**实验 4: 提示工程**

**一、 实验目的：**

1. 了解提示工程（Prompt Engineering）的基本概念和核心原则。

2. 掌握编写清晰、具体指令以及给予模型思考时间的技巧。

3. 学习使用大语言模型（LLM）完成常见的自然语言处理任务，如文本概括、推断、转换和扩展。

4. 掌握通过迭代优化Prompt以获得更好输出的方法。

5. 初步体验通过API与大语言模型进行交互。

6. **了解并实践思维链（CoT）和提示链（Prompt Chaining）等高级提示技巧。**

**二、 实验环境：**

1. Python 3.x 环境（推荐使用 Anaconda）。

2. IDE：Jupyter Notebook、VS Code 或 Cloud Studio。

3. 能够访问大语言模型（LLM）API（例如，已配置好 API Key 的 OpenAI API，或其他可访问的 LLM 服务）。

4. Python 库：openai（或所使用 LLM 的对应库），panel（用于可选的聊天机器人UI）。

5. **注意：** 需要配置好所选 LLM 的 API Key 或相应的访问凭证。实验中的代码示例默认使用 OpenAI API 格式。

**三、 实验内容：**

准备工作：

首先需要获取一个调用LLM需要的API Key，可以参考的网站：

DeepSeek（注册赠送若干余额）：[DeepSeek 开放平台](https://platform.deepseek.com/usage) 教程：[首次调用 API | DeepSeek API Docs](https://api-docs.deepseek.com/zh-cn/)

ChatGLM（GLM-4-Flash免费）：[智谱AI开放平台](https://www.bigmodel.cn/usercenter/proj-mgmt/apikeys) 教程：[智谱AI开放平台](https://open.bigmodel.cn/dev/api/thirdparty-frame/openai-sdk)

通义千问：[阿里云百炼](https://bailian.console.aliyun.com/?tab=model" \l "/api-key) 教程：[通义千问API文档](https://bailian.console.aliyun.com/?tab=api" \l "/api)

硅基流动（注册赠送14元）：[SiliconCloud](https://cloud.siliconflow.cn/account/ak) 教程：[创建文本对话请求 - SiliconFlow](https://docs.siliconflow.cn/cn/api-reference/chat-completions/chat-completions)

PiscesAPI（注册赠送2000余额）：[Pisces API](https://api.pisces.ink/dashboard) 教程：[文档 - Pisces API](https://api.pisces.ink/docs-custom)



确保你已经安装了必要的 Python 库，并配置好了 LLM 的 API 访问。

你需要一个能够调用 LLM 的函数：

*import openai  
import os  
  
# 从DEEPSEEK, CHATGLM, QWEN, SILICONFLOW, PISCES中选一个  
PLATFORM = "DEEPSEEK"  
API\_KEY = "" # 你申请的API Key  
  
if PLATFORM == "DEEPSEEK":  
 global\_model = "deepseek-chat"  
 base\_url = "https://api.deepseek.com/v1/"  
elif PLATFORM == "CHATGLM":  
 global\_model = "GLM-4-Flash-250414"  
 base\_url = "https://open.bigmodel.cn/api/paas/v4/"  
elif PLATFORM == "QWEN":  
 global\_model = "qwen-plus"  
 base\_url = "https://dashscope.aliyuncs.com/compatible-mode/v1/"  
elif PLATFORM == "SILICONFLOW":  
 global\_model = "deepseek-ai/DeepSeek-V3"  
 base\_url = "https://api.siliconflow.cn/v1/"  
else:  
 global\_model = "chatgpt-4o-latest"  
 base\_url = "https://api.pisces.ink/v1/"  
  
client = openai.OpenAI(  
 api\_key=os.getenv("OPENAI\_API\_KEY", API\_KEY),  
 base\_url=base\_url,  
)*

*# 用于单轮对话  
def get\_completion(prompt, model=global\_model, temperature=0):  
 try:  
 response = client.chat.completions.create(  
 model=model,  
 messages=[{"role": "user", "content": prompt}],  
 temperature=temperature, # 控制模型输出的随机程度  
 )  
 return response.choices[0].message.content  
 except Exception as e:  
 # print(f"API 调用出错: {e}")  
 return f"API Error: {e}"  
  
# 用于多轮对话  
def get\_completion\_from\_messages(messages, model=global\_model, temperature=0):  
 try:  
 response = client.chat.completions.create(  
 model=model,  
 messages=messages,  
 temperature=temperature # 控制模型输出的随机程度  
 )  
 return response.choices[0].message.content  
 except Exception as e:  
 # print(f"API 调用出错: {e}")  
 return f"API Error: {e}"*

