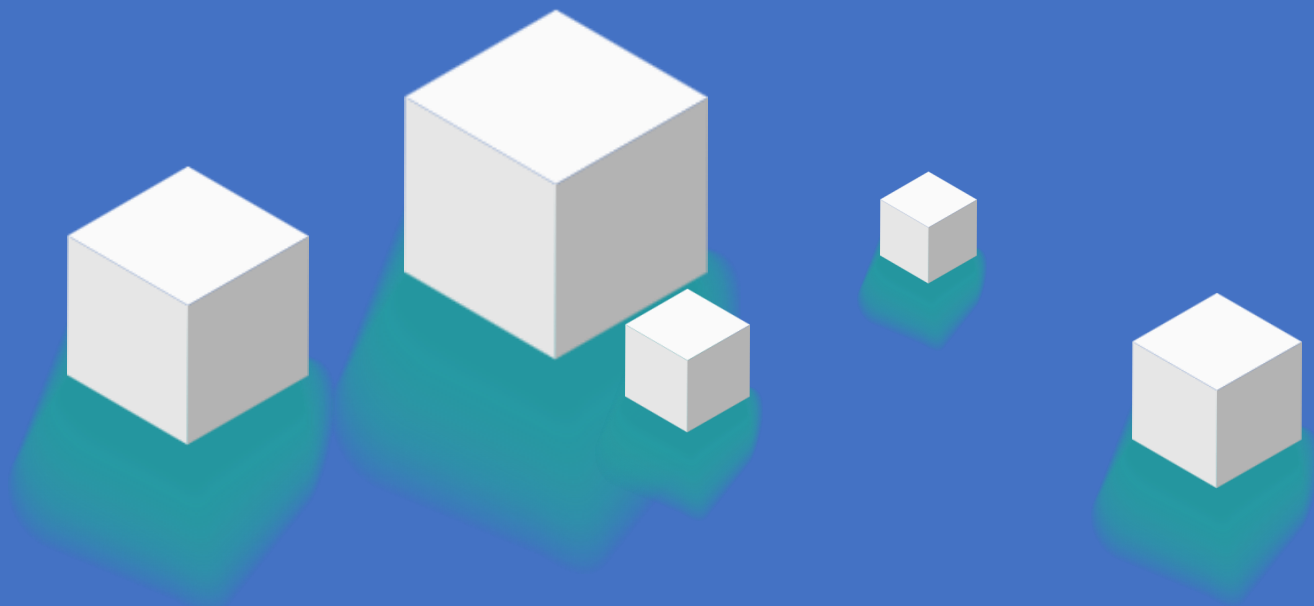


Function Calling与MCP



>> 今天的学习目标

Function Calling

- 什么是Function Calling
- CASE: 门票助手
- Qwen-Agent中的Function Calling

MCP使用

- 什么是MCP
- MCP 的核心概念
- CASE: 旅游攻略MCP
- CASE: Tavily搜索MCP
- CASE: 桌面TXT统计器 (MCP SDK使用)

Function Calling

Thinking: Function Calling在大模型中的作用是什么?

- 扩展模型能力

大模型本身无法直接操作外部系统（如数据库、计算工具），但通过调用预设函数，可以完成：

实时数据获取（天气、股价、新闻）

复杂计算（数学运算、代码执行）

操作外部系统（发送邮件、控制智能设备）

- 结构化输出

模型可将用户自然语言请求转化为结构化参数，传递给函数。例如：

用户说“明天北京天气如何？” → 模型调用 `get_weather(location="北京", date="2025-05-06")`

- 动态决策流程

模型可根据上下文决定是否/何时调用函数，甚至链式调用多个函数（如先查天气，再推荐穿搭）。

Function Call是大模型与真实世界交互的“桥梁”，从语言理解 => 具体行动

Function Calling

Thinking: Function Calling与MCP的区别?

维度	Function Calling	MCP
定位	模型厂商私有接口 (如OpenAI, Qwen)	开放协议 (类似 HTTP/USB-C)
扩展性	需为每个模型单独适配	一次开发, 多模型兼容
复杂性	适合简单、单次调用任务	支持多轮对话、复杂上下文管理
生态依赖	依赖特定模型 (如 GPT-4)	跨模型、跨平台 (如 Claude、Cursor)
安全性	依赖云端 API 密钥	支持本地化数据控制

Function Calling

Thinking: 已经有了MCP还需要Function Calling么?

简单、原子化任务使用Function Calling会更方便

- 查询天气 `get_weather(city="北京")`
- 计算数学公式 `calculate(expression="3+5")`
- 发送单条通知 `send_email(to="user@example.com")`

优势:

开发快捷: 无需配置 MCP Server, 直接通过模型 API 调用预定义函数。

低延迟: 单次请求-响应, 无需协议层开销。

MCP 可能成为主流, 但 Function Calling 作为底层能力仍将存在

CASE: 门票助手

CASE： 门票助手

TO DO： 搭建门票助手，可以对门票业务进行查询

2023年4、5、6月一日门票，二日门票的销量多少？ 帮我按照周进行统计

2023年7月的不同省份的入园人数统计

帮我查看2023年10月1-7日销售渠道订单金额排名

字段名	含义说明
order_time	订单日期
account_id	预定用户ID
gov_id	商品使用人身份证号
gender	使用人性别
age	年龄
province	使用人省份
SKU	商品SKU名
product_serial_no	商品ID
eco_main_order_id	订单ID
sales_channel	销售渠道
status	商品状态
order_value	订单金额
quantity	商品数量

tkr_orders数据表

CASE: 门票助手

Thinking: 如何使用Function Call, 整体的搭建流程是怎样的?

Step1. 系统初始化

- 设置系统 prompt, 描述门票表结构和常见查询需求。
- 注册 SQL 查询工具 (exc_sql), 用于执行数据查询。

Step2. 助手实例化

使用 Qwen-Agent 的 `Assistant` 类, 加载 LLM 配置、系统 prompt 和 function_list (只包含 exc_sql)。

Step3. 设置交互模式

选择WebUI 模式, 用户通过网页输入问题, 助手自动完成 SQL 查询并返回结果, 右侧可列出常见问题。

Step4. Function Call 机制

- 用户输入自然语言问题。
- LLM 解析意图并自动生成 SQL 查询语句。
- exc_sql 工具被自动调用, 执行 SQL 并返回查询结果。
- 结果通过终端或 WebUI 展示给用户。

CASE: 门票助手 (system_prompt)

```
system_prompt = ""我是门票助手，以下是关于门票订单表相关的字段，我可能会编写对应的SQL，对数据进行查询
```

```
-- 门票订单表
```

```
CREATE TABLE tkt_orders (  
  order_time DATETIME,      -- 订单日期  
  account_id INT,          -- 预定用户ID  
  gov_id VARCHAR(18),      -- 商品使用人ID (身份证号)  
  gender VARCHAR(10),     -- 使用人性别  
  age INT,                 -- 年龄  
  province VARCHAR(30),   -- 使用人省份  
  SKU VARCHAR(100),       -- 商品SKU名  
  product_serial_no VARCHAR(30), -- 商品ID  
  eco_main_order_id VARCHAR(20), -- 订单ID  
  sales_channel VARCHAR(20), -- 销售渠道  
  status VARCHAR(30),     -- 商品状态
```

Thinking: 是否能找到原始的数据表metadata, 是否有常用的术语需要提供, 方便后续撰写SQL

CASE: 门票助手 (exc_sql工具注册)

```
from qwen_agent.tools.base import BaseTool, register_tool
```

```
@register_tool('exc_sql')
```

```
class ExcSQLTool(BaseTool):
```

```
    """
```

```
    SQL查询工具，执行传入的SQL语句并返回结果。
```

```
    """
```

```
    description = '对于生成的SQL，进行SQL查询'
```

```
    parameters = [{
```

```
        'name': 'sql_input',
```

```
        'type': 'string',
```

```
        'description': '生成的SQL语句',
```

```
        'required': True
```

```
    ]
```

```
def call(self, params: str, **kwargs) -> str:
```

```
    import json
```

```
    args = json.loads(params)
```

```
    sql_input = args['sql_input']
```

```
    database = args.get('database', 'ubr')
```

```
    # 创建数据库连接
```

```
    engine = .....
```

```
    try:
```

```
        df = pd.read_sql(sql_input, engine)
```

```
        # 返回前10行，防止数据过多
```

```
        return df.head(10).to_markdown(index=False)
```

```
    except Exception as e:
```

```
        return f"SQL执行出错: {str(e)}"
```

Thinking: 这个地方是否可以通过 用户的需求，自适应进行调整？

CASE: 门票助手 (Assistant 初始化与 function_list)

```
def init_agent_service():
    """初始化门票助手服务"""
    llm_cfg = {
        'model': 'qwen-turbo-2025-04-28',
        'timeout': 30,
        'retry_count': 3,
    }
    try:
        bot = Assistant(
            llm=llm_cfg,
            name='门票助手',
            description='门票查询与订单分析',
            system_message=system_prompt,
            function_list=['exc_sql'], # 只传工具名字字符串
```

```
        print("助手初始化成功! ")
        return bot
    except Exception as e:
        print(f"助手初始化失败: {str(e)}")
        raise
```

CASE: 门票助手 (Web 图形界面)

```
def app_gui():
    """图形界面模式, 提供 Web 图形界面"""
    ...
    bot = init_agent_service()
    chatbot_config = {
        'prompt.suggestions': [
            '2023年4、5、6月一日门票, 二日门票的销量多少? 帮
我按照周进行统计',
            '2023年7月的不同省份的入园人数统计',
            '帮我查看2023年10月1-7日销售渠道订单金额排名',
        ]
    }
```

```
WebUI(
    bot,
    chatbot_config=chatbot_config
).run()
```

function call 相关逻辑全部通过 Assistant 的 function_list 机制与工具注册实现。
exc_sql 工具负责 SQL 查询, 注册后可被 LLM调用。
支持终端(app_tui)和 WebUI 两种交互方式。

CASE: 门票助手

2023年4、5、6月一日门票，二日门票的销量多少？帮我按照周进行统计



user

▶ Start calling tool "exc_sql" ...

