

Prompt Engineering 提示词工程

Prompt

Prompt 是什么？

Prompt 如何生成？

Prompt 的重要性

Prompt Engineering定义

Prompt Engineering最佳实践

1. 明确目标
2. 提供上下文
3. 使用具体的指示
4. 提供示例
5. 使用分步指示
6. 控制输出长度
7. 使用占位符和模板
8. 反复试验和调整
9. 指定输出格式
10. 使用多轮对话
11. 使用反思和迭代

提示技术

1. Zero-shot
2. Few-shot
3. Chain-of-Thought (CoT)
4. ReAct
5. Reflexion
6. Prompt Chaining

结构化输出

结构化输出的常见形式

设计结构化输出提示的技巧

结构化输出的示例

示例 1：生成 JSON 格式的数据

示例 2：生成 Markdown 格式的报告

示例 3：生成表格格式的数据

示例 4：生成代码格式

Prompt 应用

1. 数据处理
2. 代码生成
3. 函数定义
4. 分类任务
5. 创意生成
6. 内容生成
7. 汇总信息
8. 风险与安全管理

gpt-4o 图像提示工程技术

1. 明确图像生成目标
2. 提供详细描述
3. 使用参考图像
4. 分步生成
5. 使用多模态提示
6. 控制图像风格和细节
7. 反复试验和调整
8. 使用图像模板和占位符
9. 多轮对话生成
10. 结合图像和文本生成
11. 风险与安全管理

Prompt

Prompt 是什么？

Prompt 是一种人为构造的输入序列，用于引导 GPT 模型根据先前输入的内容生成相关的输出。简单来说，就是你向模型提供的“提示词”。

在 ChatGpt 中，我们可以通过设计不同的 prompt，让模型生成与之相关的文本。例如，假设我们想让 ChatGpt 担任英语翻译。我们可以给模型提供以下 prompt：

- ▼ Plain Text
- 1 我希望你能担任英语翻译、拼写校对和修辞改进的角色。
 - 2 我会用任何语言和你交流，你会识别语言，将其翻译并用更为优美和精炼的英语回答我。
 - 3 请将我简单的词汇和句子替换成更为优美和高雅的表达方式，确保意思不变，但使其更具文学性。
 - 4 请仅回答更正和改进的部分，不要写解释。我的第一句话是“how are you ?”，请翻译它。

这样，我们就可以期待模型生成一段文本了。

Prompt 如何生成？

prompt 如此重要，我们应该怎么去写一个好的 prompt 呢？github 上有位大佬 Matt Nigh。

<https://github.com/mattnigh/ChatGPT3-Free-Prompt-List>

README MIT license

ChatGPT3 Prompt Engineering

Guide and framework for creating ChatGPT3 Prompts

Stars 2.1k - Our GitHub Stars!

Deploy Jekyll site to Pages no status last commit february 2023

This repo was developed by @mattnigh. Follow or connect with me on [my LinkedIn](#).

What is Prompt Engineering?

Prompt engineering is the process of designing and refining the initial text or input (the prompt) that is given to a language model like ChatGPT to generate a response. It involves designing prompts that guide the model to generate a specific tone, style, or type of content.

Prompt Engineering: The Basics

- Use literal and explicit language
- Ask the model to act as if is an expert of the subject
- Ask the model to act as if it is a specific person, or combination of people
- Ask the model to think 'step-by-step', especially in medium to complex tasks
- Experiment with outputs, Give me 10 different examples
- Refine the results, Rewrite this to be more engaging, use clearer language, and use bullets to make it more readable.