第一部分：跟着做 —— 掌握基本提示技巧

请根据教程内容，运行以下示例代码，理解不同提示技巧的效果。

**练习 1：提示原则与技巧**

* 1. **使用分隔符**：

运行以下代码，观察如何使用分隔符（```）来区分指令和输入文本。

*text = f"""*

*您应该提供尽可能清晰、具体的指示，以表达您希望模型执行的任务。\*

*这将引导模型朝向所需的输出，并降低收到无关或不正确响应的可能性。\*

*不要将写清晰的提示词与写简短的提示词混淆。\*

*在许多情况下，更长的提示词可以为模型提供更多的清晰度和上下文信息，从而导致更详细和相关的输出。*

*"""*

*prompt = f"""*

*把用三个反引号括起来的文本总结成一句话。*

*```{text}```*

*"""*

*response = get\_completion(prompt)*

*print("练习 1.1 输出:")*

*print(response)*

1.2 **结构化输出**：

运行以下代码，要求模型以 JSON 格式输出。

*prompt = f"""*

*请生成包括书名、作者和类别的三本虚构的、非真实存在的中文书籍清单，\*

*并以 JSON 格式提供，其中包含以下键: book\_id、title、author、genre。*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 1.2 输出:")*

*print(response)*

1.3 **检查条件**：

运行以下两个代码块，观察模型如何根据文本是否包含指令来执行不同操作。

*# 包含指令的文本*

*text\_1 = f"""*

*泡一杯茶很容易。首先，需要把水烧开。\*

*在等待期间，拿一个杯子并把茶包放进去。\*

*一旦水足够热，就把它倒在茶包上。\*

*等待一会儿，让茶叶浸泡。几分钟后，取出茶包。\*

*如果您愿意，可以加一些糖或牛奶调味。\*

*就这样，您可以享受一杯美味的茶了。*

*"""*

*prompt\_1 = f"""*

*您将获得由三个引号括起来的文本。\*

*如果它包含一系列的指令，则需要按照以下格式重新编写这些指令：*

*第一步 - ...*

*第二步 - …*

*…*

*第N步 - …*

*如果文本中不包含一系列的指令，则直接写“未提供步骤”。"*

*\"\"\"{text\_1}\"\"\"*

*"""*

*response\_1 = get\_completion(prompt\_1)*

*print("\n练习 1.3 (Text 1) 输出:")*

*print(response\_1)*

*# 不包含指令的文本*

*text\_2 = f"""*

*今天阳光明媚，鸟儿在歌唱。\*

*这是一个去公园散步的美好日子。\*

*鲜花盛开，树枝在微风中轻轻摇曳。\*

*人们外出享受着这美好的天气，有些人在野餐，有些人在玩游戏或者在草地上放松。\*

*这是一个完美的日子，可以在户外度过并欣赏大自然的美景。*

*"""*

*prompt\_2 = f"""*

*您将获得由三个引号括起来的文本。\*

*如果它包含一系列的指令，则需要按照以下格式重新编写这些指令：*

*第一步 - ...*

*第二步 - …*

*…*

*第N步 - …*

*如果文本中不包含一系列的指令，则直接写“未提供步骤”。"*

*\"\"\"{text\_2}\"\"\"*

*"""*

*response\_2 = get\_completion(prompt\_2)*

*print("\n练习 1.3 (Text 2) 输出:")*

*print(response\_2)*

1.4 **少样本提示 (Few-shot Prompting)**：

运行以下代码，观察模型如何学习示例的风格进行回答。

*prompt = f"""您的任务是以一致的风格回答问题。*

*<孩子>: 请教我何为耐心。*

*<祖父母>: 挖出最深峡谷的河流源于一处不起眼的泉眼；最宏伟的交响乐从单一的音符开始；最复杂的挂毯以一根孤独的线开始编织。*

*<孩子>: 请教我何为韧性。"""*

*response = get\_completion(prompt)*

*print("\n练习 1.4 输出:")*

*print(response)*

1.5 **要求模型先思考再判断**：

运行以下代码，观察指导模型先自行解决问题如何提高判断的准确性。

*prompt = f"""*

*请判断学生的解决方案是否正确，请通过如下步骤解决这个问题：*

*步骤：*

*首先，自己解决问题。*

*然后将您的解决方案与学生的解决方案进行比较，对比计算得到的总费用与学生计算的总费用是否一致，并评估学生的解决方案是否正确。*

*在自己完成问题之前，请勿决定学生的解决方案是否正确。*

*使用以下格式：*

*问题：问题文本*

*学生的解决方案：学生的解决方案文本*

*实际解决方案和步骤：实际解决方案和步骤文本*

*学生计算的总费用：学生计算得到的总费用*

*实际计算的总费用：实际计算出的总费用*

