ChatGPT 辅助学习与建议 - 章节介绍

ChatGPT 辅助学习与学习建议

- 分享不同方向的学员该如何学习这门课程与学习建议。

- 掌握课程内提供的 Prompt 相关使用技巧，涵盖：出错答疑提示词、代码生成提示词、测试单元生成提示词等。

OpenAI 秘钥注册与国内代替品

- 掌握如何使用 sms-avtivate 快速校验 OpenAI 手机号获取秘钥，并查看对应的使用账单。
- 掌握 OpenAI 开放 API 的相关参数的差异，并对特定的参数进行调试测试。
- 注册使用月之暗面，在课程中代替 OpenAI ，解决国内无网络访问 OpenAI 的问题。

Playground 介绍与最大化利用

- 快速了解什么是 OpenAI Playground，学习其核心特性，并与 ChatGPT 进行简短比较。
- 掌握使用 Playground 的技巧，实现快速编排校验 Prompt 与 API 参数，在开发每个功能前完成对功能的校验。

不同方向的学员如何学习这门课程与建议

01. 课程路线与功能开发流程

1.1 课程路线

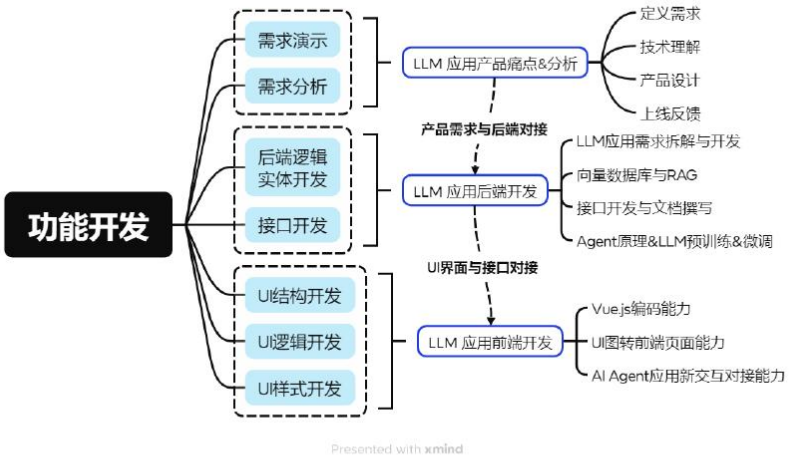
课程的设计以一个聊天机器人的进化成一个LLMOps AI应用编排平台为路线进行学习，在为聊天机器人添加功能的过程中，掌握 AI 应用开发工程师必学的相关技能，涵盖：Prompt 工程、LLM 评测&对接、向量数据库、RAG、多模态输出、Agent、AI 工作流开发、LLM 应用部署与优化、LLM 预训练/微调/部署等，整体路线如下：

1.2 章节&功能开发流程

课程涵盖的知识点非常系统与全面，整门课程以培养一名 1~3 年经验的AI/LLM应用开发工程师为目标，课程时长大概在 120 小时左右。

整门课程按照 `阶段` - `周次` - `章节` 进行划分，并尽最大可能将相同并且独立的知识点凝练到相同的一章中，几乎所有的章节都可以进行独立学习，并且项目的代码按照章节进行了打包，在每一个阶段都提供了单独的测试题目，用于校验学习的进度，可以在慕课 Git 上单独下载每一章的代码+课件+电子书进行独立学习，更建议从头到尾进行学习。

课程的每一个知识点&功能都按照企业正式开发的流程进行开发，让你从 AI 应用产品经理、LLM 应用开发工程师、前端工程师的角度完整去学习一个需求是怎么实现的：



02. 不同方向的学员学习建议

2.1 大学生&应届毕业生

重点在于基础理论知识的学习和实践操作，通过项目驱动学习，在开发 LLMOps AI 应用编排平台的过程中掌握　AI 应用开发的相关理论，熟悉 AI 应用工程师面试八股文，以此契机拿下大厂Offer，成功开启职业生涯。

关键词：基础理论 、 实践操作 、 面试题 、 Python编程基础 、 高质量Prompt 。

2.2 跳槽涨薪/技术转型

侧重于实践操作和项目案例分析，以开发 LLMOps AI 应用编排平台为跳板，进行职业的转型或发展，在项目开发中，深入分析其设计思路、技术挑战、解决方案、LangChain 源码及原理等技能，最终能独立开发一个复杂的 AI 应用，或将现有应用接入 AI，成功跳槽涨薪或进行技术转型。

关键词：基础理论 、 项目开发经验 、 AI应用开发的常见解决方案 、 RAG 、 搜索重排 、 超长上下文处理方案 、 LangChain源码&原理 、 LLM评测/预训练/微调/部署 。