CRISPE Prompt Framework，CRISPE 是首字母的缩写，分别代表以下含义：

CR：Capacity and Role（能力与角色），你希望 ChatGPT 扮演怎样的角色。

I：Insight（洞察），背景信息和上下文。

S: Statement (陈述), 你希望 ChatGPT 做什么。

P: Personality (个性), 你希望 ChatGPT 以什么风格或方式回答你。

E: Experiment (实验), 要求 ChatGPT 为你提供多个答案。

github 上的那 prompt 角色大全基本都是 CRISPE 框架。

担任产品经理

请确认我的以下请求。请您作为产品经理回复我。我将会提供一个主题，您将帮助我编写一份包括以下章节标题的PRD文档：主题、简介、问题陈述、目标与目的、用户故事、技术要求、收益、KPI指标、开发风险以及结论。我的需求是：做一个赛博朋克的网站首页。

做表格

请你充当表格生成器。您只会回复我一个包含10行的表格。我会告诉你在单元格中写入什么，你只会以 markdown 表格形式回复结果，而不是其他任何内容。请注意，您的回答应该是简明扼要的，不需要附带任何额外的解释。你只会回复 markdown 表的作为结果。首先，回复我十二生肖表。

充当英语发音帮手

请为说汉语的人提供英语发音帮助。我会给你汉语句子，你需回答正确的英语发音。仅回答发音，不需要翻译或解释。请使用汉语谐音注音。首句：“上海的天气怎么样？”

充当旅游指南

我想让你做一个旅游指南。我会把我的位置写给你，你会推荐一个靠近我的位置的地方。在某些情况下，我还会告诉您我将访问的地方类型。您还会向我推荐靠近我的第一个位置的类似类型的地方。我的第一个建议请求是“我在上海，我只想参观博物馆。”

先定角色，后说背景，再提要求，最后定风格。是否生成多个例子可以看自己喜好。

Prompt 的重要性

合理使用 prompt 可以为 ChatGpt 带来很多好处。以下是一些例子：

- 提高生成准确性：通过正确的 prompt 引导，模型能够更好地理解用户的意图，从而生成更加准确的文本。
- 增强自由度：通过多种不同的 prompt，我们可以让模型生成各种各样的文本，增强了模型的表现力和自由度。
- 提高效率：如果我们已经知道要生成的文本大致内容，通过正确的 prompt 可以让模型更快地生成出我们想要的结果。

Prompt Engineering定义

定义：Prompt Engineering 是设计和优化输入提示（prompt）以获得预期输出的过程。在与大型语言模型（如 GPT-4）交互时，如何构造提示会显著影响模型的回答质量。

例子：

- 简单提示： "告诉我关于猫的事情。"
- 优化提示： "请详细描述猫的生物学特征、行为习惯以及它们在不同文化中的象征意义。"

通过优化提示，用户可以引导模型生成更详细和有用的回答。

Prompt Engineering 是设计和优化输入提示以获得预期输出的过程。为了在使用大型语言模型（如 GPT-4）时获得最佳结果，以下是一些最佳实践：

Prompt Engineering最佳实践

1. 明确目标

最佳实践：明确你希望模型完成的任务或回答的问题。

示例：

- 目标不明确： "告诉我关于气候变化的事情。"
- 目标明确： "请简要描述气候变化的主要原因及其对农业的影响。"

2. 提供上下文

最佳实践：为模型提供必要的背景信息或上下文，以帮助其理解任务。

示例：

- 无上下文： "解释一下微积分。"
- 有上下文： "作为一名高中生，我正在学习微积分。请用简单的语言解释一下微积分的基本概念。"

3. 使用具体的指示

最佳实践：使用明确的指示和要求，避免模糊不清的提示。

示例：

- 模糊指示： "写一篇关于技术的文章。"

- 具体指示: "请写一篇关于人工智能在医疗领域应用的文章, 包含以下几点: 应用场景、优势和挑战。"

4. 提供示例

最佳实践: 通过提供示例来展示你期望的输出格式或内容。

示例:

- 无示例: "生成一个关于产品的报告。"
- 有示例: "生成一个关于产品的报告, 格式如下: \n\n- 产品名称: \n- 价格: \n- 特点: \n- 优点: \n- 缺点: "