▶ Finished tool calling.



以下是2023年4、5、6月一日门票和二日门票的周销量统计：

周数	一日门票销量	二日门票销量
2023-13	3367	2732
2023-14	19444	14267
2023-15	20123	14888
2023-16	19399	15007
2023-17	19489	14603
2023-18	19933	15249
2023-19	19422	14783
2023-20	19594	14428
2023-21	19163	14339
2023-22	19573	14297

▼ Start calling tool "exc_sql" ...

```
{ "sql_input": "SELECT DATE_FORMAT(order_time, '%Y-%U') AS week, \n SUM(CASE WHEN SKU LIKE 'Universal Studios Beijing One-Day%' THEN quantity ELSE 0 END) AS one_day_ticket_sales, \n SUM(CASE WHEN SKU LIKE 'USB%' THEN quantity ELSE 0 END) AS two_day_ticket_sales \nFROM tkt_orders \nWHERE order_time BETWEEN '2023-04-01' AND '2023-06-30' \nGROUP BY week \nORDER BY week;" }
```

▼ Finished tool calling.

week	one_day_ticket_sales	two_day_ticket_sales
2023-13	3367	2732
2023-14	19444	14267
2023-15	20123	14888
2023-16	19399	15007
2023-17	19489	14603
2023-18	19933	15249
2023-19	19422	14783
2023-20	19594	14428
2023-21	19163	14339
2023-22	19573	14297



门票助手

如果需要更详细的分析或进一步的数据，请告诉我！



CASE: 门票助手

2023年7月的不同省份的入园人数统计



user

▶ Start calling tool "exc_sql" ...

▶ Finished tool calling.

以下是2023年7月不同省份的入园人数统计:

省份	入园人数
北京市	99893
江苏省	47290
河北省	46602
山东省	43762
广东省	42601
上海市	40782
浙江省	38042
辽宁省	24910
湖北省	24231
四川省	22672



▼ Start calling tool "exc_sql" ...

```
{"sql_input": "SELECT province, SUM(quantity) AS visitor_count \nFROM tkt_orders \nWHERE order_time BETWEEN '2023-07-01' AND '2023-07-31' \nGROUP BY province \nORDER BY visitor_count DESC;"}
```

▼ Finished tool calling.

province	visitor_count
北京市	99893
江苏省	47290
河北省	46602
山东省	43762
广东省	42601
上海市	40782
浙江省	38042
辽宁省	24910
湖北省	24231
四川省	22672

如果需要更详细的分析或进一步的数据, 请告诉我!



门票助手

CASE: 门票助手

帮我查看2023年10月1-7日销售渠道订单金额排名



user

▼ Start calling tool "exc_sql" ...

```
{"sql_input": "SELECT sales_channel, SUM(order_value) AS total_order_value \nFROM tkt_orders \nWHERE order_time BETWEEN '2023-10-01' AND '2023-10-07' \nGROUP BY sales_channel \nORDER BY total_order_value DESC;"}
```

▼ Finished tool calling.

sales_channel	total_order_value
B2B_OTA	2.0915e+07
B2C_UBRAPP	1.03013e+07
B2B_TTAGIT	3.51182e+06

以下是2023年10月1日至7日销售渠道的订单金额排名：

销售渠道	订单金额
B2B_OTA	20,915,000
B2C_UBRAPP	10,301,300
B2B_TTAGIT	3,511,820

如果需要更详细的分析或进一步的数据，请告诉我！

TO DO: 快来搭建你的业务助手吧!



门票助手



打卡：业务助手



练习使用LLM/Agent + Function Calling使用，可以使用Qwen3或Qwen-Agent

如果你在本地有MySQL，可以基于自己的数据，进行业务查询

- 1) 某个月份的销量是多少？
- 2) 相比于上个月，这个月的销量环比增长多少？
- 3) 不同省份的销售额是多少？
- 4) 某个时间段，销售金额Top3的渠道是哪些？

CASE： 门票助手（可视化图表）

Thinking： 如何在exc_sql查询数据之后，将数据进行可视化图表呈现？

方法1： 编写新的函数 plot_data，传入 Markdown的图表，进行可视化

方法2： 在原有exc_sql函数基础上，增加 plot_data的功能，返回结果包括：数据表markdown 以及可视化的图表png

Thinking： 方法1 的问题是什么？

Markdown传参可能更大，另外要绘制的x, y参数也不一定能传递准确

编写Markdown绘图有一定的难度，也需要先将Markdown转化为df (dataframe格式) 然后再进行绘制

在qwen-agent中，很难保存中间的 df，不同用户实例之间的维护成本可能较高

=> 采用方法2

可视化图表：实现步骤

在传统实现中，数据查询和可视化通常是分开的两个步骤（工具）：

- 先执行 SQL 查询获取数据
- 再调用可视化工具进行图表绘制

我们的优化是将两者集成到一个工具中，实现：

- 一次调用，完成查询和可视化
- 自动推断，图表类型和字段映射（x轴、y轴）
- 结果双输出，同时返回表格和图表

可视化图表：实现步骤

Step1, SQL查询获取数据

```
# 执行SQL查询
```

```
df = pd.read_sql(sql_input, engine)
```

```
# 生成markdown表格
```

```
md = df.head(10).to_markdown(index=False)
```

Step2, 自动推断图表字段

```
# 自动推断x/y字段
```

```
x_candidates = df.select_dtypes(include=['object']).columns.tolist()
```

```
if not x_candidates:
```

```
    x_candidates = df.columns.tolist()
```

```
x = x_candidates[0]
```

```
y_candidates = df.select_dtypes(include=['number']).columns.tolist()
```

```
y_fields = y_candidates
```

字段推断逻辑：

x轴字段： 优先选择第一个字符串类型（object）的列，如日期、分类名称等

y轴字段： 选择所有数值类型的列，支持多系列数据展示

可视化图表：实现步骤

Step3, 柱状图绘制

```
plt.figure(figsize=(8, 5))  
  
bar_width = 0.35 if len(y_fields) > 1 else 0.6  
  
x_labels = df[x].astype(str)  
x_pos = range(len(df))  
  
for idx, y_col in enumerate(y_fields):  
    plt.bar([p + idx*bar_width for p in x_pos], df[y_col], width=bar_width, label=y_col)
```

绘图逻辑：

- 创建适当大小的图表
- 根据y轴字段数量调整柱宽
- 支持多系列数据的并列柱状图
- 每个y轴字段绘制一组柱子
- 自动错开位置，避免柱子重叠

可视化图表：实现步骤

Step4, 图表样式设置

```
plt.xlabel(x)
```

```
plt.ylabel(','.join(y_fields))
```

```
plt.title(f"{' & '.join(y_fields)} by {x}")
```

```
plt.xticks([p + bar_width*(len(y_fields)-1)/2 for p in x_pos], x_labels, rotation=45, ha='right')
```

```
plt.legend()
```

```
plt.tight_layout()
```

样式设置：

- 设置x轴、y轴标签
- 自动生成图表标题
- x轴标签45度倾斜，避免重叠
- 添加图例，区分多系列数据
- 调整图表布局，确保所有元素可见

可视化图表：实现步骤

Step5, 图表保存与返回

自动创建目录

```
save_dir = os.path.join(os.path.dirname(__file__), 'image_show')
```

```
os.makedirs(save_dir, exist_ok=True)
```

生成唯一文件名

```
filename = f'bar_{int(time.time()*1000)}.png'
```

```
save_path = os.path.join(save_dir, filename)
```

```
plt.savefig(save_path)
```

```
plt.close()
```

```
img_path = os.path.join('image_show', filename)
```

```
img_md = f'![柱状图]({img_path})'
```

```
return f"{md}\n\n{img_md}"
```