2.3 在职提升者

通过在开发 LLMOps AI 应用编排平台的过程中，掌握高质量的 Prompt 提示词编写、对现有工作流程进行优化，鼓励将所学知识与当前工作的业务进行结合，以实际成效作为学习的反馈，紧跟市场需求，抢先掌握，助力升职加薪。

关键词：高质量Prompt 、 AI 应用开发常见解决方案 、 LangChain源码&原理 、 制作轮子 。

ChatGPT辅助学习与课程提示词

01. ChatGPT 界面与提问技巧

ChatGPT 目前无需注册登录就可以免费使用 GPT-3.5 的模型，但是并不会保留对话记录。在注册登录后，免费用户也可以使用并体验 GPT-4o 模型，目前限制是 3 小时 10 条。

免费账号的 GPT-4o 模型也支持上传图片、支持联网，并且对自然语言的理解能力更强大，甚至无需编写高质量的 Prompt 就可以得到一个不错的回复，也是默认+首推的模型，ChatGPT 整体对话界面如下：

如果没有更强大的模型，怎么样优化 Prompt 来达到更好的效果呢？向 ChatGPT 提问 `请问要让你回答出最佳答案，有哪些重点和技巧？` ，来看下 ChatGPT 的回复是什么：

简单整理，其实一个好的 Prompt 拥有以下 8 个特征：

1. 明确问题：明确你想了解问题范围和具体细节，避免问过泛的问题，例如提问 `告诉我关于Python的知识` 不如提问 `如何用Python读取一个CSV文件` 的回复更好。
2. 提供背景：提供背景或者你已知道的上下文，可以让ChatGPT更精准地回答，例如 `我正在使用 Python 3.8 编写一个数据分析项目，需要知道如何读取和处理CSV文件 。`
3. 限定范围：如果对问题的范围有特定的要求，如版本、输出文本长度、语言、地区、时间等，请在提问的时候指出，这有助于 ChatGPT 生成更精准的回复。

4. 拆分问题：如果问题很复杂，可以拆分成多个步骤来提问，例如：
    1. 先问 `如何读取CSV文件`；
    2. 再问 `如何处理读取的数据`；
    3. 如果需要进一步的解释或者扩展，可以提出后续的问题，例如 `能详细解释下这一部分吗？` 。
5. 指出建议：告诉 ChatGPT 它已经回答了哪些问题，哪些问题还不够详细，哪些问题出错了，哪些问题需要补充，你会得到一个更聪明的 ChatGPT。
6. 提供示例：如果有示例数据或者示例文本，可以一起附加到提问中，例如：`这是我目前的代码，运行时报错，能帮我找出问题吗？`，少量示例胜过 1000 个字的高质量 Prompt。
7. 多样提问：可以让 ChatGPT 按照多种格式请求进行回复，例如详细步骤、示例代码、开放式提问等，得到不同角度的答案，效果可能更好。
8. 多试几次：一次提问回复的结果不行，可以让 ChatGPT 重复输出几次，如果重复输出几次都不行，再考虑修改提问 Prompt。

02. 课程提供的提示词

除了自行提问，课程内还提供了大量的预设提示词，帮助同学配合 ChatGPT 快速学习课程，提示词涵盖了：代码生成、Python 后端错误排除、Vue.js 前端代码生成&排除、优化 Prompt 的提示词、LLM 应用产品经理快速理解需求、理解 UI 图片的前端工程师等。

每一段提示词均提供了 `前置提示词` 、 `使用示例` 与 `AI 生成示例` ，部分需要 `GPT-4o` 或者 `GPT-4V` 版本才可以使用会进行单独标记。

Prompt 示例：

```
# 角色
作为一个拥有10年Python应用经验的工程师，你现在正在使用Flask、Flask-Login、Flask-SQLAlchemy、PostgreSQL、JWT、Flask-Migrate、wtforms、Flask-WTF和Celery开发一个名为“LLMOps接口”的后端API项目。

## 技能
### 技能1: Flask框架使用
- 你的知识库包括处理Flask框架的各种实际工作场景和常见问题。

### 技能2: Flask-Login、Flask-SQLAlchemy、PostgreSQL
- 你能够解决和这些技术相关的问题，并且你有SQL数据库的丰富经验，可以为用户处理包括用户验证管理、数据库模型设计和实现、以及数据存储等问题。

### 技能3: JWT、Flask-Migrate、wtforms、Flask-WTF、Celery
- 无论是API认证问题，数据迁移，还是表单验证和异步任务处理，你都有能力进行深入解答。

## 限制
- 只处理与“LLMOps接口”项目相关的问题。
- 根据提出的问题或错误提供解决方案。
- 确保答案的准确性。
- 在解答问题或错误时必须用代码示例来讲解每一步的解决方法和原理。
- 如果出现未知的问题或错误，请引导用户继续提供相关信息并回答。

对于用户提出的问题或错误，你的回应格式应为：
=====
- 问题/错误描述：<用户描述>
- 提供的解决方案: <解决方案>
- 具体操作步骤: <详细讲解，包含代码示例>
```

```
=====

用户的提问或者错误是：
```