5. 使用分步指示

最佳实践: 对于复杂任务, 分解为多个步骤, 逐步引导模型完成。

示例:

- 一步完成: "解释并解决这个数学问题: $2x + 3 = 7$ 。"
- 分步指示: "首先, 解释如何解方程。然后, 解方程 $2x + 3 = 7$ 。"

6. 控制输出长度

最佳实践: 通过提示控制输出的长度, 确保内容简洁或详细。

示例:

- 无长度控制: "解释一下量子力学。"
- 有长度控制: "用不超过100字解释量子力学的基本概念。"

7. 使用占位符和模板

最佳实践: 使用占位符和模板来指示需要填充的内容或格式。

示例:

- 无模板: "生成一个用户注册表单。"
- 有模板: "生成一个用户注册表单, 包含以下字段: 用户名、密码、邮箱、电话号码。"

8. 反复试验和调整

最佳实践：不断试验和调整提示，观察模型的输出，并根据需要进行优化。

示例：

- 初始提示： "描述一下Python编程语言。"
- 调整提示： "描述一下Python编程语言的主要特点和常见应用场景。"

9. 指定输出格式

最佳实践：明确指定输出格式，确保生成内容符合预期。

示例：

- 无格式指定： "生成一个关于公司财务状况的报告。"
- 有格式指定： "生成一个关于公司财务状况的报告，格式如下：\n\n1. 收入：\n2. 支出：\n3. 净利润：\n4. 财务分析："

10. 使用多轮对话

最佳实践：在需要时，通过多轮对话逐步引导模型生成所需内容。

示例：

- 单轮对话： "告诉我关于Python编程的所有信息。"
- 多轮对话：
 - 用户： "告诉我Python编程的主要特点。"
 - 模型： "Python是一种高级编程语言，具有易读性、广泛的库支持和跨平台兼容性。"
 - 用户： "请详细描述Python的常见应用场景。"
 - 模型： "Python常用于Web开发、数据科学、人工智能、自动化脚本和软件开发。"

11. 使用反思和迭代

最佳实践：在生成初步答案后，反思并可能修改其回答，以提高准确性和质量。

示例：

- 初步回答： "Python是一种编程语言。"
- 反思和修改： "Python是一种高级编程语言，广泛用于Web开发、数据科学、人工智能等领域，因其易读性和丰富的库支持而受到欢迎。"

通过遵循这些最佳实践，可以更有效地引导大型语言模型生成高质量的输出，满足各种任务需求。

提示技术

1. Zero-shot

定义：Zero-shot 是指模型在没有任何示例的情况下完成任务。模型必须依靠其预训练知识和提示来生成答案。

例子：

- 提示： "翻译这句话： 'The cat is on the roof.'"
- 回答： "猫在屋顶上。"

模型没有看到过具体的翻译示例，但仍然能够正确翻译句子。

2. Few-shot

定义：Few-shot 是指模型在完成任务之前，先提供几个示例来帮助模型理解任务。

例子：

- 提示： "翻译以下句子： 'The dog is in the garden.' -> '狗在花园里。' 'The bird is in the tree.' -> '鸟在树上。' 'The cat is on the roof.' ->"
- 回答： "猫在屋顶上。"

通过提供几个翻译示例，模型可以更准确地完成翻译任务。

3. Chain-of-Thought (CoT)

定义：Chain-of-Thought 是一种提示技术，通过展示模型思考过程的步骤来解决复杂问题。这种方法可以帮助模型更好地推理和生成答案。

例子：

- 提示： "如果一个苹果和一个香蕉的总价是3美元，一个苹果的价格是2美元，那么一个香蕉的价格是多少？请展示你的思考过程。"
- 回答： "首先，苹果的价格是2美元。总价是3美元，所以香蕉的价格是3美元减去2美元，等于1美元。"