*学生计算的费用和实际计算的费用是否相同：是或否*

*学生的解决方案和实际解决方案是否相同：是或否*

*学生的成绩：正确或不正确*

*问题：*

*我正在建造一个太阳能发电站，需要帮助计算财务。*

*- 土地费用为每平方英尺100美元*

*- 我可以以每平方英尺250美元的价格购买太阳能电池板*

*- 我已经谈判好了维护合同，每年需要支付固定的10万美元，并额外支付每平方英尺10美元;*

*作为平方英尺数的函数，首年运营的总费用是多少。*

*学生的解决方案：*

*设x为发电站的大小，单位为平方英尺。*

*费用：*

*1. 土地费用：100x美元*

*2. 太阳能电池板费用：250x美元*

*3. 维护费用：100,000+100x=10万美元+10x美元*

*总费用：100x美元+250x美元+10万美元+100x美元=450x+10万美元*

*实际解决方案和步骤：*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 1.5 输出:")*

*print(response)*

**练习 2：文本概括**

使用以下商品评论文本：

*prod\_review = """*

*这个熊猫公仔是我给女儿的生日礼物，她很喜欢，去哪都带着。*

*公仔很软，超级可爱，面部表情也很和善。但是相比于价钱来说，*

*它有点小，我感觉在别的地方用同样的价钱能买到更大的。*

*快递比预期提前了一天到货，所以在送给女儿之前，我自己玩了会。*

*"""*

2.1 **限制长度**：

概括评论，最多30个字。

*prompt = f"""*

*您的任务是从电子商务网站上生成一个产品评论的简短摘要。*

*请对三个反引号之间的评论文本进行概括，最多30个字。*

*评论: ```{prod\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 2.1 输出:")*

*print(response)*

2.2 **侧重特定角度**：

概括评论，最多30个字，侧重快递服务。

*prompt = f"""*

*您的任务是从电子商务网站上生成一个产品评论的简短摘要。*

*请对三个反引号之间的评论文本进行概括，最多30个字，并且侧重在快递服务上。*

*评论: ```{prod\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 2.2 输出:")*

*print(response)*

2.3 **信息提取**：

只提取评论中与产品运输相关的信息，最多30个字。

*prompt = f"""*

*您的任务是从电子商务网站上的产品评论中提取相关信息。*

*请从以下三个反引号之间的评论文本中提取产品运输相关的信息，最多30个词汇。*

*评论: ```{prod\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 2.3 输出:")*

*print(response)*

**练习 3：文本推断**

使用以下台灯评论文本：

*lamp\_review = """*

*我需要一盏漂亮的卧室灯，这款灯具有额外的储物功能，价格也不算太高。\*

*我很快就收到了它。在运输过程中，我们的灯绳断了，但是公司很乐意寄送了一个新的。\*

*几天后就收到了。这款灯很容易组装。我发现少了一个零件，于是联系了他们的客服，他们很快就给我寄来了缺失的零件！\*

*在我看来，Lumina 是一家非常关心顾客和产品的优秀公司！*

*"""*

3.1 **情感倾向分析**：判断评论的情感是“正面”还是“负面”。

*prompt = f"""*

*以下用三个反引号分隔的产品评论的情感是什么？*

*用一个单词回答：「正面」或「负面」。*

*评论文本: ```{lamp\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 3.1 输出:")*

*print(response)*

3.2 **识别情感类型**：识别评论作者表达的情感（不超过五项，逗号分隔）。

*prompt = f"""*

*识别以下评论的作者表达的情感。包含不超过五个项目。将答案格式化为以逗号分隔的单词列表。*

*评论文本: ```{lamp\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 3.2 输出:")*

*print(response)*

3.3 **提取关键信息**：提取评论中购买的物品和制造商，以 JSON 格式输出。

*prompt = f"""*

*从评论文本中识别以下项目：*

*- 评论者购买的物品*

*- 制造该物品的公司*

*评论文本用三个反引号分隔。将你的响应格式化为以 “物品” 和 “品牌” 为键的 JSON 对象。*

*如果信息不存在，请使用 “未知” 作为值。*

*让你的回应尽可能简短。*

*评论文本: ```{lamp\_review}```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 3.3 输出:")*