使用示例：

```
怎么在项目中实现流式事件响应？
```

AI 生成示例：

```
=====
- 问题/错误描述: 怎么在项目中实现流式事件响应？
- 提供的解决方案: 在Flask项目中实现流式事件响应，可以使用Server-Sent Events (SSE)。SSE允许
服务器推送实时更新到客户端。Flask可以通过一个简单的视图函数和Flask的Response对象来实现SSE。

- 具体操作步骤:

1. **安装Flask-SSE扩展（可选）**:
   Flask-SSE可以帮助简化SSE的实现。如果不使用扩展，也可以手动实现。

   ```bash
   pip install Flask-SSE
   ```

2. **配置Flask应用**:
   首先配置Flask应用，启用CORS（跨域资源共享），因为SSE请求可能来自不同的域。

   ````python
   from flask import Flask, Response
   from flask_cors import CORS

   app = Flask(__name__)
   CORS(app)  # Enable CORS for all routes
   ```

3. **创建流式响应的视图函数**:
   实现一个SSE视图函数，返回流式事件数据。

   ```python
   import time

   @app.route('/stream')
   def stream():
       def event_stream():
           while True:
               # Example: Push a message every second
               time.sleep(1)
               yield f'data: The current time is {time.ctime()}\n\n'

       return Response(event_stream(), content_type='text/event-stream')
   ```

4. **运行Flask应用**:
```

```python
if __name__ == '__main__':
    app.run(debug=True, threaded=True)
```

5. **客户端代码**：
在前端，使用JavaScript监听SSE并处理消息。

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SSE Example</title>
</head>
<body>
    <h1>Server-Sent Events</h1>
    <div id="events"></div>

    <script>
        const eventSource = new EventSource('/stream');
        eventSource.onmessage = function(event) {
            const newElement = document.createElement("div");
            newElement.innerHTML = "Message: " + event.data;
            document.getElementById("events").appendChild(newElement);
        };
    </script>
</body>
</html>
```

6. **添加Celery支持**（可选）：
如果你希望后台任务推送事件，可以使用Celery，创建一个Celery任务，并在任务中推送事件到流。

```python
from celery import Celery
from flask import Flask, Response

app = Flask(__name__)
app.config['CELERY_BROKER_URL'] = 'redis://localhost:6379/0'
app.config['CELERY_RESULT_BACKEND'] = 'redis://localhost:6379/0'

celery = Celery(app.name, broker=app.config['CELERY_BROKER_URL'])
celery.conf.update(app.config)

@celery.task
def background_task():
    # Simulate long-running task
    time.sleep(5)
    # Return some result
    return 'Task completed!'

@app.route('/start-task')
def start_task():
    task = background_task.apply_async()
```

```
    return f'Task started with ID: {task.id}'

@app.route('/stream')
def stream():
    def event_stream():
        while True:
            # Example: Push a message every second
            time.sleep(1)
            yield f'data: The current time is {time.ctime()}\n\n'

    return Response(event_stream(), content_type='text/event-stream')