通过展示思考过程，模型可以更清晰地推理出正确答案。

4. ReAct

定义：ReAct 是一种结合反应（Reaction）和行动（Action）的提示技术，通常用于交互式任务或复杂决策。

例子：

- 提示： "你是一个虚拟助手。用户问： '我今天应该穿什么衣服？ ' 你需要根据天气情况给出建议。步骤1： 查询天气。步骤2： 根据天气给出建议。"
- 回答：
 - 步骤1： "查询天气： 今天的天气是晴天， 温度在25-30度之间。"
 - 步骤2： "建议： 今天适合穿轻便的夏装， 比如T恤和短裤。"

通过分步骤反应和行动，模型可以更有效地完成复杂任务。

5. Reflexion

定义：Reflexion 是一种提示技术，模型在生成初步答案后，反思并可能修改其回答。这个过程可以提高答案的准确性和质量。

例子：

- 提示： "解释为什么天空是蓝色的。"
- 初步回答： "因为大气中的氧气和氮气散射阳光中的蓝色光。"
- 反思： "这解释不够准确。实际上，蓝色光被散射得更多是因为瑞利散射效应。"
- 修改回答： "天空是蓝色的，因为阳光穿过大气层时，短波长的蓝色光比其他颜色的光被空气分子散射得更多，这种现象被称为瑞利散射。"

通过反思和修改，模型可以提供更准确和详细的回答。

6. Prompt Chaining

定义：Prompt Chaining 是将多个提示串联起来，以分步解决复杂问题或完成多步骤任务。

例子：

- 任务： 写一篇关于气候变化的文章。
- 提示链：
 - a. "首先，简要介绍什么是气候变化。"
 - b. "接下来，描述气候变化的主要原因。"
 - c. "然后，讨论气候变化的影响。"

- d. “最后，提出应对气候变化的建议。”

通过将任务分解为多个步骤，模型可以更系统和有条理地完成复杂任务。

这些技术和方法帮助用户更有效地与大型语言模型互动，获得更高质量的输出。

结构化输出

结构化输出在 Prompt Engineering 中是指通过设计特定的提示，引导语言模型生成具有明确格式或结构的输出。这在需要处理数据表格、生成代码、创建报告等任务中尤为重要。通过结构化输出，可以确保生成的内容符合预期的格式，便于后续处理和使用。

结构化输出的常见形式

1. **JSON 格式**：适用于需要生成或处理数据对象的任务。
2. **Markdown 格式**：用于生成文档或报告，便于阅读和展示。
3. **表格格式**：适用于数据展示和分析。
4. **代码格式**：用于生成特定编程语言的代码片段。

设计结构化输出提示的技巧

1. **明确格式要求**：在提示中清晰地说明输出的格式。
2. **提供示例**：通过示例展示期望的输出格式。
3. **使用占位符**：在提示中使用占位符来指示需要填充的内容。

结构化输出的示例

示例 1：生成 JSON 格式的数据

提示：

```
1 请生成一个包含以下信息的 JSON 对象：
2 - 名字
3 - 年龄
4 - 职业
5 - 兴趣爱好
6
7 示例格式：
8 {
9   "name": "张三",
10  "age": 30,
11  "occupation": "软件工程师",
12  "hobbies": ["阅读", "旅行", "编程"]
13 }
```

输出：

```
1 {
2   "name": "李四",
3   "age": 25,
4   "occupation": "数据分析师",
5   "hobbies": ["绘画", "跑步", "音乐"]
6 }
```

示例 2：生成 Markdown 格式的报告

提示：

1 请根据以下信息生成一份 Markdown 格式的报告：

- 2 - 标题：气候变化对农业的影响
- 3 - 引言：简要介绍气候变化的背景。
- 4 - 影响：详细描述气候变化对农业的具体影响。
- 5 - 结论：总结并提出应对措施。