*print(response)*

**练习 4：文本转换**

4.1 **翻译**：将“您好，我想订购一个搅拌机。”翻译成西班牙语。

*prompt = f"""*

*将以下中文翻译成西班牙语: \*

*```您好，我想订购一个搅拌机。```*

*"""*

*response = get\_completion(prompt)*

*print("\n练习 4.1 输出:")*

*print(response)*

4.2 **语气调整**：将非正式的句子转换成商务信函格式。

*prompt = f"""将以下文本翻译成商务信函的格式:*

*```小老弟，我小羊，上回你说咱部门要采购的显示器是多少寸来着？```"""*

*response = get\_completion(prompt)*

*print("\n练习 4.2 输出:")*

*print(response)*

4.3 **格式转换**：将 Python 字典（JSON 格式）转换为 HTML 表格。

*data\_json = { "resturant employees" :[*

*{"name":"Shyam", "email":"shyamjaiswal@gmail.com"},*

*{"name":"Bob", "email":"bob32@gmail.com"},*

*{"name":"Jai", "email":"jai87@gmail.com"}*

*]}*

*prompt = f"""将以下Python字典从JSON转换为HTML表格，保留表格标题和列名：{data\_json}"""*

*response = get\_completion(prompt)*

*print("\n练习 4.3 输出:")*

*print(response)*

*# 你可以使用以下代码在 Jupyter Notebook 中显示 HTML 表格*

*# from IPython.display import display, HTML*

*# display(HTML(response))*

4.4 **拼写与语法纠错**：纠正文本中的错误。

*text = "This phrase is to cherck chatGPT for spelling abilitty"*

*prompt = f"""请校对并更正以下文本，注意纠正文本保持原始语种，无需输出原始文本。*

*如果您没有发现任何错误，请说“未发现错误”。*

*```{text}```"""*

*response = get\_completion(prompt)*

*print("\n练习 4.4 输出:")*

*print(response)*

**练习 5：文本扩展与温度参数**

使用以下评论和情感：

*sentiment = "消极的"*

*review = f"""*

*他们在11月份的季节性销售期间以约49美元的价格出售17件套装，折扣约为一半。\*

*但由于某些原因（可能是价格欺诈），到了12月第二周，同样的套装价格全都涨到了70美元到89美元不等。\*

*11件套装的价格也上涨了大约10美元左右。\*

*虽然外观看起来还可以，但基座上锁定刀片的部分看起来不如几年前的早期版本那么好。\*

*不过我打算非常温柔地使用它...（省略部分细节）...大约一年后，电机发出奇怪的噪音，我打电话给客服，但保修已经过期了，所以我不得不再买一个。\*

*总的来说，这些产品的总体质量已经下降，因此它们依靠品牌认可和消费者忠诚度来维持销售。\*

*货物在两天内到达。*

*"""*

5.1 **生成邮件回复**：

根据评论和情感生成一封客服邮件。

*prompt = f"""*

*你是一位客户服务的AI助手。*

*你的任务是给一位重要客户发送邮件回复。*

*根据客户通过“```”分隔的评价，生成回复以感谢客户的评价。提醒模型使用评价中的具体细节*

*用简明而专业的语气写信。*

*作为“AI客户代理”签署电子邮件。*

*客户评论：*

*```{review}```*

*评论情感：{sentiment}*

*"""*

*response = get\_completion(prompt, temperature=0) # 使用 temperature=0 保证结果一致性*

*print("\n练习 5.1 输出:")*

*print(response)*

5.2 **体验温度参数**：

使用相同的 Prompt，但设置 `temperature=0.7`，运行两次，观察两次输出的差异。

*print("\n练习 5.2 输出 (第一次, T=0.7):")*

*response1 = get\_completion(prompt, temperature=0.7)*

*print(response1)*

*print("\n练习 5.2 输出 (第二次, T=0.7):")*

*import time*

*time.sleep(20) # 等待20秒，避免可能的频率限制*

*response2 = get\_completion(prompt, temperature=0.7)*

*print(response2)*

**练习 6: 思维链 (Chain of Thought) 与提示链 (Prompt Chaining)**

背景知识:

\* **思维链 (Chain of Thought - CoT):** 这是一种提示技巧，通过引导大语言模型在给出最终答案之前，先展示其推理或思考的过程（即“思考步骤”），从而提高其在复杂推理任务（如数学应用题、逻辑推理等）上的表现。简单地在 Prompt 末尾加上“让我们一步一步地思考”或类似的话，往往就能触发这种行为。

