具合同和十个评价样例，再考虑新增框架与模型。

课程没有复现收藏项目的训练流程，也没有把全部收藏下载成为知识库。你已经完成的是一套可以迁移的方法和可运行教学底座；下一阶段继续保持每天约一小时，每周保留复盘与缓冲。

02 · 一步一做

命令在解压后的 practice 文件夹执行。Windows可使用 .\.venv\Scripts\python.exe；macOS/Linux可使用 .venv/bin/python 替代 python。

1. 找到本课的输入与配置

为什么做：先知道程序读什么，后面才能解释输出为何改变。

在 VS Code 中打开 experiments.json，定位 day30。用下面的命令确认当前目录正确，并把各参数的作用写在笔记中。

```
python -c "import json; print(json.load(open('experiments.json', encoding='utf-8'))['day30'])"
```

预期结果：终端显示本课配置；键名与文件中的内容一致。

2. 运行本课基线

为什么做：保持默认设置的一次运行，是后续比较的参照。

执行命令，打开新生成的 outputs/day30.json。先检查 day、mode 和 result，不急着修改代码。

```
python run.py day 30
```

预期结果：结果写入 outputs/day30.json；主线 mode 明确为 offline，代表规则教学流程。

3. 沿输出检查机制

为什么做：不能只看程序有没有报错，要检查它实际做了什么。

格式化阅读输出，重点检查answer、evaluation、capability_card 与 project_checks。把一个输入参数和一个输出字段用箭头对应起来。

```
python -m json.tool outputs/day30.json
```

预期结果：能够定位answer、evaluation、capability_card 与 project_checks，并用自己的话解释该结果。

4. 完成演示与能力卡

为什么做：主动制造可解释的变化，才能验证自己理解的机制。

先完成下方预算实验，再恢复默认设置。用正常、拒绝、受控失败三个案例各录制或写下1分钟演示；对照capability_card补充自己实际验证过的环境与可选路径，最后写下一阶段一个具体任务。

预期结果：交付包含可复现入口、三个案例、能力卡与下一阶段验收标准。

改一改 · 让结业演示触及预算边界

- 先执行 python run.py day 30，将 outputs/day30.json 复制为自己的基线备份。

- 在 VS Code 打开 experiments.json，只修改 day30.max_steps 为 1，保持其他键不变并保存。

- 执行 python -m json.tool experiments.json 检查语法，再执行 python run.py day 30。

- 对比两份结果，重点观察answer、evaluation、capability_card 与 project_checks；解释改变原因后恢复原设置或明确记录保留值。

观察什么：主回答应反映步骤预算不足或相应停止状态；评价和项目检查按自身逻辑运行。记录不同组件的真实结果，不把一个步骤失败扩展成所有功能都失败。

机制解释：一个完整交付要能解释局部失败及其影响范围。预算受控地终止，比隐瞒失败或无限继续更可复现。

常见问题

修改配置后结果没有变化

原因：改了其他天的字段，或编辑后尚未保存。

处理：核对 experiments.json 中的 day30，保存后重新运行；检查实际执行目录。

把受控失败误当成程序坏了

原因：本课实验主动制造了边界条件。

处理：先对照 expected 和状态字段，区分设计内拒绝与未处理异常；保留错误信息再修复。

03 • 验收

交付作品、运行证据和诚实的能力卡，把一个月的学习变成可继续改进的项目。 保留基线、修改结果与自己的解释。

成果文件：outputs/day30.json

☐ 可从README复现正常、拒绝与受控失败三个案例。

☐ 能力卡明确主线离线规则、公开评测、合成数据及可选路径验证状态。

☐ 后续任务包含数据来源、允许动作、十个拟定样例和每天一小时安排。

自测与解析

1. 只有离线主线测试通过时，能否宣称真实模型已完整验证?

- A. 可以，接口相似就够了

- B. 不可以，要分别记录真实模型路径的执行证据

- C. 只要装过依赖就可以

- D. 只要有API key就可以

答案：B。 不同运行路径的外部依赖和失败方式不同，验证结论不能相互替代。

2. 选下一阶段项目时，哪种目标最容易验收?

- A. 做一个万能科研助手

- B. 对一份公开CSV生成有字段说明和数值复核的摘要

- C. 让模型无限自主探索

- D. 先安装所有热门框架

答案：B。 明确输入、输出与检查条件，可以把学习时间转化为可完成、可评价的成果。

小词典

- Capability card：如实记录系统能力、限制、输入输出与验证状态的说明。

- Acceptance criterion：可以通过具体观察判断是否完成的验收条件。

- Next experiment：针对一个明确未知点安排的下一次可比较试验。

04 · 我的记录

- 一个月后，我最能独立解释与修改的机制是什么?

- 如果只再投入七小时，我会先补哪一个真实风险，为什么?

忙碌日：20分钟

- 用5分钟阅读第一个概念，写出本课要解决的问题。

- 用10分钟运行 python run.py day 30，打开 outputs/day30.json，检查一个关键字段。

- 用5分钟完成第一道自测并留一句解释，完整实验与验收留到下次。

保存一份运行结果和自己的解释；精简练习单独记录，不等于整课验收全部完成。

有余力再做：20分钟选学 · 把收藏中的一个问题写成下一阶段实验

- 从CLEAR-Microbiome、SCUT_Minilab_MEF或BioAgentLab选择一个公开、具体、一天可缩小的问题。

- 写出数据来源、字段含义、允许工具与十个拟定评价样例；此步骤不等于已经复现原项目。

- 安排下一周每天60分钟：前三天实现最小流程，两天检查与改错，一天演示，一天复盘或补缺。

参考资料与阅读范围

BioAgentLab：科研 Agent 教材：回看 book/zh/06_tool_contracts.md、18_evaluation_and_calibration.md、28_reproducibility_and_safety.md，核对最终能力卡。
https://github.com/zbyloveyj/BioAgentLab

Anthropic：Building effective agents：重读何时使用简单工作流、何时引入自主Agent，解释自己的设计取舍。
https://www.anthropic.com/research/building-effective-agents

下一步

完成本课后，以一个明确真实任务继续迭代；允许休息与回顾，课程进度不替代实际掌握。