6

7 示例格式：

8 # 标题

9

10 ## 引言

11

12 内容

13

14 ## 影响

15

16 内容

17

18 ## 结论

19

20 内容

输出：

1 # 气候变化对农业的影响

2

3 ## 引言

4

5 气候变化是指长期气候模式的变化，主要由人类活动引起的温室气体排放导致。这些变化对全球的生态系统和人类社会产生了深远的影响。

6

7 ## 影响

8

9 气候变化对农业的影响主要体现在以下几个方面：

10 1. **作物生长周期**：温度升高和降水模式的变化影响作物的生长周期，可能导致减产。

11 2. **病虫害**：气候变化可能增加病虫害的发生频率和严重程度，对农业生产构成威胁。

12 3. **水资源**：气候变化导致的极端天气事件，如干旱和洪水，影响农业用水的可用性和质量。

13

14 ## 结论

15

16 为了应对气候变化对农业的影响，需要采取以下措施：

17 1. 推广耐高温和抗旱作物品种。

18 2. 改进农业灌溉技术，提高用水效率。

19 3. 加强农业病虫害监测和防治工作。

示例 3：生成表格格式的数据

提示：

```
1 请生成一张包含以下信息的表格：
2 - 产品名称
3 - 价格
4 - 库存数量
5 - 供应商
6
7 示例格式：
8 | 产品名称 | 价格 | 库存数量 | 供应商 |
9 |-----|-----|-----|-----|
10 | 产品A    | 100  | 50      | 供应商1 |
11 | 产品B    | 200  | 30      | 供应商2 |
```

输出：

```
1 | 产品名称 | 价格 | 库存数量 | 供应商 |
2 |-----|-----|-----|-----|
3 | 产品C    | 150  | 20      | 供应商3 |
4 | 产品D    | 250  | 10      | 供应商4 |
```

示例 4：生成代码格式

提示：

```
1 请根据以下要求生成一段 Python 代码：
2 - 定义一个函数 `add_numbers`，接受两个参数 `a` 和 `b`，返回它们的和。
3 - 打印函数的调用结果，参数为 5 和 10。
4
5 示例格式：
6 ```python
7 def function_name(parameters):
8     # function body
9
10 print(function_name(arguments))
```

```
1
2 **输出**：
3 ```python
4 def add_numbers(a, b):
5     return a + b
6
7 print(add_numbers(5, 10))
```

通过设计结构化输出的提示，可以有效地引导模型生成符合预期格式的内容，方便后续处理和使用。

Prompt 应用

在不同应用场景中使用 Prompt Engineering 可以显著提升大型语言模型的表现。以下是一些具体的应用场景及其最佳实践，包括数据处理、代码生成、函数定义、分类任务、创意生成、内容生成、汇总信息，以及风险与安全管理。

1. 数据处理

应用场景：数据清理、转换和分析。

示例：

- 数据清理：
 - 提示： "请将以下数据转换为标准日期格式： '12/31/2023', '01-01-2024', '2024.02.28'。"
 - 输出： "2023-12-31, 2024-01-01, 2024-02-28"
- 数据转换：
 - 提示： "请将以下CSV数据转换为JSON格式： \nName, Age, City\nAlice, 30, New York\nBob, 25, Los Angeles"
 - 输出：

```
1 [
2   {"Name": "Alice", "Age": 30, "City": "New York"},
3   {"Name": "Bob", "Age": 25, "City": "Los Angeles"}
4 ]
```

2. 代码生成

应用场景：生成和优化代码片段。

示例：

- 代码生成：
 - 提示： "请用Python编写一个函数，计算一个列表中所有数字的平均值。"
 - 输出：

```
1 def calculate_average(numbers):
2     return sum(numbers) / len(numbers)
```

- 代码优化:

- 提示: "请优化以下Python代码以提高性能: \n\n```\nfor i in range(1000000):\n result.append(i * 2)\n```"

- 输出:

```
1 #列表推导式: 通过列表推导式, 我们可以在一行代码中完成原本需要多行的操作。它不仅简洁, 而且通常更高效。
2 #性能提升: 列表推导式在底层进行了优化, 减少了函数调用和解释器的开销, 因此在处理大规模数据时能显著提升性能。
3
4 result = [i * 2 for i in range(1000000)]
```