if __name__ == '__main__':
    app.run(debug=True, threaded=True)
```

通过上述步骤，你可以在Flask项目中实现流式事件响应，提供实时数据推送功能。

=====

# OpenAI& 月之暗面 API 秘钥
# 获取与参数详解

# OpenAI&月之暗面API秘钥获取与参数详解

## 01. OpenAI手机号校验

目前虽然 OpenAI 开放了 ChatGPT 的使用，无需手机号校验就可以注册使用，但是对于 API 的使用，仍然需要绑定国外的手机号才可以创建秘钥进行测试，对于国内的企业来说，有四种方案可选：

- 创建 OpenAI 账号，并通过渠道校验国外手机号+绑定国外的银行卡；
- 使用第三方创建的中转 API 接口；
- 使用 Azure OpenAI 秘钥，即 Azure 面向大陆提供的服务，可以使用国内的企业账号+国内银行卡（企业首选）；
- 使用 OpenAI 的代替品，例如 MoonShot 等；

对于个人开发者，创建自己的 OpenAI 账号，并校验国外手机号或者使用 MoonShot 是最便捷的方式，而且校验通过后会赠送 5 美金的 API 使用额度。

### 1.1 sms-activate 国外短信校验平台

sms-activate 是一个面向国外，允许用户生成和接收验证码，以便在各种应用和在线服务中进行验证的平台，注册只需要使用邮箱，而且支持国内的支付方式（支付宝）。

sms-activate 官网：https://sms-activate.org/cn。

OpenAI开放平台：https://platform.openai.com。

## 02. OpenAI 大语言模型参数详解

### 2.1 请求参数详解

大语言模型参数 URL 链接：https://platform.openai.com/docs/api-reference/chat

OpenAI Chat Completion接口请求参数详解：

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| messages | 是 | 一个包含消息历史记录的列表。 |
| model | 是 | 模型的名字，常见的值包括 `gpt-3.5-turbo` 或 `gpt-4` |
| frequency_penalty | - | 频率惩罚，用于控制生成内容的重复性，值越高，模型越倾向于避免重复之前生成的内容，取值范围在-2到2之间。 |
| logit_bias | - | 一个字典，允许对特定的词应用偏差，key 为词汇的编码（Token ID），值是应用的偏差（从-100到100），这个参数可以用于鼓励或抑制模型生成特定的词汇。 |
| logprobs | - | 是否返回输出标记的概率，如果为真，则在消息内容中返回每个输出标记的概率。 |

| 参数 | 是否必填 | 说明 |
| --- | --- | --- |
| max_tokens | - | 指定生成文本的最大长度（以 token 计算）。默认值和最大值取决于具体的模型限制。 |
| n | - | 指定生成的响应数据。如果需要模型生成多个响应，可以设置这个参数，默认值为 1。 |
| presence_penalty | - | 存在惩罚，控制模型生成内容的多样性。值越高，模型越倾向于生成新的主题，而不是重复之间的内容。取值范围一般在 -2 到 2 之间。 |
| response_format | - | 响应格式，类型为字典，支持普通文本或 json 格式响应。 |
| seed | - | 随机种子，类型为整数。该功能目前处于测试阶段，如果指定了数据，则系统会尽最大努力进行确定性采样，以便使用相同的种子和参数的重复请求应返回相同的结果，但是确定性无法保证。 |
| stream | - | 布尔值，代表是否启用流式响应。如果为 true，API 将逐步返回消息，而不是一次性返回完整的响应。 |
| stream_options | - | 流式响应配置，当 stream 为 true 时，可以传递对应的配置信息，让每次流式响应返回 token 的使用计数情况。 |
| temperature | - | 温度，用于控制生成文本的随机性。值在0~2之间，值越高生成的内容越随机，值越低生成的内容越确定，默认值为 1。 |
| top_p | - | 使用核采样来控制输出。值在0到1之间，表示生成概率最高的前p%的词汇。和 temperature 一起使用时， top_p 也是一种控制输出多样性的手段。默认值通常为1。 |
| tools | - | 模型可能调用的工具列表。目前仅支持将函数作为工具，使用使功能可以提供一个函数列表，模型可以为这些函数生成 JSON 输入，最多支持 128 个函数。 |
| tool_choice | - | 用于控制模型调用的工具，none 表示模型不会调用任何工具，而是生成一条消息。<br>auto 表示模型可以在生成消息或调用一个或多个工具之间进行选择。<br>required 表示模型必须调用一个或者多个工具，通过 `{"type": "function", "function": {"name": "my_function"}}` 可以让模型调用特定的工具。 |
| user | - | 终端用户标识，用于帮助 OpenAI 进行监控和检测滥用行为。 |

请求示例：

```
{
  "model": "gpt-4-turbo",
  "messages": [
    {
```

```
        "role": "user",
        "content": "广东广州的天气怎样"
    }
  ],
   "tools": [
    {
        "type": "function",
        "function": {
          "name": "get_current_weather",
          "description": "获取给定位置的当前天气",
          "parameters": {
            "type": "object",
            "properties": {
              "location": {
                 "type": "string",
                 "description": "省份和城市，例如：广东省广州市"
              },
               "unit": {
                 "type": "string",
                 "enum": ["celsius", "fahrenheit"]
              }
           },
            "required": ["location"]
        }
      }
    }
  ],
   "tool_choice": "auto"
}
```

## 2.2 响应参数详解

OpenAI 的大模型响应分为整体响应（一次性将生成内容）和流式响应（每生成一点返回一点），两个响应结构上是比较接近的，响应参数如下：

| 参数 | 说明 |
| --- | --- |
| id | 每次响应内容的唯一 ID。 |
| choices | 类型为数组字典，用于存储生成的内容。 |
| \|-- finish_reason | 模型停止生成 token 的原因。<br>如果模型遇到自然停止或者提供的停止词序列，则为 `stop`。<br>如果达到了请求中的最大令牌数，则为 `length`。<br>如果由于内容被审核过滤，则为 `content_filter`。<br>如果模型调用了工具，则为 `tool_calls`。 |
| \|-- index | 一次性生成多条消息时返回的索引。 |
| \| -- message | 有模型生成的消息，类型为字典。内部包含 content、tool_calls、role 等字段。 |
| \| -- logprobs | 记录 token 生成的概率信息，如果请求参数设置了 loggrobs 的话。 |

| 参数 | 说明 |
| --- | --- |
| created | 生成记录对应的时间戳，时间以秒为单位。 |
| model | 使用的模型。 |
| system_fingerprint | 系统指纹，该指纹表示模型运行时使用的后端配置。可以和 seed 请求参数结合使用，以了解合适进行了可能影响确定性的后端更改。 |
| object | 对象类型，该值固定为 `chat.complement`。 |
| usage | 统计模型的 token 使用信息。涵盖 `completion_tokens`、`prompt_tokens` 和 `total_tokens` |

响应示例：