保存与返回逻辑：

- 自动创建图片保存目录
- 生成基于时间戳的唯一文件名
- 保存图片到本地
- 生成markdown格式的图片引用
- 返回"表格+图片"组合结果

CASE： 门票助手 （可视化图表）

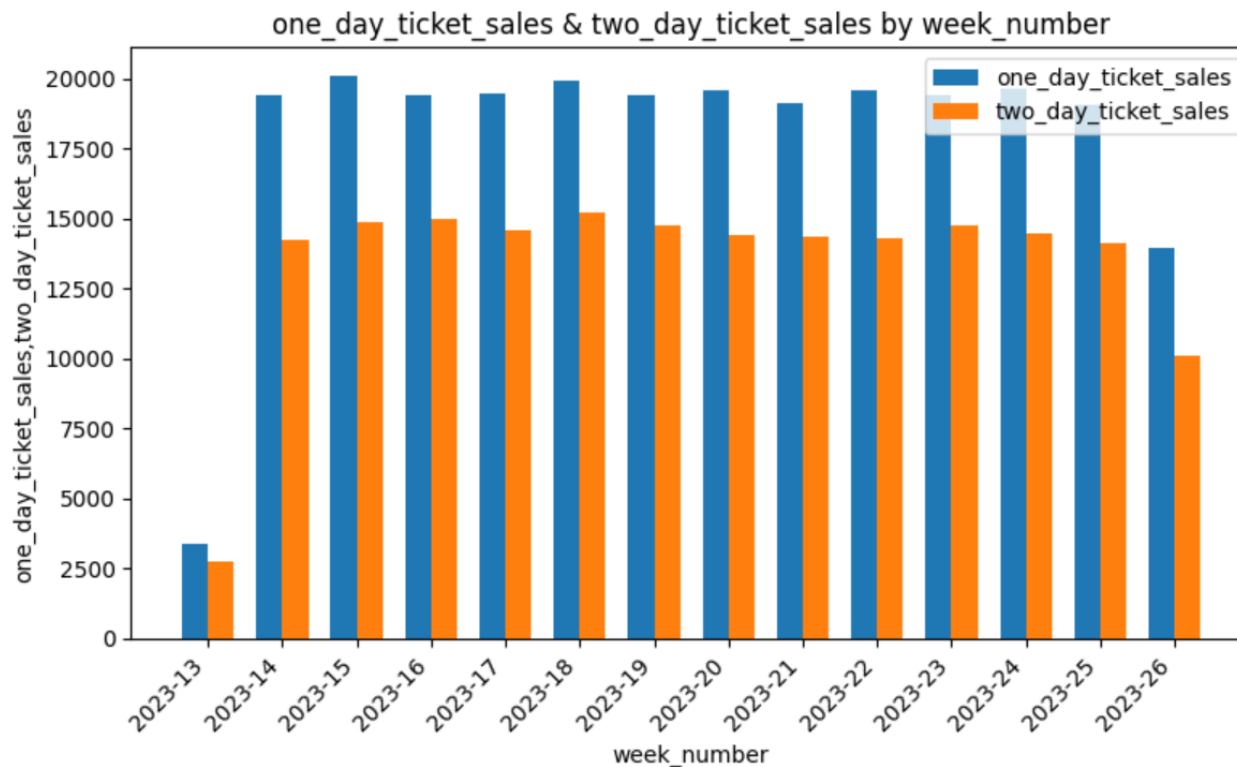
2023年4、5、6月一日门票，二日门票的销量多少？ 帮我按照周进行统计

▶ Start calling tool "exc_sql" ...

▶ Finished tool calling.

以下是2023年4、5、6月一日门票和二日门票的周销量统计：

week_number	一日门票销量	二日门票销量
2023-13	3367	2732
2023-14	19444	14267
2023-15	20123	14888
2023-16	19399	15007
2023-17	19489	14603
2023-18	19933	15249
2023-19	19422	14783
2023-20	19594	14428
2023-21	19163	14339
2023-22	19573	14297



CASE： 门票助手（可视化图表）

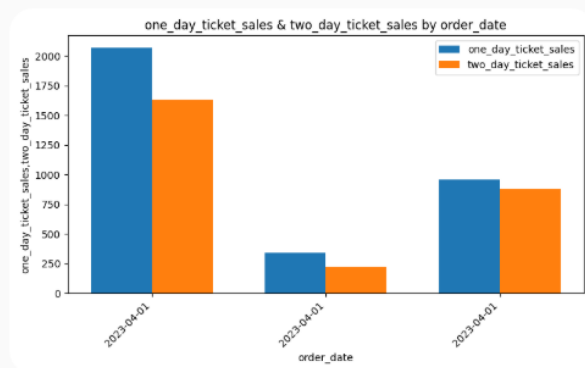
我看到第13周数据有异常，帮我按照天、渠道的维度进行统计，给我他们的表现

► Start calling tool "exc_sql" ...

► Finished tool calling.

以下是2023年第13周（即2023-04-01）一日门票和二日门票按天和销售渠道的销量统计：

order_date	sales_channel	一日门票销量	二日门票销量
2023-04-01	B2B_OTA	2069	1633
2023-04-01	B2B_TTAGIT	339	222
2023-04-01	B2C_UBRAPP	959	877



Thinking: 这里为什么提取的数据不对?

WHERE

```
order_time BETWEEN '2023-04-01' AND '2023-06-30'  
AND DATE_FORMAT(order_time, '%Y-%U') = '2023-13'
```

CASE： 门票助手（可视化图表）

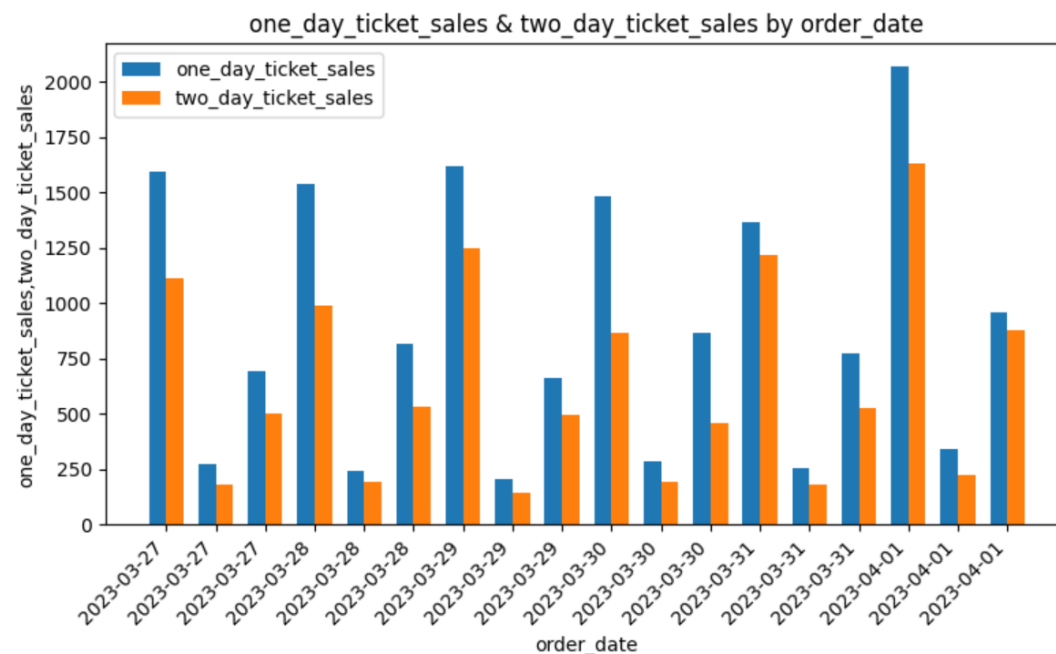
我看到第13周（2023年3月27日-4月2日）数据有异常，帮我按照天、渠道的维度进行统计，给我他们的表现

► Start calling tool "exc_sql" ...

► Finished tool calling.