3. 函数定义

应用场景: 定义特定功能的函数。

示例:

- 提示: "请定义一个Python函数, 接受两个字符串参数, 返回它们的拼接结果。"

- 输出:

```
1 def concatenate_strings(str1, str2):
2     return str1 + str2
```

4. 分类任务

应用场景: 文本分类、情感分析等。

示例:

- 文本分类:

- 提示: "请将以下句子分类为'正面'或'负面': '我今天很开心。' '这个产品真糟糕。'"

- 输出: "正面: '我今天很开心。' 负面: '这个产品真糟糕。'"

- 情感分析:

- 提示: "请分析以下评论的情感: '这部电影非常棒!' '我对这次服务很失望。'"

- 输出: "正面: '这部电影非常棒!' 负面: '我对这次服务很失望。'"

5. 创意生成

应用场景: 生成创意内容, 如故事、广告文案等。

示例:

- 故事生成:
 - 提示: "请编写一个关于勇敢小狗冒险的短篇故事。"
 - 输出: "从前有一只勇敢的小狗, 它每天都在森林里冒险。有一天, 它遇到了一个迷路的小女孩, 决定帮助她找到回家的路....."
- 广告文案:
 - 提示: "请为一款新型智能手表编写一段广告文案。"
 - 输出: "体验未来科技, 掌握健康生活。我们的新型智能手表不仅时尚, 还能实时监测你的健康数据, 助你轻松掌控每一天。"

6. 内容生成

应用场景: 生成文章、报告等。

示例:

- 文章生成:
 - 提示: "请写一篇关于人工智能在医疗领域应用的文章。"
 - 输出: "人工智能在医疗领域的应用正在迅速发展, 从辅助诊断到个性化治疗, AI技术正在改变医疗行业的面貌....."
- 报告生成:
 - 提示: "请生成一份关于2023年公司财务状况的报告, 包含收入、支出和净利润分析。"
 - 输出:

1 ▾ # 2023年公司财务状况报告

2

3 ▾ ## 收入

4 公司在2023年的总收入为5000万美元，主要来源于产品销售和服务收入。

5

6 ▾ ## 支出

7 总支出为3000万美元，主要包括运营成本、员工薪资和研发费用。

8

9 ▾ ## 净利润

10 净利润为2000万美元，比上一年增长了10%。

11

12 ▾ ## 分析

13 2023年，公司在收入和利润方面均实现了稳步增长，主要得益于新产品的成功上市和市场扩展。

7. 汇总信息

应用场景：汇总和总结信息。

示例：

- 信息汇总：

- 提示： "请总结以下文章的主要观点： '文章内容： AI对社会的影响 '"
- 输出： "主要观点： 1. 气候变化是全球性问题。2. 需要全球合作应对。3. 可再生能源是解决方案之一。"

8. 风险与安全管理

应用场景：识别和管理潜在风险与安全问题。

示例：

- 风险评估：

- 提示： "请评估以下项目的潜在风险： '项目描述： AI技术医疗项目 '"
- 输出： "潜在风险： 1. 项目延期。2. 预算超支。3. 技术实现难度大。"

- 安全建议：

- 提示： "请提供关于数据隐私保护的安全建议。"
- 输出： "1. 使用强密码和双因素认证。2. 定期更新和补丁系统。3. 加密敏感数据。4. 进行定期安全审计。"

通过这些应用场景的最佳实践，可以更有效地利用 Prompt Engineering 来实现各种任务，提高模型输出的质量和效率。

gpt-4o 图像提示工程技术

在 GPT-4 及其后续版本中，图像提示工程（Image Prompt Engineering）是一项新兴的技术，旨在通过设计和优化图像提示来引导模型生成高质量的图像输出。这种技术不仅适用于文本生成，还可以在图像生成、图像理解 and 多模态任务中发挥重要作用。

