北京大学人工智能研究院 INSTITUTE FOR ARTIFICIAL INTELLIGENCE, PEKING UNIVERSITY
北京大学元培学院 YUANPEI COLLEGE PEKING UNIVERSITY

ToNG: Tutorials on Necessary Groundwork

通班人工智能科研先导课 · 第零讲

自学有道
文档、搜索与 AI

Self-Directed Learning: Documentation, Search & AI

主讲单位
北京大学通用人工智能实验班

主讲人
李子阔

为什么优先读文档

文档：一手信息

官方文档

文档是作者写给使用者的直接说明：比如Python 文档、PyTorch API、Linux的`man`手册。他们不一定最好读，但一定是**最权威、最完整、最接近当前版本**的。

其他：二手转述

博客 / 视频 / 论坛

一般由作者先理解、再转述，可能带来**遗漏、误解、过时**——2022年的视频教程未必适用于现在的API接口。

— Diátaxis 框架

情境不同，读不同的文档

- 入门 → Tutorial：来点新手教程（Python、Git、LaTeX 初次上手）
- 做任务 → How-to：想做具体的某件事（读 CSV、回退 commit、查端口）
- 查细节 → Reference：某某参数是什么意思，某某函数需要哪些参数（`dict.get()` 默认值、`open()` 所需参数）
- 理解原理 → Explanation：为什么可以这样写（虚拟环境原理、可变默认参数注意事项等）

很多"读不懂"，其实是选错了文档类型。

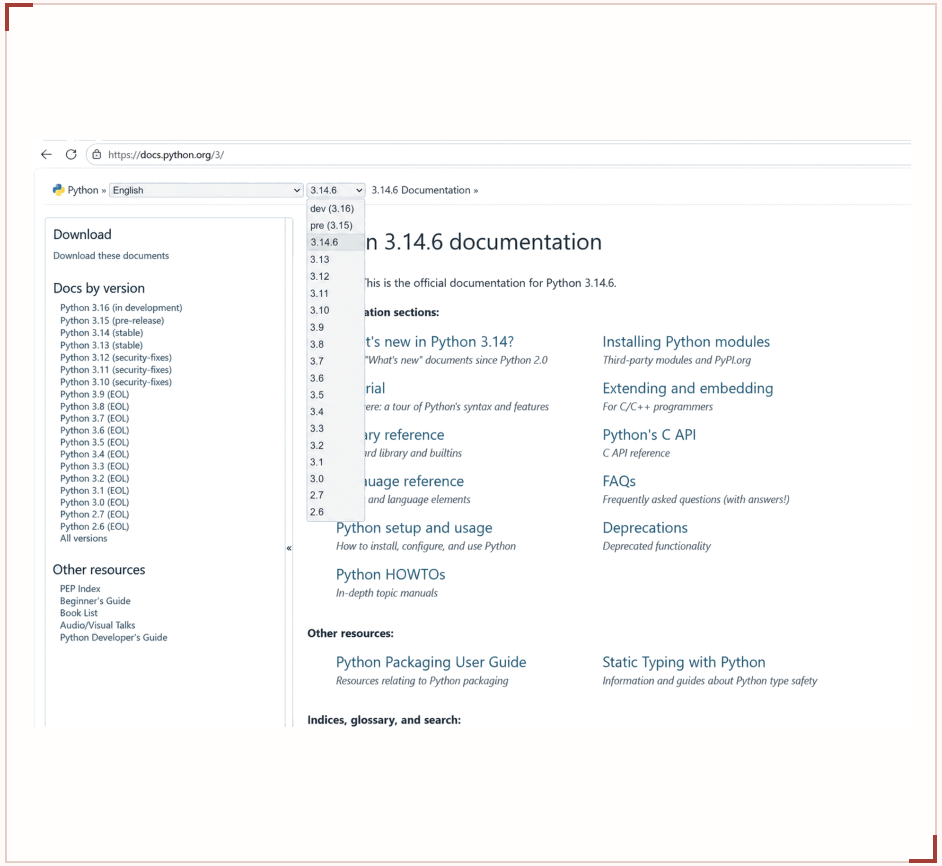
— 版本很重要

查文档前，先定版本

工具的版本更新往往会带来：新增函数、删除参数、改变默认行为，甚至改变安装方式；而驱动版本甚至会直接决定所兼容的库版本，比如某些版本的 `CUDA` 只能对应某些版本的 `pytorch`。