以下是2023年3月27日至4月2日（第13周）一日门票和二日门票按天和销售渠道的销量统计：

order_date	sales_channel	一日门票销量	二日门票销量
2023-03-27	B2B_OTA	1596	1111
2023-03-27	B2B_TTAGIT	274	183
2023-03-27	B2C_UBRAPP	694	499
2023-03-28	B2B_OTA	1537	991
2023-03-28	B2B_TTAGIT	245	190
2023-03-28	B2C_UBRAPP	816	535
2023-03-29	B2B_OTA	1618	1249
2023-03-29	B2B_TTAGIT	205	144
2023-03-29	B2C_UBRAPP	663	493
2023-03-30	B2B_OTA	1486	868



Thinking： 如何将渠道显示出来？

CASE： 门票助手（可视化图表）

Thinking： 如何将渠道显示出来？

采用分组与透视

如果存在 object_columns， 则用 pd.pivot_table 以 x 轴为 index， object_columns 为 columns， num_columns 为 values， 自动聚合

支持多 object、 多数值列， 自动生成多级分组

```
def generate_chart_png(df_sql, save_path):  
    .....
```

Version更新：

assistant_ticket_bot-1： 实现了function call调用exc_sql

assistant_ticket_bot-2： 添加了基本的绘制图表功能

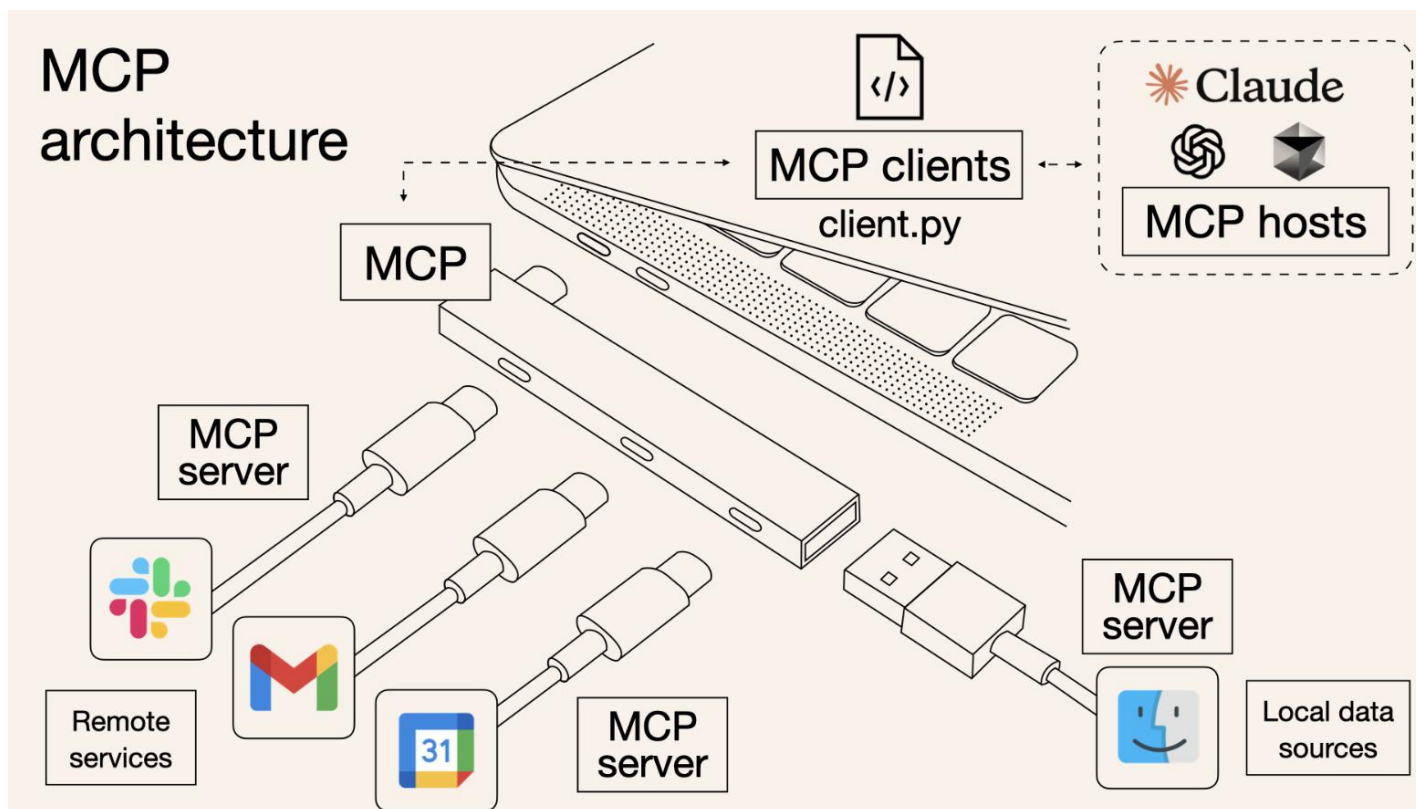
assistant_ticket_bot-3： 进一步完善绘制图表功能， 支持多类别变量的透视图可视化

MCP使用

什么是MCP

Thinking: 什么是MCP?

Model Context Protocol (MCP) 是由 Anthropic 公司于 2024 年 11 月推出的一种开放协议标准，目的在于标准化 LLM 与外部数据源、工具及服务之间的交互方式。MCP 被广泛类比为“AI 领域的 USB-C 接口”



MCP与Function Calling的区别

类别	MCP (Model Context Protocol)	Function Calling
性质	协议	功能
范围	通用 (多数据源、多功能)	特定场景 (单一数据源或功能)
目标	统一接口, 实现互操作	扩展模型能力
实现	基于标准协议	依赖于特定模型实现
开发复杂度	低: 通过统一协议实现多源兼容	高: 需要为每个任务单独开发函数
复用性	高: 一次开发, 可多场景使用	低: 函数通常为特定任务设计
灵活性	高: 支持动态适配和扩展	低: 功能扩展需要额外开发
常见场景	复杂场景, 如跨平台数据访问与整合	简单任务, 如天气查询、商品推荐等

MCP与Function Calling的区别

MCP 的核心概念

1. 架构与组件

MCP 采用 客户端-服务器 (Client-Server) 架构, 主要包括以下核心组件:

- **MCP Host:** 运行 AI 模型的环境, 如 Claude Desktop、Cursor IDE 等。
- **MCP Client:** 嵌入在 Host 中的组件, 负责发起请求并与 MCP Server 通信。
- **MCP Server:** 轻量级服务, 提供特定功能 (如数据查询、API 调用等), 供 AI 模型调用。

MCP 的核心概念

2. 核心功能

MCP 提供三种关键能力：

- **Resources (知识扩展)**：提供结构化数据（如数据库、文档）以增强 AI 的上下文理解。
- **Tools (工具调用)**：允许 AI 执行外部操作（如发送邮件、查询 GitHub、调用智能合约等）。
- **Prompts (提示模板)**：预定义的指令模板，优化 AI 的任务执行。

CASE: 旅游攻略MCP

CASE：旅游攻略MCP

Step1, 获取高德地图 MCP 的使用授权 (免费) <https://lbs.amap.com/>

Step2, 注册成为开发者 => 创建应用 => 应用服务平台 选择 Web服务 => 获取 API KEY



The screenshot shows the Amap Open Platform developer console. The top navigation bar includes the Amap logo, the text "高德开放平台 lbs.amap.com", and the word "控制台". On the right side of the navigation bar are links for "应用", "账号信息", and "订单发票". A left sidebar contains a menu with items: "首页", "应用管理", "我的应用", "回收站", "流量分析", and "账号中心". The main content area is titled "我的应用" and displays a list of applications. The first application is "出行", created on 2025/4/26. Below it is a table with columns: "Key 名称", "Key ② 商用说明", "安全密钥 (点击查看安全密钥使用说明)", and "绑定服务". The "出行" application row shows a key name of "出行", a key value of "58fa1b02f183bdcad7cef9b29f9360c1" (highlighted with a red box), a dash for the security key, and "Web服务" for the bound service (also highlighted with a red box). Below this are two more application entries: "天气" (created 2024/3/22) and "地理位置查询" (created 2021/8/16).

Key 名称	Key ② 商用说明	安全密钥 (点击查看安全密钥使用说明)	绑定服务
出行	58fa1b02f183bdcad7cef9b29f9360c1	—	Web服务

CASE：旅游攻略MCP

Step3, 使用具有MCP功能的客户端进行调用

比如：Cursor具有MCP功能，可以在Agent模式下输入

用高德MCP，做上海一天旅游攻略

我会使用高德地图 MCP 来帮您规划上海一天的旅游攻略。首先让我们搜索上海的主要景点：

> Called MCP tool maps_text_search ✓

让我们继续搜索另一个著名景点东方明珠：

> Called MCP tool maps_text_search ✓

让我们获取豫园的信息：

> Called MCP tool maps_text_search ✓

我会使用高德地图 MCP 来帮您规划上海一天的旅游攻略。首先让我们搜索上海的主要景点：

Called MCP tool maps_text_search ✓

Parameters:

```
{  
  "keywords": "上海外滩"  
  "city": "上海",  
  "types": "景点"  
}
```

Result:

根据获取到的信息，我为您规划一条上海一日游路线：

1. 上午：豫园（上海豫园）
 - 地址：福佑路168号
 - 特色：明代古典园林，可以体验江南园林文化
 - 建议游览时间：2-3小时
 - 可以同时游览豫园老街，品尝上海特色小吃
2. 中午：外滩
 - 地址：中山东一路49号
 - 特色：欣赏外滩建筑群，感受上海十里洋场的历史风貌
 - 建议游览时间：1-2小时
 - 可以在外滩附近的餐厅享用午餐
3. 下午：东方明珠
 - 地址：世纪大道1号
 - 特色：上海地标性建筑，可以登塔俯瞰整个上海
 - 建议游览时间：2-3小时
 - 可以在旋转餐厅享用晚餐，欣赏夜景