以下是一些在图像提示工程中常用的技术和最佳实践：

1. 明确图像生成目标

最佳实践：明确你希望模型生成的图像类型、风格或内容。

示例：

- **目标不明确：** "生成一张图片。"
- **目标明确：** "生成一张卡通风格的森林场景，包含动物和树木。"

2. 提供详细描述

最佳实践：为模型提供详细的描述，包括场景、颜色、风格等信息。

示例：

- **简单描述：** "生成一张城市图片。"
- **详细描述：** "生成一张现代城市的图片，包含高楼大厦、街道和行人，使用明亮的色调。"

3. 使用参考图像

最佳实践：提供参考图像，帮助模型理解期望的风格或内容。

示例：

- **无参考图像：** "生成一张风景图片。"
- **有参考图像：** "生成一张类似于下面这张图片内容的图像。 \n [参考图像链接: <https://image.pollinations.ai/prompt/dog>]"

4. 分步生成

最佳实践：对于复杂的图像生成任务，可以分步进行，逐步细化图像内容。

示例：

- 一步生成： "生成一张包含海滩、椰子树和日落的图片。"
- 分步生成：
 - a. "首先生成一张海滩的基础图像。"
 - b. "在海滩上添加椰子树。"
 - c. "在背景中添加日落。"

5. 使用多模态提示

最佳实践：结合文本和图像提示，提供更加丰富的信息。

示例：

- 单一提示： "生成一张森林的图片。"
- 多模态提示： "生成一张森林的图片，参考下面的描述和图像。\\n描述：一个宁静的森林，阳光透过树叶。\\n[参考图像链接： <https://image.pollinations.ai/prompt/forest>]"

6. 控制图像风格和细节

最佳实践：通过提示控制图像的风格和细节，确保生成的图像符合预期。

示例：

- 无风格控制： "生成一张花园的图片。"
- 有风格控制： "生成一张印象派风格的花园图片，使用柔和的色调和模糊的边缘。"

7. 反复试验和调整

最佳实践：不断试验和调整提示，观察模型生成的图像，并根据需要进行优化。

示例：

- 初始提示： "生成一张城市夜景的图片。"
- 调整提示： "生成一张城市夜景的图片，包含霓虹灯、车流和高楼大厦的灯光。"

8. 使用图像模板和占位符

最佳实践：使用图像模板和占位符来指示需要填充的内容或格式。

示例：

- 无模板: "生成一张包含建筑物和树木的图像。"
- 有模板: "生成一张包含建筑物和树木的图像, 建筑物在左侧, 树木在右侧。"

9. 多轮对话生成

最佳实践: 在需要时, 通过多轮对话逐步引导模型生成所需的图像内容。

示例:

- 单轮对话: "生成一张包含湖泊和山脉的图片。"
- 多轮对话:
 - 用户: "生成一张包含湖泊的图片。"
 - 模型: "生成的湖泊图片。"
 - 用户: "在湖泊背景中添加山脉。"
 - 模型: "生成的湖泊和山脉图片。"

10. 结合图像和文本生成

最佳实践: 结合图像和文本生成, 创建更加丰富和多样的内容。

示例:

- 图像生成: "生成一张宇宙场景的图片。"
- 图像和文本生成: "生成一张宇宙场景的图片, 并生成一段描述这张图片的文字。"
 - 输出:
 - 图像: 宇宙场景图片
 - 文本描述: "这是一张充满星星和星云的宇宙场景图片, 背景是深邃的黑色, 星星闪烁着微弱的光芒。"

11. 风险与安全管理

最佳实践: 在生成图像时, 注意内容的风险和安全问题, 避免生成不当或敏感的图像。

示例:

- 提示: "生成一张适合所有年龄段观看的风景图片。"
- 输出: 确保生成的图像没有敏感或不当内容。

通过遵循这些最佳实践, 可以更有效地利用图像提示工程技术来生成高质量的图像内容, 满足各种应用需求。