\* **提示链 (Prompt Chaining):** 这是一种将复杂任务分解为多个简单子任务，并通过多个 Prompt 按顺序执行的方法。前一个 Prompt 的输出可以作为后一个 Prompt 的输入，形成一个“链条”，逐步完成整个任务。

**6.1 思维链 (Chain of Thought)**

\* 目标： 解决一个简单的数学应用题，观察模型直接回答与使用 CoT 提示的区别。

\* 问题： 假设一个水果摊有 20 个苹果，卖掉了 5 个，又新进了 15 个。现在水果摊有多少个苹果？

\* 尝试 1 (直接提问):

*prompt\_direct = """*

*一个水果摊有 20 个苹果，卖掉了 5 个，又新进了 15 个。现在水果摊有多少个苹果？请直接给出最终数量。*

*"""*

*response\_direct = get\_completion(prompt\_direct)*

*print("\n练习 6.1 输出 (直接提问):")*

*print(response\_direct)*

*\** 尝试 2 (使用 CoT):

*prompt\_cot = """*

*一个水果摊有 20 个苹果，卖掉了 5 个，又新进了 15 个。现在水果摊有多少个苹果？让我们一步一步地思考。*

*"""*

*response\_cot = get\_completion(prompt\_cot)*

*print("\n练习 6.1 输出 (使用 CoT):")*

*print(response\_cot)*

\* **观察：**对比两次输出，注意两次输出的计算步骤差别。

**6.2 提示链 (Prompt Chaining)**

\* 目标： 分两步处理一段文本：先提取关键人物和地点，再根据提取的信息生成摘要。

\* 文本：

*story\_text = """  
在阳光明媚的周六下午，张伟和他的朋友李娜决定去参观位于市中心的科技博物馆。他们在博物馆里看到了许多有趣的展品，特别是关于人工智能和未来交通的部分。参观结束后，他们在附近的公园散步，讨论着刚才看到的展品。  
"""*

\* 步骤 1: *提取关键信息*

*import json  
  
# 步骤 1  
prompt\_step1 = f"""  
从以下文本中提取所有的人物姓名和地点名称。  
将结果格式化为一个 Python 列表，其中每个元素是一个包含 'type' (人物/地点) 和 'name' (名称) 的字典。  
  
只返回Python类型的字典，无需返回任何其他内容！  
文本：```{story\_text}```  
"""  
response\_step1\_str = get\_completion(prompt\_step1)  
print("\n练习 6.2 输出 (步骤 1 - 提取信息):")  
print(response\_step1\_str)  
  
def extract\_python\_code\_blocks(text):  
 """  
 从文本中提取所有被 ```python 和 ``` 包围的代码块  
 :param text: 输入文本  
 :return: 匹配到的代码块列表  
 """  
 import re  
 pattern = r'```python(.\*?)```'  
 matches = re.findall(pattern, text, re.DOTALL)  
 # 去除每个匹配结果两端的空白字符  
 return [match.strip() for match in matches]  
  
def parse\_entities(response\_step1\_str):  
 """  
 解析实体列表，兼容字符串形式的Python列表或直接是Python列表  
 :param response\_step1\_str: 可能是字符串形式的Python列表或直接是列表  
 :return: 解析后的实体列表  
 """  
 if isinstance(response\_step1\_str, list):  
 # 如果已经是列表，直接返回  
 return response\_step1\_str  
  
 try:  
 # 先尝试用json解析（更安全）  
 return json.loads(response\_step1\_str)  
 except json.JSONDecodeError:  
 try:  
 return extract\_python\_code\_blocks(response\_step1\_str)  
 except (ValueError, SyntaxError) as e:  
 print(f"解析实体列表失败: {e}")  
 return [] # 如果解析失败，返回空列表  
  
# 注意：需要将字符串形式的列表转换为真实的 Python 列表  
# 在实际应用中，需要更健壮的解析方法，这里为了演示简化处理  
extracted\_info = parse\_entities(response\_step1\_str)  
print(extracted\_info)*

\* 步骤 2: *基于提取信息生成摘要*

*# 步骤 2  
  
# 确保 extracted\_info 是一个列表且不为空  
if isinstance(extracted\_info, list) and extracted\_info:  
 prompt\_step2 = f"""  
 根据以下提取的关键信息，为原始文本生成一个简短的摘要（不超过 30 字），摘要应包含这些关键信息。  
  