Cursor, Cherry Studio集成了MCP服务

CASE: 旅游攻略MCP

让我们看看天气情况:

> Called MCP tool `maps_weather` ✓

让我们继续搜索下午可以游览的景点:

> Called MCP tool `maps_text_search` ✓

> Called MCP tool `maps_search_detail` ✓

让我们看看从外滩到南京路步行街的步行路线:

> Called MCP tool `maps_direction_walking` ✓

让我们看看天气情况:

Called MCP tool `maps_weather` ✓

```
"forecasts": [  
  {  
    "date": "2025-04-26",  
    "week": "6",  
    "dayweather": "阴",  
    "nightweather": "阴",  
    "daytemp": "23",  
    "nighttemp": "15",  
    "daywind": "东南",  
    "nightwind": "东南",  
    "daypower": "1-3",
```

让我们看看从外滩到南京路步行街的步行路线:

Called MCP tool `maps_direction_walking` ✓

Result:

```
{  
  "route": {  
    "origin":  
      "121.492156,31.233462",  
    "destination":  
      "121.479494,31.235983",  
    "paths": [  
      {  
        "distance": "1513"
```

CASE: 旅游攻略MCP

在Cursor中设置 mcp.json, 然后在 Cursor Settings中打开

```
{  
  "mcpServers": {  
    "amap-maps": {  
      "command": "npx",  
      "args": [  
        "-y",  
        "@amap/amap-maps-mcp-server"  
      ],  
      "env": {  
        "AMAP_MAPS_API_KEY": "你的key"  
      }  
    }  
  }  
}
```

The screenshot shows the Cursor Settings interface. At the top, there are tabs for 'txt_counter.py', 'Cursor Settings', and 'mcp.json'. The 'Cursor Settings' panel is open, showing a sidebar with categories: General, Features, Models, Rules, MCP, and Beta. The 'MCP Servers' section is active, displaying a list of servers. The 'amap-maps' server is highlighted, and its toggle switch is turned on (indicated by a red box). Below the server list, there are several tool names and a command: 'npx -y @amap/amap-maps-mcp-server'. The 'Cursor Settings' panel also shows other settings, including 'Default new chat mode' (set to 'Agent'), 'Auto-refresh chats' (checked), 'Auto-scroll to bottom' (checked), 'Auto-apply to files outside context in Manual mode' (checked), and 'Include project structure' (unchecked). At the bottom, the 'Enable auto-run mode' checkbox is checked (indicated by a red box).

CASE：旅游攻略MCP

TO DO：制作大连一日游旅行规划，并生成网页

Step1，制作旅行规划表提示词

```
1 # 旅行规划表设计提示词
2
3 你是一位优秀的平面设计师和前端开发工程师，具有丰富的旅行信息可视化经验，曾为众多知名旅游平台设计过清晰实用的旅行规划表。现在需要为我创建一个A4纸张大小的旅行规划表，适合打印出来随身携带使用。请使用HTML、CSS和JavaScript代码实现以下要求：
4
5 ---
6
7 ## 基本要求
8
9 ### 尺寸与基础结构
10 - 严格符合A4纸尺寸（210mm×297mm），比例为1:1.414
11 - 适合打印的设计，预留适当的打印边距（建议上下左右各10mm）
12 - 采用单页设计，所有重要信息必须在一页内完整呈现
13 - 信息分区清晰，使用网格布局确保整洁有序
14 - 打印友好的配色方案，避免过深的背景色和过小的字体
15 | Ctrl+L to chat, Ctrl+K to generate
16 ### 技术实现
17 - 使用打印友好的CSS设计
18 - 提供专用的打印按钮，优化打印样式
```

旅行规划.md

@旅行规划.md 根据这个，生成大连一日游的网页

我来帮您规划大连一日游的行程，并生成一个网页。首先让我们搜索大连的主要景点：

> Called MCP tool maps_text_search ✓

> Called MCP tool maps_text_search ✓

让我们获取这些景点的详细信息：

> Called MCP tool maps_search_detail ✓

> Called MCP tool maps_search_detail ✓

让我们查看大连的天气情况：

> Called MCP tool maps_weather ✓

CASE：旅游攻略MCP

现在我来创建一个网页文件：

```
<> dalian_tour.html +157 -1 ✓  
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>大连一日游攻略</title>  
  <style>  
    body {  
      font-family: 'Microsoft YaHei'  
      line-height: 1.6;  
      margin: 0;  
      padding: 0;  
      background-color: #f5f5f5;  
    }  
  </style>  
</head>  
</html>
```

我已经创建了一个美观且信息丰富的大连一日游网页。这个网页包含以下主要内容：

1. 响应式设计，适配各种设备
2. 清晰的布局结构
3. 主要内容区块：
 - 天气预报
 - 行程安排（星海广场和发现王国）
 - 交通建议
 - 实用贴士



dalian_tour.html > html > head > style > body

```
1 <!DOCTYPE html>  
2 <html lang="zh-CN">  
3 <head>  
4   <meta charset="UTF-8">  
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6   <title>大连一日游攻略</title>  
7   <style>  
8     body {  
9       font-family: 'Microsoft YaHei', sans-serif;  
10      line-height: 1.6;  
11      margin: 0;  
12      padding: 0;  
13      background-color: #f5f5f5;  
14    }  
15    .container {  
16      max-width: 1200px;  
17      margin: 0 auto;  
18      padding: 20px;  
19    }  
20    header {  
21      background-color: #1e88e5;
```

CASE：旅游攻略MCP

大连一日游完美攻略

探索浪漫之都，体验海滨风情

天气预报

今日天气：多云转雷阵雨

温度：11°C - 21°C

风向：西南风

风力：5级

温馨提示：

- 建议携带雨具，以防下雨
- 注意防风保暖，气温变化较大
- 户外活动请注意天气变化

交通建议

星海广场交通：

- 地铁：乘坐1号线至星海广场站
- 公交：16路、18路、22路、23路等多条线路可达

前往发现王国：

- 乘坐地铁1号线至终点站后转乘公交
- 也可选择打车或包车前往，车程约40分钟

行程安排

上午：星海广场 (9:00-11:30)

地址：大连市沙河口区中山路

开放时间：24小时

推荐游览时间：2.5小时

特色亮点：

- 亚洲最大的城市广场之一
- 标志性音乐喷泉表演
- 绝美海景观赏点
- 周边众多特色餐厅

下午：海昌发现王国 (13:00-17:30)

地址：大连市金石滩国家旅游度假区金石路36号

门票：60元

推荐游览时间：4-5小时

特色亮点：

- 大型主题乐园，众多刺激游乐项目
- 精彩的海洋动物表演
- 独特的海洋文化体验
- 适合全家游玩的综合性乐园

CASE：旅游攻略MCP（Cherry Studio）

Thinking：如何在Cherry Studio中配置 MCP服务？

Step1，确保MCP服务工具安装好，这里会用到 UV, Bun 工具

<https://github.com/oven-sh/bun/releases>

```
powershell -c "irm bun.sh/install.ps1|iex"
```

<https://github.com/astral-sh/uv/releases>

```
powershell -ExecutionPolicy Bypass -c "irm
```

```
https://github.com/astral-
```

```
sh/uv/releases/download/0.6.17/uv-installer.ps1 | iex"
```



CASE：旅游攻略MCP (Cherry Studio)

Step2, 搜索MCP

输入 @amap 找到高德地图相关的MCP服务

添加对应的MCP服务

NPX 包列表

搜索并添加 npm 包作为 MCP 服务

@amap

@mcpmarket @modelcontextprotocol @gongrzhe

amap-maps-mcp-server v0.0.8 +

MCP server for using the AMap Maps API

用法: npx @amap/amap-maps-mcp-server

<https://www.npmjs.com/package/@amap/amap-maps-mcp-server>

amap-maps-mcp-server 保存

* 名称
amap-maps-mcp-server

描述
MCP server for using the AMap Maps API

* 类型
 STUDIO SSE

* 命令
npx

* 参数 ⓘ
-y
@amap/amap-maps-mcp-server

环境变量 ⓘ
AMAP_MAPS_API_KEY=58fa1b02f183bdcad7cef9b29f9360c1

这里先点击保存, 然后再打开服务, 变绿说明服务可以正常使用

CASE：旅游攻略MCP（Cherry Studio）

Step3，配置LLM（需要具备工具功能）

比如添加 qwen-turbo功能

编辑模型

* 模型 ID ?

qwen-turbo

模型名称 ?

qwen-turbo

分组名称 ?

qwen-turbo

保存

更多设置 ^

选择模型类型:

视觉 嵌入 推理 工具

选择qwen-turbo模型，然后打开MCP服务器，进行提问

用高德MCP，做上海一天旅游攻略

MCP 服务器



fetch-server



amap-maps-mcp-server



MCP server for using the AMap Maps API 用法: npx @a...