所以我们代码跑不起来，不一定是代码错了，可能只是**运行环境的版本不同**。

所以，看文档先确认版本切换器：用 Python 3.10，就别看 3.12 的新特性；看视频也留意发布时间。



— 找文档

怎么找到文档

搜索引擎：搜"工具名 + documentation"，官方文档基本排在前几位。

GitHub：如果我们在用一些小项目，务必读 README——它是项目的总说明书。

命令行：直接在终端输入 `git --help` / `man` 等命令以获得文档，与本机版本一致、最快最准确。

`python --help` · `git status --help` · `man ls`

注意：把文档当**知识网络**：带着问题去跳转（例如学 `datetime` → 搜 `timedelta` → 遇到 `date` 再点进去），需要什么读什么，而不是从头读到尾。从这个意义上，先读目录是很重要的。

— 搜索

把自然语言拆成关键词

搜索引擎匹配关键词，不完整理解意图。关键词**不需要成句**。

- 去掉"什么是、怎么弄、为什么、的、了、吗"等虚词，保留核心名词、技术名词、错误类型、工具名。
- 带上语言/框架名：`Python KeyError dict` 比 `KeyError dict` 更准。
- 技术搜索尽量英文：Stack Overflow、GitHub Issues、和官方文档质量较高。中文社区不太容易直接通过浏览器搜索得到。

反例："python中如何对列表排序" → 正例：`Python sort list`

快速练习

◆ 30 秒

q01

高质量搜索问题

把下面两个问题改写为中文 + 英文搜索式，只保留核心名词：

"VS Code 里写 Python，按 Ctrl+S 保存文件没有反应。"

"向 GitHub 仓库 push 时，总是提示不匹配。"

参考答案 ① `VS Code Ctrl+S 不生效` / `VS Code save not working`

② `git push 失败 不匹配` / `git push doesn't match`

— 搜索算符

六个搜索运算符

- `"..."` 引号用来精确匹配：`"NoneType" "not subscriptable" Python`
- `site:` 限定结果网站：`site:docs.python.org` · `site:edu.cn`
- `-` 排除关键词：`高等数学 考试 -期中`（- 前有空格、词间不留）
- `filetype:` 限定结果格式：`machine learning filetype:pdf`
- `intitle:` / `inurl:` 限定标题或网址内的关键词
- `*` 通配符：`"the * of programming"`

可组合：`site:docs.python.org "virtual environment"`

— Regex

◆ 匹配一类文本 · 初学够用：集合、数量词、锚点、分组

正则表达式：匹配一类文本

Regex

.	任意一个字符	c.t → cat, cut, c3t
[0-9]	字符集合	[a-zA-Z] 字母 , [^0-9] 非数字
^ $	行首 / 行尾	^error , \.py$
* + ?	数量词	a*c, ab+ , colou?r
\d \w \s	字符类别	\d+ → 202410
(...)	分组	(ab)+ → abababab……
|	选择	cat|dog , cat\.(jpg|png)
\	转义	\. 才是普通句点

— 独立排查报错

搜索报错：三步法

先读报错：找出错误类型（NameError / TypeError / KeyError / IndexError 等）、描述、位置。

提取核心信息：去掉本地路径、用户名，保留语言 + 类型 + 最有辨识度的描述。

优先可靠结果：官方文档、GitHub Issues、Stack Overflow（看阅读量，留意发布时间）。

`data["email"]` 触发 `KeyError: 'email'`，应搜 `Python KeyError dictionary key not found`，而不是只搜 `KeyError`。

练习

◆ 1 分钟

q02

从报错中提取关键词

```
FileNotFoundError: [Errno 2] No such file or directory: '/Users/真实姓名/Desktop/data.csv'
```

提取适合搜索的关键词；

标出公开求助前应替换为占位符的信息。

隐私：报错含用户名、路径。公开前替换为 `/home/user/...`、 。

参考答案：搜 `Python FileNotFoundError data.csv`（或 `FileNotFoundError open file`）；"真实姓名"是用户名，需替换。