 关键信息：{extracted\_info}  
 原始文本：```{story\_text}```  
 """  
 response\_step2 = get\_completion(prompt\_step2)  
 print("\n练习 6.2 输出 (步骤 2 - 生成摘要):")  
 print(response\_step2)  
else:  
 print("\n练习 6.2 步骤 2: 未能成功提取或解析关键信息，无法生成摘要。")*

*\** 观察：步骤 1 的输出如何被用作步骤 2 的输入，以完成更复杂的任务。

第二部分：试着做 —— 应用与探索

现在，请运用你在第一部分和教程中学到的知识，完成以下任务。你需要自行设计 Prompt 并记录结果。

**问题 1：新闻文章处理**

请**自行寻找一篇近期的新闻文章（中文）**，然后设计 Prompt 并执行，完成以下任务：

a) 将文章概括为不超过 50 个字。

b) 提取文章中的关键信息（例如：涉及的人物、机构、地点、事件核心要素等）。以 JSON 格式输出。

c) 概括文章内容，并侧重于文章中讨论的某个特定方面（例如：潜在的挑战、积极影响或未来前景等，可自行选择侧重点）。

**问题 2：批量评论分析**

请**自行通过网络搜索或使用爬虫工具获取 3-5 条商品评论或电影评论（中文）**。然后，设计一个 Prompt，能够一次性处理这些评论，完成以下任务：

a) 判断每条评论的情感倾向（正面/负面/中性）。

b) 提取每条评论主要讨论的话题（例如：音质、物流、易用性、剧情、演员表现等）。

c) 将结果以 JSON 列表的形式输出，每个对象包含评论原文、情感倾向和话题。

**问题 3：文本格式与风格转换**

假设你有以下一段描述制作柠檬水的非正式文字：

```

想做柠檬水？挺简单的。先搞点柠檬，使劲挤出汁来。然后加点水，看你喜欢多浓。再放糖，搅匀了尝尝甜不甜。最后扔几块冰进去，搞定！

```

请设计 Prompt 并执行，完成以下任务：

a) 将这段文字改写成一个清晰的、编号的步骤列表。

b) 将生成的步骤列表翻译成英文。

c) 对原始的非正式文字进行拼写和语法检查与修正（假设它可能包含一些常见的口语错误或错别字，即使这个例子很干净，也请尝试设计一个能处理错误的prompt）。

**问题 4：聊天机器人角色设计**

设计一个系统提示（System Prompt），用于定义一个聊天机器人的角色。这个机器人是一个“古代历史知识问答助手”，它应该以一种略带文言的、严谨但友好的风格回答问题。

然后，假设用户问：“请问秦始皇统一六国是在哪一年？” 请根据你设计的系统提示，写出这个聊天机器人可能的第一个回复。

**问题 5：Prompt 迭代优化**

假设你最初使用了以下 Prompt 来提取食谱中的主要食材：

初始 Prompt:

```

从下面的食谱描述中提取主要食材列表。

食谱：一份美味的番茄炒蛋需要新鲜的番茄、几个鸡蛋、少许葱花、盐和油。首先将鸡蛋打散，番茄切块。热锅倒油，先炒鸡蛋，然后盛出。再加点油，放入番茄块翻炒，加盐调味，最后倒入炒好的鸡蛋和葱花，快速翻炒均匀即可。

```

假设模型返回了：“番茄, 鸡蛋, 葱花, 盐, 油, 鸡蛋, 油, 番茄, 盐, 鸡蛋, 葱花”。这个结果包含了重复项，并且包含了调味料（盐、油）。

请：

a) 指出初始 Prompt 和模型输出存在的问题。

b) 展示你如何通过至少两次迭代修改 Prompt 来解决这些问题，目标是只提取主要的、非调味料的食材，且不重复。记录每次迭代的 Prompt 和对应的输出。

**四、 实验要求：**

1. 配置好实验环境，确保能够成功调用 LLM API。

2. 完成“第一部分：跟着做”的所有练习，理解每个技巧的作用。在实验报告中记录关键代码（Prompt）和对应的输出结果。

3. 完成“第二部分：实验着做”的所有问题。在实验报告中详细记录你为每个问题设计的 Prompt、执行代码、得到的输出结果。对于问题 5，清晰展示 Prompt 的迭代过程和理由。

4. **提交实验报告：是。** 实验报告应包含实验目的、环境、详细的实验内容（包括代码、Prompt、输出结果）以及对探索性问题的分析和思考。