这里先点击保存，然后再打开服务，变绿说明服务可以正常使用

CASE：旅游攻略MCP（Cherry Studio）

qwen-turbo | 阿里云

04/26 10:34

maps_text_search | 已完成 ✓

好的，为了制定上海的一日游攻略，我们可以从以下几个方面入手：首先确定几个主要的旅游景点，然后规划合理的交通方式，最后安排好吃饭和休息的时间。以下是步骤：

- 选择旅游景点：**我们可以选择一些具有代表性的景点，例如外滩、东方明珠、豫园等。
- 规划交通：**我们可以使用高德地图的路径规划功能来规划从一个景点到另一个景点的交通方式。
- 安排时间：**考虑到每个景点的游览时间和交通时间，合理安排整个行程。现在我们开始第一步，选择几个主要的旅游景点并获取它们的经纬度。让我们使用 `fQWbL5e0io_LJQfxomkBv4` 工具来搜索这些景点的位置信息。根据您的要求，为您准备了上海一日游的景点推荐：
- 上海豫园
 - 地址：福佑路168号
 - 图片：
- 东方明珠广播电视塔
 - 地址：世纪大道1号(陆家嘴地铁站1号入口步行220米)
 - 图片：!
- 南京路步行街
 - 地址：南京东路558号



• 图片：

您可以根据这些景点的具体位置和开放时间规划您的行程。希望您在上海玩得开心！

maps_text_search | 已完成 ✓

```
{
  "params": {
    "city": "上海",
    "keywords": "外滩,东方明珠,豫园"
  },
  "response": {
    "content": [
      {
        "type": "text",
        "text": "{\n \"suggestion\": {\n \"keywords\": [],\n \"ciytes\": []\n },\n \"pois\": {\n {\n \"id\":\n \\\"B00155MF55\\\", \n \"name\": \"上海豫园\", \n \"address\": \"福佑路168号\", \n \"typecode\": \"110202\", \n \"photos\": {\n \"title\": [], \n \"url\":\n \\\"http://store.is.autonavi.com/showpic/faeb0264854cd82fbb315cb2ccacea0f\\\", \n \"provider\": []\n }\n }, \n {\n \"id\": \"B00150F6D6\", \n \"name\": \"东方明珠广播电视塔\", \n \"address\": \"世纪大道1号(陆家嘴地铁站1号入口步行220米)\", \n \"typecode\": \"110202\", \n \"photos\": {\n \"title\": [], \n \"url\":
```

LLM会自主选择工具进行调用

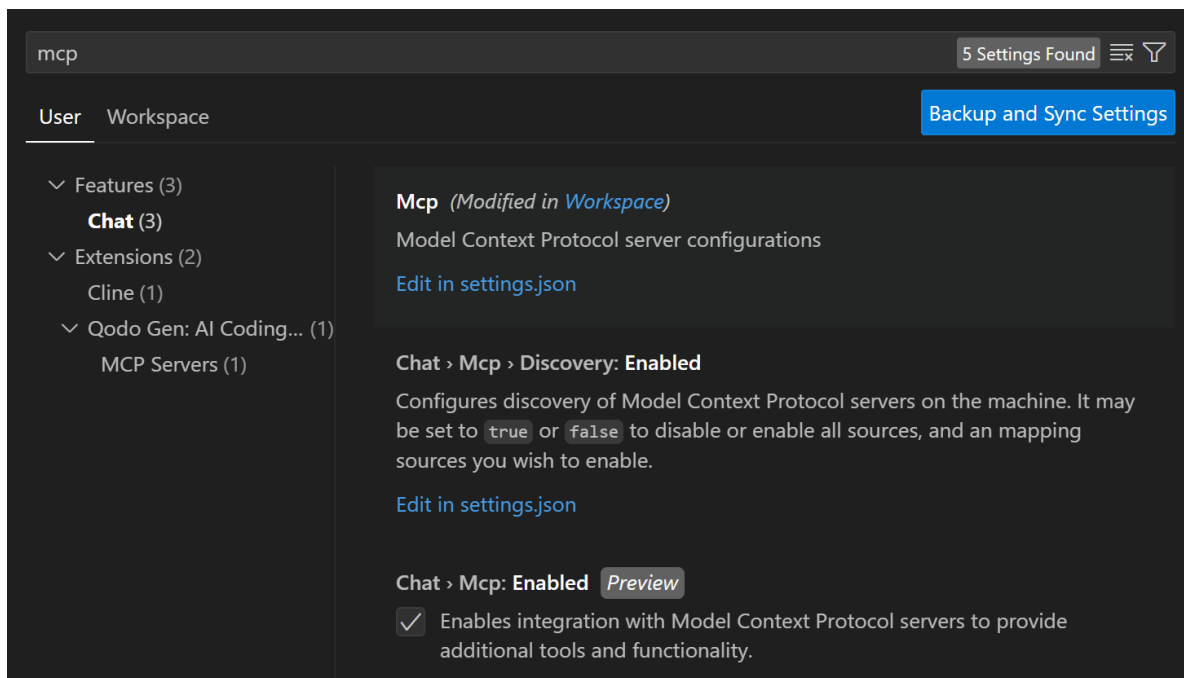
CASE: 旅游攻略MCP (Github Copilot)

Thinking: 如何在Github Copilot中配置 MCP服务?

Step1, 打开VS Studio, 配置mcp servers

Preferences => Settings => 搜索 mcp

打开Edit in settings.json, 可以看到具体的内容, 找到mcp相关的部分, 并添加



CASE: 旅游攻略MCP (Github Copilot)

```
"mcpServers": {
  "amap-maps": {
    "command": "npx",
    "args": [
      "-y",
      "@amap/amap-maps-mcp-server"
    ],
    "env": {
      "AMAP_MAPS_API_KEY": "你的API KEY"
    }
  }
}
```



```
46 "chat.mcp.discovery.enabled": true,
47 "mcpServers": {
48   "amap-maps": {
49     "command": "npx",
50     "args": [
51       "-y",
52       "@amap/amap-maps-mcp-server"
53     ],
54     "env": {
55       "AMAP_MAPS_API_KEY": ""
56     }
57   }
58 }
59 }
60 }
```

CASE: 旅游攻略MCP (Github Copilot)

✓ Select tools that are available to chat 15 Selected OK

- ✓ MCP Server: amap-maps From Cursor (Global) (Running)
 - ✓ maps_regeocode 将一个高德经纬度坐标转换为行政区划地址信息
 - ✓ maps_geo 将详细结构化地址转换为经纬度坐标。支持对地标性名胜景区、建筑物名称解析为经纬...
 - ✓ maps_ip_location IP 定位根据用户输入的 IP 地址, 定位 IP 的所在位置
 - ✓ maps_weather 根据城市名称或者标准adcode查询指定城市的天气
 - ✓ maps_search_detail 查询关键词搜或者周边搜获取到的POI ID的详细信息
 - ✓ maps_bicycling 骑行路径规划用于规划骑行通勤方案, 规划时会考虑天桥、单行线、封路等情况。...
 - ✓ maps_direction_walking 步行路径规划 API 可以根据输入起点终点经纬度坐标规划100km 以内的步...
 - ✓ maps_direction_driving 驾车路径规划 API 可以根据用户起终点经纬度坐标规划以小客车、轿车通...
 - ✓ maps_direction_transit_integrated 公交路径规划 API 可以根据用户起终点经纬度坐标规划综合各...
 - ✓ maps_distance 距离测量 API 可以测量两个经纬度坐标之间的距离,支持驾车、步行以及球面距离测量
 - ✓ maps_text_search 关键词搜, 根据用户传入关键词, 搜索出相关的POI
 - ✓ maps_around_search 周边搜, 根据用户传入关键词以及坐标location, 搜索出radius半径范围的POI
- ✓ MCP Server: fetch From CASE-MCP Demo-copilot/.vscode/mcp.json (Running)
 - ✓ fetch Fetches a URL from the internet and optionally extracts its contents as markdown. Although...