— LLM is coming

AI: 站在人类肩膀上的智慧结晶

时至今日，大语言模型已经有能力极大地加速我们学习的进程，是我们必须学会拥抱使用的工具。

怎么选工具

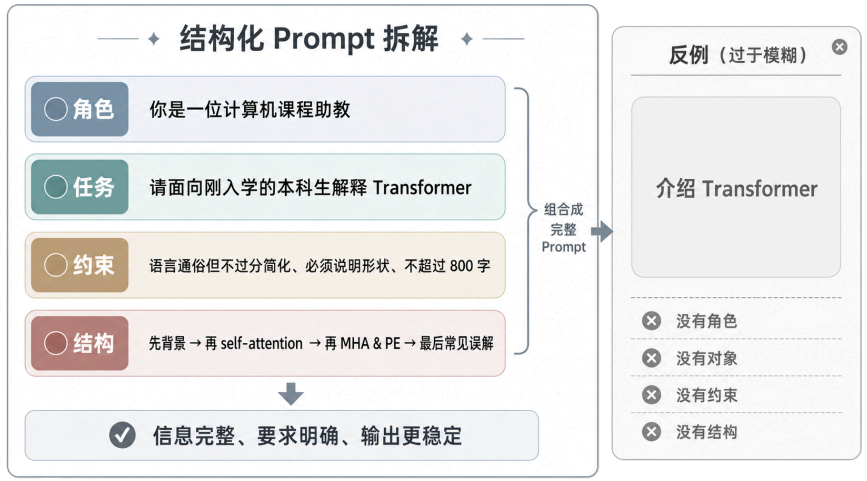
- 技术问题优先 ChatGPT、Claude、DeepSeek 等专业工具；DeepSeek V4、GLM 5.2 对轻量问答性价比高。
- 不推荐豆包、元宝等轻量工具处理技术问题。
- 网页端适合解释 / 翻译 / 轻量调试；API 或客户端适合复杂自动化。

— 简单的提示词工程

角色 · 任务 · 约束 · 结构

- 角色——希望它用什么身份、什么深度回答；
- 任务——要它完成什么；
- 约束——不能做什么、必须包含什么；
- 结构——按什么顺序回答。

例：你是一个introduction to computer science课程的助教，请解释进程和线程的区别，必须给一个 C 语言例子，不要用高度抽象的类比，先给定义再比较最后总结误区。



— AI VS 报错

调试提问：四个要素

- 目标——想实现什么，正常应产生什么结果；
- 代码——最小可复现示例（MRE），删掉无关部分；
- 结果——完整报错，或"预期 vs 实际输出"；
- 已尝试——查过什么、改过什么、结果如何。

整理这四要素，本身就是调试的一部分。

— example

◆ 四要素提问

一个"合格提问"长什么样

example

目标：写一个函数，统计每个字符出现次数，返回出现最多的字符。

代码：

```
def most_common_char(text):
    counts = {}
    for ch in text:
        counts[ch] = counts.get(ch, 0) + 1
    return max(counts.values())
```