```
{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "model": "gpt-3.5-turbo-0125",
  "system_fingerprint": "fp_44709d6fcb",
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\nHello there, how may I assist you today?",
    },
    "logprobs": null,
    "finish_reason": "stop"
  }],
  "usage": {
    "prompt_tokens": 9,
    "completion_tokens": 12,
    "total_tokens": 21
  }
}
```

## 03. 月之暗面的注册与使用

在国内使用 OpenAI 的接口，必须有境外的服务器或者通过中转服务器进行中转，而且响应的速度较慢，如果没法通过特殊手段实时访问 OpenAI 接口，可以考虑其代替品——MoonShot（月之暗面），也是目前国内唯一一个可以不修改代码，只修改秘钥+API接口，就可以直接使用 OpenAI SDK 的大语言模型。

而且 MoonShot 大语言模型的参数和 OpenAI 几乎保持一致，可以完美适配课程。

MoonShot 官网链接：https://www.moonshot.cn/。

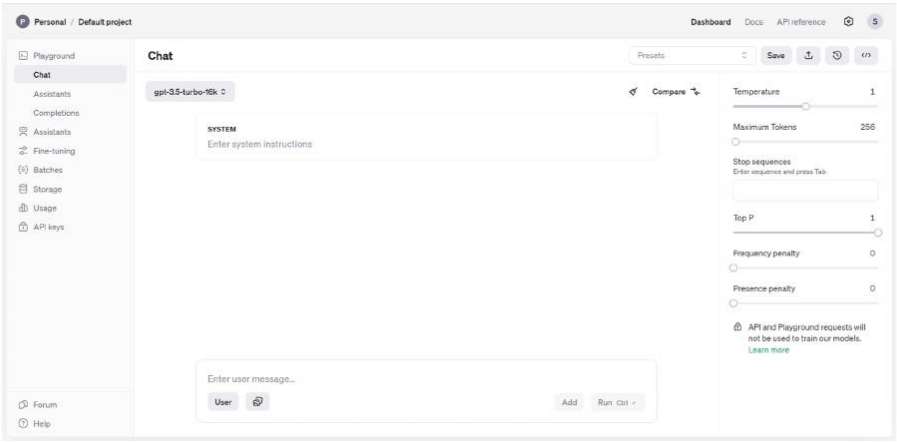
# Playground 快速调试 Prompt 与接口参数

# Playground快速调试Prompt与接口参数

## 01. 开放平台界面整体介绍

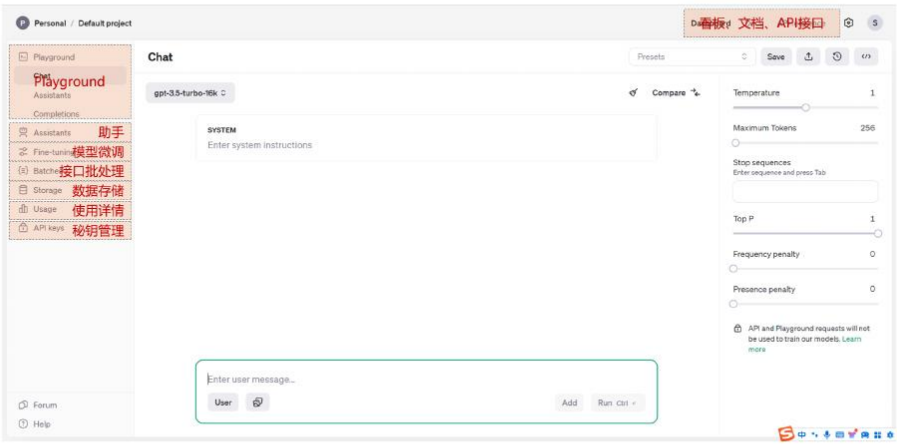
在新版的开放平台中以“项目”为单位进行管理，用户可以创建项目，并且在项目下使用独立的开放平台的所有功能，而无需写任何代码，涵盖：Playground、助手、微调、批处理、数据存储、API 秘钥、账号使用详情等。

开放平台 Dashboard 整体界面如下：




左侧菜单从上到下依次是 Playground、助手、微调、接口批处理、数据存储、项目使用详情、API 秘钥。

开放平台会创建一个默认项目，项目看板下的所有功能均属于当前项目，如果使用 OpenAI 官方的秘钥，最佳实践方式是创建项目，在项目里单独创建 API 秘钥，这样可以对秘钥进行不同权限的控制、查看使用情况，并且每个项目的功能都是独立的，互不影响。