⌵ ⚠️ 13 Add Context... settings.json Current file 🔍

Edit files in your workspace in agent mode

🎤 Agent Claude 3.7 Sonnet || ⏸️

CASE: 旅游攻略MCP (Github Copilot)

用高德MCP, 做上海一天旅游攻略

 GitHub Copilot

我将帮您制作上海一天旅游攻略, 可以利用高德地图的相关功能来规划路线和景点。我会使用高德地图的MCP服务来获取景点信息、路线规划等。

首先, 让我们搜索上海的热门旅游景点, 然后规划一条合理的一日游路线。

> Ran `maps_text_search`

我已经获取到上海的主要旅游景点信息。根据高德地图的数据, 上海有许多值得游览的地方, 包括东方明珠、上海豫园、陆家嘴、静安寺等知名景点, 以及七宝古镇、朱家角等特色古镇。

现在, 我将为您制定一个上海一日游的合理路线。我会选择市中心区域的几个代表性景点, 这样可以减少交通时间, 让您在一天内游览更多地方。

让我们选择几个经典景点并获取它们的详细信息:

> Run `maps_search_detail`



```

  Ran maps_search_detail
  {
    "id": "B00155MF55"
  }
  {
    "id": "B00155MF55",
    "name": "上海豫园",
    "location": "121.492497,31.227714",
    "address": "福佑路168号",
    "business_area": "城隍庙",
    "city": "上海市",
    "type": "风景名胜;风景名胜;国家

```

CASE: 旅游攻略MCP (Github Copilot)

```
> Ran maps_text_search
```

我已经获取到上海的主要旅游景点信息。根据高德地图的数据，上海有许多值得游览的地方，包括东方明珠、上海豫园、陆家嘴、静安寺等知名景点，以及七宝古镇、朱家角等特色古镇。

现在，我将为您制定一个上海一日游的合理路线。我会选择市中心区域的几个代表性景点，这样可以减少交通时间，让您在一天内游览更多地方。

让我们选择几个经典景点并获取它们的详细信息：

```
> Ran maps_search_detail
```

我正在为您规划上海一天的旅游攻略，已经获取了上海主要的旅游景点信息。首先，我看到上海豫园是一处非常值得游览的地方，它位于福佑路168号，是国家AAAA级景点，门票40元，开放时间为09:00-16:30（停止入场时间16:00），周二至周日开放。

让我继续查询其他几个主要景点的信息：

```
> Ran maps_search_detail
```

让我继续获取更多景点信息，包括东方明珠和静安寺：

```
> Ran maps_search_detail
```

现在我将继续获取静安寺的信息，以完成我们的旅游攻略规划：

```
> Ran maps_search_detail
```

我已经为您收集了上海几个主要景点的详细信息，现在可以为您制定一个合理的一日游攻略了。以下是我基于高德地图数据整理的上海一天旅游计划：

上海一日游攻略

上午行程：上海豫园 + 城隍庙区域
08:30 - 11:30 上海豫园

地址：福佑路168号

门票：40元

开放时间：09:00-16:30（16:00停止入场）

亮点：豫园是明代古典园林，有九曲桥、湖心亭、玉玲珑等著名景点，建议游览2-3小时

周边可以游览城隍庙商业区，品尝上海特色小吃如南翔小笼、生煎包等

中午：前往陆家嘴区域

11:30 - 12:30 午餐时间

可以在豫园周边享用上海本地特色美食

午餐后乘坐地铁前往陆家嘴区域（可乘坐10号线在豫园站上车，在陆家嘴站下车）

下午行程：东方明珠 + 陆家嘴金融区

13:00 - 16:00 东方明珠广播电视塔

地址：世纪大道1号（距陆家嘴地铁站1号出口步行约220米）

门票：80元

开放时间：09:00-21:00（最晚入场时间20:30）

等级：AAAAA级旅游景区

亮点：259米高的观光层可以360°俯瞰上海全景；还可参观上海历史发展陈列馆

16:00 - 17:30 陆家嘴金融区漫步

可以在陆家嘴金融区漫步，欣赏上海环球金融中心、上海中心大厦、金茂大厦等现代建筑群

推荐前往外滩，隔江欣赏浦东现代化天际线

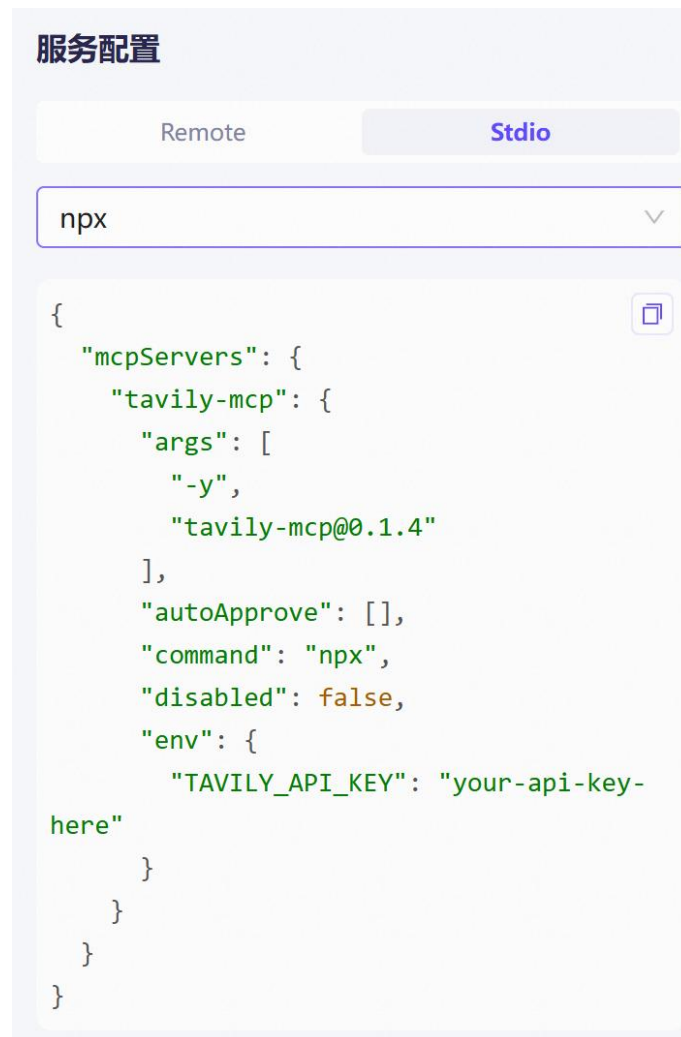
.....

CASE: Tavily 搜索

CASE: Tavily 搜索

Step1, 添加 Tavily MCP

```
{
  "mcpServers": {
    "tavily-mcp": {
      "args": [
        "-y",
        "tavily-mcp@0.1.4"
      ],
      "autoApprove": [],
      "command": "npx",
      "disabled": false,
      "env": {
        "TAVILY_API_KEY": "your-api-key-here"
      }
    }
  }
}
```



<https://modelscope.cn/mcp/servers/@tavily-ai/tavily-mcp>

CASE: Tavily 搜索

Step2, 使用具有MCP功能的客户端进行调用

比如: Cursor具有MCP功能, 可以在Agent模式下输入

用Tavily MCP, 查找 黄金 相关的新闻

✓ Ran tavily-search

RESPONSE

Detailed Results:

Title: 黄金价格走势- 即时黄金,白银,铂金和钯金现货价格- BullionVault.com

URL: <https://zh-hans.bullionvault.com/gold-price-chart.do>

Content: 我们使用 cookies (包括第三方Cookies) 来记录您使用我们网站时的偏好, 并帮助我们了解网站访问者如何使用我们的网站, 以及使我们在第三方网站上展示的任何广告更具相关性, 以便我们改进自身网站的服务。要了解更多信息, 请参阅我们的 隐私政策 以及我们的 cookie 政策。请同意我们使用 cookies, 点击'接受'或选择'选项'来设置您需要的 cookie 种类。目前电话和邮件量较大, 回复时间可能会比平时更长, 感谢您的耐心等待。

2025 年圣诞节 & 2026年新年: 交易继续 24/7。如需客户支持, 请参阅2025年圣诞节和2026年新年假期办公时间 | 铂金、黄金和白银因“美元贬值”交易再度跳涨 2025年12月23日

Show more

已使用 Tavily MCP 搜索到与“黄金”相关的新闻。主要结果如下:

搜索结果摘要

1. 黄金价格走势

- BullionVault.com: 贵金属价格周二再度飙升, 黄金和白银以所有货币计价均创下新高, 黄金价格逼近4500美元关口仅差3美元
- TradingView: 黄金价格在全球金融市场受到广泛关注, 2025年黄金大阳收盘, 2026年还有高点

2. 市场分析

- Trading Economics: 黄金当前价格为4454.39美元, 上涨0.18%
- 英为财经: 黄金价格在欧洲交易时段周五上涨, 这是2026年第一个交易日, 在去年创下历史性涨幅后, 美元走弱和降息预期推动价格上涨

3. 期货交易

- COMEX黄金期货: 收涨0.9%, 报4371.4美元/盎司
- 东方财富: COMEX黄金期货当前价格4415.8, 涨幅1.65%

4. 市场动态

- 2025年黄金涨幅超过60%, 刷新50多次历史新高
- 地缘政治紧张局势、美元走弱、货币政策转向预期等因素推动金价上涨
- 高盛预测2026年目标4900美元

5. 相关机构

- 上海黄金交易所行情走势
- 中国银行黄金市场分析
- 新浪财经纽约黄金期货行情

需要我进一步分析某个具体方面吗?

