能解释一下 RAG（检索增强生成）的原理吗？：

RAG 即检索增强生成，是指利用外部知识库，通过检索相似文档或片段，然后将这些信息与用户输入一起提供给大模型，从而提高模型回答的准确性和时效性。这种方式有效解决了大模型幻觉问题，保证答案的准确可靠。

在实际开发中如何评测 RAG 或 GraphRAG 的效果？

常见方法包括人工评估和自动评估：

人工评估：请人工标注答案的准确度、相关性；

自动评估：计算召回率、准确率、BLEU 或 ROUGE 指标。

如何构建一个简单的 RAG 系统？

构建 RAG 系统的一般流程：

构建知识库：如使用向量数据库（如 FAISS），存储文档向量；

文档编码：使用 Embedding 模型（如 OpenAI Embeddings）进行向量化；

检索相关文档：使用相似度检索（余弦相似度）；

生成答案：将检索的上下文拼接到 Prompt 中，通过大模型生成答案。

GraphRAG 相比普通 RAG 有哪些区别或优势？

普通 RAG 基于文本片段相似度检索；

GraphRAG 则是将知识表示为图结构，通过图检索，更能体现知识的关联性和层次性；

GraphRAG 适合知识结构复杂且有明显关系的数据集，能更好地提高答案的准确度和逻辑一致性。

谈一下你对 FastAPI 的理解？它有哪些优点？

FastAPI 是高性能的 Web 框架，基于标准 Python 类型提示构建，具有以下优点：

高效的异步支持；

自动生成交互式 API 文档（Swagger、Redoc）；

利用 Pydantic 进行数据校验与序列化；

性能接近 Node.js、Go 语言框架，开发效率高。

Numpy 数组与 Python 列表的区别有哪些？

Numpy 数组效率高，底层采用连续内存存储；

支持矩阵运算、广播操作；

数组要求元素同类型，Python 列表可以存储异构元素；

在数据科学和机器学习场景下，Numpy 更适合处理大量数据运算。

遇到代码效率问题，你如何定位并优化？

使用 Python 内置性能分析工具（如 cProfile）分析代码瓶颈；

优化逻辑复杂性、减少冗余计算；

优化数据结构的选择，比如使用字典或集合减少查找时间；

必要时利用 Numba、Cython 或多线程提高性能。

【 实验二 磁盘调度～测试用例 】

假设磁盘有200个柱面，编号为0－199，当前磁头在30号位置上，并刚刚完成了对39号柱面的服务请求。如果目前磁盘读写请求的队列顺序为95，180，35，120，10，122，64，68；

试问：为完成上述请求，下列算法磁头移动顺序是什么，磁头移动的总量是多少？

(1)先来先服务FCFS

(2)最短查找时间优先算法SSTF

(3)电梯调度