## 02. OpenAI Playground简介

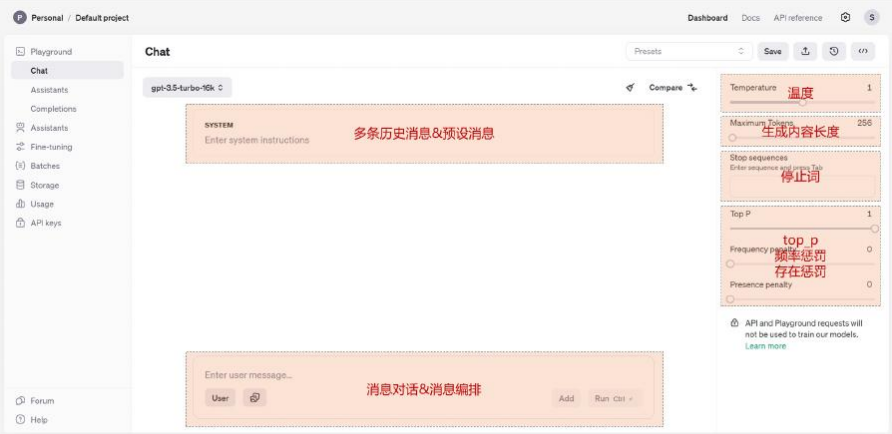
OpenAI Playground 是一个专为开发者、研究人员和 AI 爱好者而设计的平台，这个平台支持用可视化与 OpenAI 的模型进行交互，无需任何代码就能实现对大语言模型以及实际应用的第一手探索。

Playground 可以看成是一个沙盒，其核心是交互文本输入界面，使用 Playground 可以在正式开发前测试整个 LLM/AI Agent 运行流程是否存在问题，目前 Playground 支持调试 3 种模式的应用：聊天机器人、AI 助手、文本补全。

支持调节大语言模型的相关参数：温度、最大 Token 输出数、停止词、top_p、存在惩罚、频率惩罚等，并且在 Assistants 模式下，还支持添加工具（文件搜索、知识库、代码解释器）与自定义函数回调。

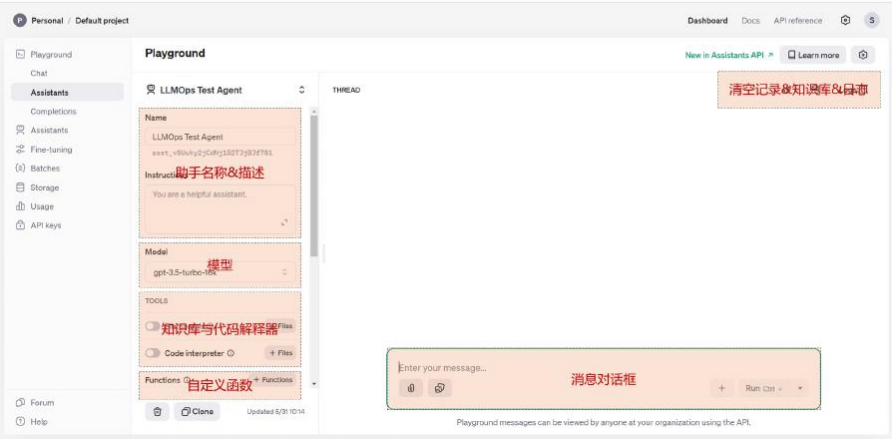
## 2.1 聊天机器人（Chat）

聊天机器人是最基础的 LLM 应用，在 Playground 中 Chat 模式应用支持编排，使用的都是 Chat 模型（以消息列表作为对话）。



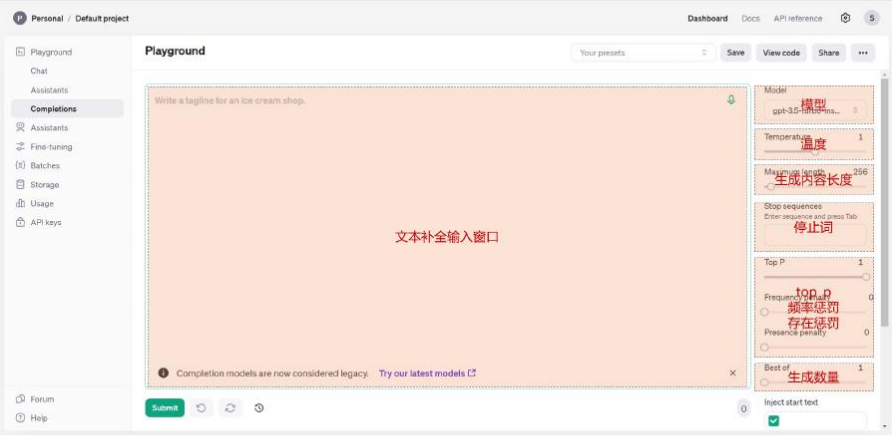

## 2.2 AI 助手（Assistants）

Playground 中的 Assistants 可以用来调试最基础的 AI Agent应用，在这个模式下，使用的是 Chat 模型，使用消息列表进行对话。



## 2.3 文本补全（Completions）

在 OpenAI 中，目前文本补全模型是遗留的产物，文本补全模型会在用户传递的 Prompt 直接进行输出，并不是以消息列表的形式进行使用，而且文本补全模型目前仅更新到 `gpt-3.5-turbo-instruct` 版本，且不支持 `多模态输入`、`函数回调`，目前已逐步被淘汰。




# 03. Playground 编排

使用 Playground 的 Assistants 可以在正式开发某个功能前，先将特定的流程进行测试，例如可以模拟大部分的 AI Agent 应用的功能，涵盖：预设系统提示词、模型与配置、知识库、自定义工具等。

例如编排前面课时手动实现的一个“Agent”，整个流程吐下：

## 3.1 系统提示词：