实际结果：输入 "abca" 返回 2，预期 "a"；无报错。
已尝试：counts 与 max 行为已确认，但不知如何得到字符。

请解释问题出在哪里，并尽量少改代码给出方案。

— 自主验证

Next Token prediction ≠ 完全准确的内容

易过时

时事与版本

信息可能过时，或把不同版本混在一起。

中间步易错

复杂推理

中间步骤出错但结论写得自然，或最难的一步被省略。

边界与接口

代码

调用不存在的函数、使用废弃接口、遗漏边界情况。

尤其危险

引用论文

AI 会编造看似真实的文献。**不要用 AI 直接生成参考文献**——用 AlphaXiv 等检索真实论文。

把 AI 的回答当**待验证的建议**，不当最终依据。

— 综合实践

搜索定位 → 文档确认 → AI 解释 → 交叉验证

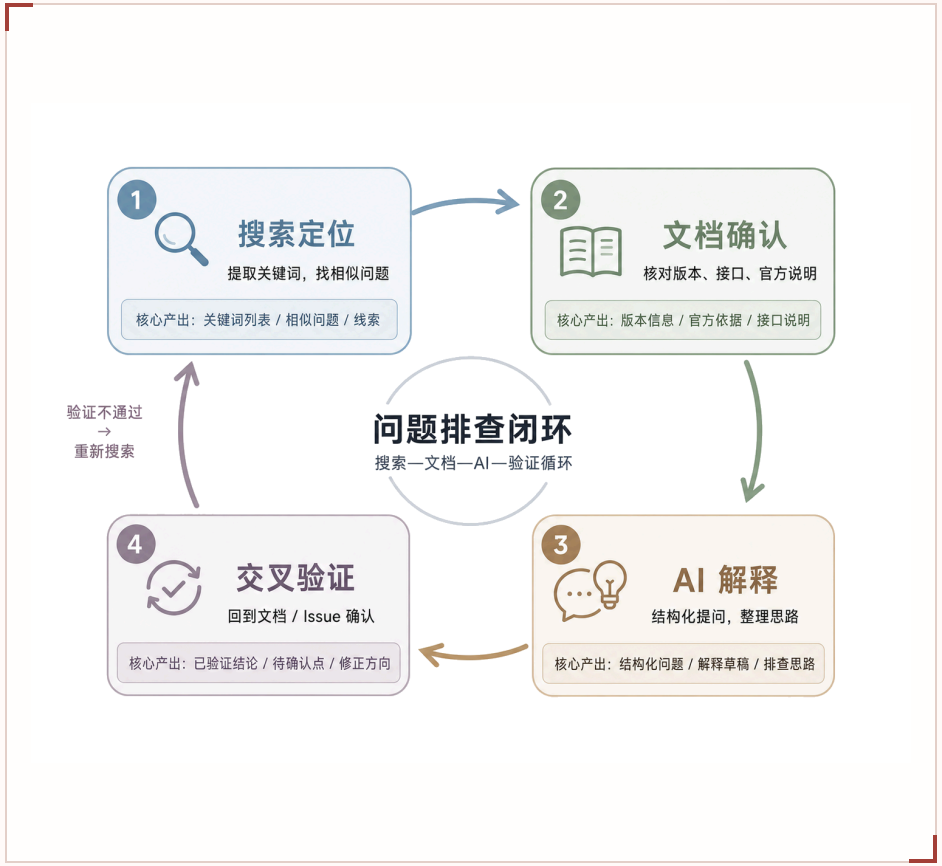
搜索定位：读报错，提取关键词，找相似问题；

文档确认：核对版本、依赖与安装命令；

AI 解释：把最小报错、环境、已尝试交给 AI；

交叉验证：回文档或实际情况验证 AI 判断。

验证不通过就重新搜索或者进一步逼问AI——不断用新事实缩小问题范围。



— Bash

◆ 先自查，别急着重装

实例：库装上了，却仍然报错

Bash

```bash
# pip install rich 后运行脚本，仍报错：
# ModuleNotFoundError: No module named 'rich'

python --version                                    # 解释器版本
python -c "import sys; print(sys.executable)"       # 当前解释器路径
python -m pip --version                             # pip 对应哪个解释器
python -m pip show rich                             # rich 是否在当前环境
```

— 组合使用

缩小问题，再求助搜索与 AI

问题从"为什么没有这个包"缩小为：**装包和运行脚本用的是不是同一个解释器？** 搜索 `Python ModuleNotFoundError package installed wrong interpreter`，优先看官方文档，里面会提到一些虚拟环境之类问题的说明。

或者可以向 AI 提问，把事实给足：

```
ModuleNotFoundError: No module named 'rich' (Windows 上运行脚本时)。
python -m pip show rich 的结果是 <PIP_SHOW_OUTPUT>,
python -c "import sys; print(sys.executable)" 是 <PYTHON_PATH>。
请列出最可能的原因，以及不会修改环境的检查步骤。
```

— 收尾与复盘

用当前解释器重装，并做一分钟复盘

```
python -m pip install rich
python -c "import rich; print(rich.__file__)"
```

最初的猜测？——多为"包坏了 / 没装上"；实际原因：安装环境与实际运行环境不一致

关键命令？——`python -m pip show rich`

为何 `python -m pip` 可以解决？——这个命令明确表示用当前python解释器的 pip安装。

下次怎么查？——先解释器，再环境

留下的不只是一个结果，还有一套可复用的排查方法。

—— 如果只能带走三句话 ——

自学有道

学新工具：先找官方文档，先确认目录与版本。

搜技术问题：提炼关键词、善用英文、精确匹配与范围限定，学会搜报错。

用 AI：它提高效率，但不能代替你的理解、判断与验证。

通