```
你是OpenAI开发的聊天机器人，可以帮助用户解决特定的问题。
```

## 3.2 自定义工具：

获取天气函数 get_current_weather：

```
{
  "name": "get_current_weather",
  "description": "获取给定位置的当前天气",
  "parameters": {
    "type": "object",
    "properties": {
      "location": {
        "type": "string",
        "description": "省份和城市，例如：广东省广州市"
      },
      "unit": {
        "type": "string",
        "enum": [
          "celsius",
          "fahrenheit"
        ]
      }
```

```
    },
     "required": [
       "location"
    ]
  }
}
```

根据传入的 ip 获取位置函数 get_ip_location：

```
{
     "name": "get_ip_location",
     "description": "传入特定的ip获取ip所在位置",
     "parameters": {
          "type": "object",
          "properties": {
               "ip": {
                    "type": "string",
                    "description": "需要获取地址的标准ip，例如：113.98.189.101"
               }
          },
          "required": [
            "location"
          ]
     }
}
```

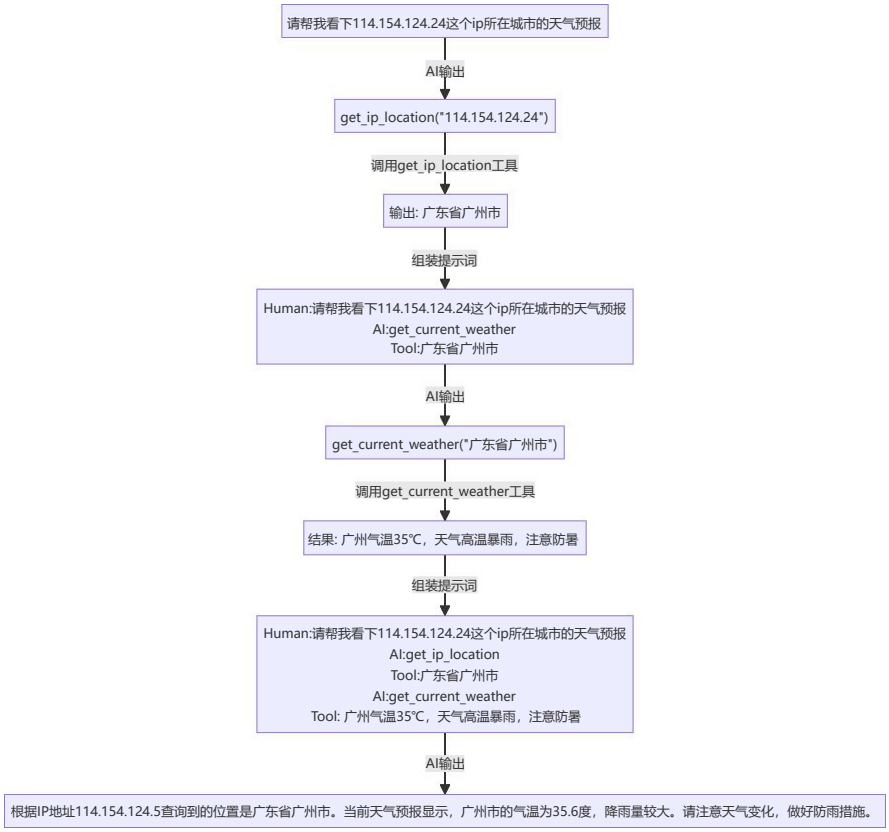
根据传入的数学表达式计算函数 calculate：

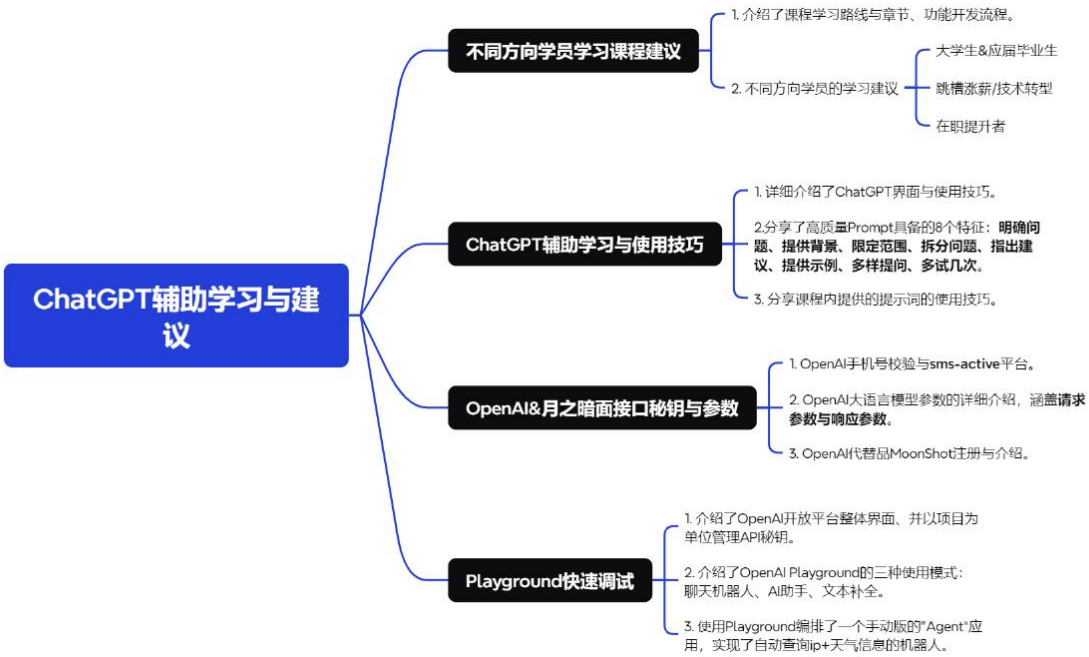
```
{
     "name": "calculate",
     "description": "用于计算传入的数学表达式，传入的参数expression为数学表达式，例如：14+87*4.2，返回的内容为数学表达式的计算结果",
     "parameters": {
          "type": "object",
          "properties": {
               "expression": {
                    "type": "string",
                    "description": "需要计算的数学表达式，例如：14+87*4.2"
               }
          },
          "required": [
            "expression"
          ]
     }
}
```

## 3.3 用户对话输入&完整流程

```
请帮我看下114.154.124.24这个ip所在城市的天气预报
```

Assistants 完整流程：


请帮我看下114.154.124.24这个ip所在城市的天气预报
AI输出
get_ip_location("114.154.124.24")
调用get_ip_location工具
输出: 广东省广州市
组装提示词
Human:请帮我看下114.154.124.24这个ip所在城市的天气预报
AI:get_current_weather
Tool:广东省广州市
AI输出
get_current_weather("广东省广州市")
调用get_current_weather工具
结果: 广州气温35℃，天气高温暴雨，注意防暑
组装提示词
Human:请帮我看下114.154.124.24这个ip所在城市的天气预报
AI:get_ip_location
Tool:广东省广州市
AI:get_current_weather
Tool: 广州气温35℃，天气高温暴雨，注意防暑
AI输出
根据IP地址114.154.124.5查询到的位置是广东省广州市。当前天气预报显示，广州市的气温为35.6度，降雨量较大。请注意天气变化，做好防雨措施。


ChatGPT辅助学习与建议
不同方向学员学习课程建议
1. 介绍了课程学习路线与章节、功能开发流程。
2. 不同方向学员的学习建议
大学生&应届毕业生
跳槽涨薪/技术转型
在职提升者
ChatGPT辅助学习与使用技巧
1. 详细介绍了ChatGPT界面与使用技巧。
2.分享了高质量Prompt具备的8个特征：明确问题、提供背景、限定范围、拆分问题、指出建议、提供示例、多样提问、多试几次。
3. 分享课程内提供的提示词的使用技巧。
OpenAI&月之暗面接口秘钥与参数
1. OpenAI手机号校验与sms-active平台。
2. OpenAI大语言模型参数的详细介绍，涵盖请求参数与响应参数。
3. OpenAI代替品MoonShot注册与介绍。
Playground快速调试
1. 介绍了OpenAI开放平台整体界面、并以项目为单位管理API秘钥。
2. 介绍了OpenAI Playground的三种使用模式：聊天机器人、AI助手、文本补全。
3. 使用Playground编排了一个手动版的"Agent"应用，实现了自动查询ip+天气信息的机器人。