Qwen-Agent + Tavily

TO DO: 在Qwen-Agent中, 集成tavily

Step1, 在tools中添加 tavily

```
# MCP 工具配置
tools = [{
    "mcpServers": {
        "tavily-mcp": {
            "command": "npx",
            "args": [
                "-y",
                "tavily-mcp@0.1.4"
            ],
            "autoApprove": [],
            "disabled": False,
            "env": {
                "TAVILY_API_KEY": os.getenv('TAVILY_API_KEY',
            }
        }
    }
}]
```

调整聊天界面中的建议提示词

```
chatbot_config = {
    'prompt.suggestions': [
        '查找黄金相关的新闻',
        '搜索最新的AI技术发展趋势',
        '查找2026年经济预测相关文章'
    ]
}
```

Qwen-Agent + Tavily

帮我搜索一下关于AI的最新新闻

帮我搜索一下关于AI的最新新闻



user

▶ Start calling tool "tavily-mcp-tavily-search" ...

▶ Finished tool calling.

以下是关于AI的一些最新新闻：

- 英伟达推出革命性Alpamayo平台**：在CES 2026上，英伟达推出了专为实体机器人和自动驾驶汽车设计的开源AI平台Alpamayo，旨在提升复杂场景下的智能决策能力。[来源](#)
- 亚马逊正式推出Alexa + 专属网站**：亚马逊新推出的Alexa+专属网站允许用户通过浏览器直接对话，显示其与ChatGPT竞争加剧。新版AI助手在语音识别和理解能力上显著提升。[来源](#)
- 黄仁勋宣布机器人迎来“ChatGPT时刻”**：英伟达CEO黄仁勋在CES2026上表示，AI正加速从数字世界迈向物理现实，并开源了多个“物理AI”模型以帮助机器人理解和适应现实世界的物理属性。[来源](#)
- 高通发布Dragonwing IQ10处理器**：面向自主移动机器人和全尺寸人形机器人市场，高通发布的这款处理器整合了边缘计算能力，标志着该公司在布局下一代智能终端方面的关键一步。[来源](#)
- 吉利携最新智能汽车AI技术亮相CES2026**：吉利在此次展会上发布了基于WAM世界行为模型的全域AI 2.0技术体系，实现了AI技术在整车各域的跨域融合。[来源](#)

这些新闻涵盖了自动驾驶、智能家居助手以及智能汽车等多个领域内的最新进展。希望这对你有所帮助！如果需要更多详细信息或有其他问题，请随时告诉我。



搜索助手

Qwen-Agent + Tavily

<https://www.aibase.com/zh/news/24360> 这篇文章的详细内容

▼ Start calling tool "tavily-mcp-tavily-extract" ...

```
{"urls": ["https://www.aibase.com/zh/news/24360"], "extract_depth": "basic"}
```

► Finished tool calling.

这篇文章的标题是《Google Gemini 推出“引导式学习”新功能：AI 私人导师式步步指导，革新个性化教育》。该文章介绍了 Google 最近在 Gemini 平台上推出的新功能——“引导式学习”。这项功能将学习过程转变为一种个性化的、互动式的体验，就像拥有一个专属的私人导师一样。它不仅仅提供答案，而是通过逐步分解复杂的主题、适应用户的学习节奏并验证理解程度来帮助用户深入掌握知识。

“引导式学习”充分利用了 Gemini 的多模态能力，创建了结构化学习路径。用户可以通过 Gemini 网站或应用程序轻松访问此功能。他们只需启动一个新的对话，在工具栏中选择“引导式学习”模式，然后输入问题或上传文档（如笔记、PDF 或教科书）即可开始使用，无需额外设置。激活后，系统将以互动对话的形式展开教学。

这个功能会将主题分解成小块，并首先给出基本解释，同时结合图像、图表和 YouTube 视频等多媒体内容辅助理解。例如，在学习量子纠缠时，Gemini 会从基本概念入手，通过提出引导性问题来评估用户的理解程度（如“你认为接下来会发生什么？”），只有当确认用户已经理解后才会继续下一步。

此外，“引导式学习”还支持自定义测验和闪卡生成，并能够根据用户的反馈动态调整难度——对于初学者来说，会得到简化的解释；而对于高级用户，则会深入探讨实际应用。这种设计基于教育科学原理，如主动回忆和间隔重复，已被证明可以提高长期记忆和实际应用能力。与传统的搜索引擎或静态资源不同，“引导式学习”强调渐进式的对话方式，允许用户随时提问、澄清或探索相关主题。早期用户反馈表明，该功能在编程、语言学习以及科学理论等领域表现优异，能够将抽象的概念转化为直观的体验。

总之，“引导式学习”与普通的 AI 聊天相比，其特点在于其结构化和验证机制。它不是简单地输出事实，而是即时构建“脚手架”式的课程。此外，相比于其他竞争产品如 ChatGPT 的类似模式，Gemini 在多媒体整合和深度聚焦方面更胜一筹，并且支持文档上传分析，非常适合专业人士提升技能或学生备考。

Summary (Tavily 搜索)

tavily-search (实时网络搜索)

- query: 搜索关键词, 字符串, 必填
- max_results: 返回结果条数, 整数, 可选 (默认 3-10)
- search_depth: 搜索深度, 枚举 basic / advanced, 可选 (advanced 会抓取并总结页面正文)
- include_domains / exclude_domains: 限定或排除域名, 数组, 可选
- time_range: 时间过滤, 如 “day” “week” “month”, 可选
- include_answer: 是否要求直接返回答案, 布尔, 可选
- include_raw_content: 是否携带原始 HTML, 布尔, 可选

tavily-extract (网页内容智能提取)

- urls: 要提取的页面地址, 字符串或字符串数组, 必填
- extract_depth: 提取深度, 枚举 basic / advanced, 可选 (advanced 会解析更多细节并过滤噪音)
- format: 返回格式, 枚举 text / markdown / json, 可选 (默认 text)
- include_images: 是否同时提取图片链接, 布尔, 可选

CASE: 桌面TXT统计器
(MCP SDK使用)

MCP SDK功能

Python MCP SDK

```
pip install mcp -i https://pypi.tuna.tsinghua.edu.cn/simple
```

- **创建 MCP Server**: 提供标准化的 API, 供 AI 模型调用外部功能 (如文件操作、数据库查询等)。
- **注册工具 (Tools)**: 通过 `@mcp.tool()` 装饰器, 将 Python 函数暴露给 AI 模型使用。
- **安全交互**: 支持权限控制, 确保 AI 不会越权访问敏感数据。
- **跨平台兼容**: 支持与 OpenAI、Anthropic Claude 等 LLM 集成。

FastMCP功能

FastMCP 是 Python MCP SDK中的轻量级服务器框架：

- **简单易用**： 仅需几行代码即可启动 MCP Server。
- **支持多种传输方式**： 如 stdio（标准输入输出）、HTTP 等。
- **自动工具发现**： 通过 `@mcp.tool()` 注解自动注册函数，供 AI 调用

CASE：桌面TXT统计器（MCP SDK使用）

TO DO：搭建一个桌面TXT文件数量统计器，使用FastMCP

功能1：count_desktop_txt_files，统计桌面上.txt文件的数量

功能2：list_desktop_txt_files，获取桌面上所有.txt文件的列表

```
import os
from pathlib import Path
from mcp.server.fastmcp import FastMCP
# 创建 MCP Server
mcp = FastMCP("桌面 TXT 文件统计器")
@mcp.tool()
def count_desktop_txt_files() -> int:
    """统计桌面上.txt文件的数量"""
    # 获取桌面路径
    desktop_path = Path(os.path.expanduser("~/Desktop"))
    # 统计.txt文件
    txt_files = list(desktop_path.glob("*.txt"))
    return len(txt_files)
```

```
@mcp.tool()
def list_desktop_txt_files() -> str:
    """获取桌面上所有.txt文件的列表"""
    # 获取桌面路径
    desktop_path = Path(os.path.expanduser("~/Desktop"))
    # 获取所有.txt文件
    txt_files = list(desktop_path.glob("*.txt"))
    # 返回文件名列表
    if not txt_files:
        return "桌面上没有找到.txt文件。"
    # 格式化文件名列表
    file_list = "\n".join([f"- {file.name}" for file in txt_files])
    return f"在桌面上找到 {len(txt_files)} 个.txt文件：\n{file_list}"

if __name__ == "__main__":
    # 初始化并运行服务器
    mcp.run()
```

@mcp.tool() 注解告诉 MCP 该函数可被 AI 调用。
mcp.run() 启动服务器，可选择 stdio 或 http 等通信方式。

CASE: 桌面TXT统计器 (MCP SDK使用)

mcp dev txt_counter.py

```
Starting MCP inspector...
Proxy server listening on port 3000
New SSE connection
Query parameters: {
  transportType: 'stdio',
  command: 'uv',
  args: 'run --with mcp mcp run txt_counter.py',
  env: '{}'}
}
Stdio transport: command=D:\AppData\Local\Programs\Python\Python311\Scripts\uv.exe, args=run,--with,mcp,mcp,run,txt_counter.py
```

🔗 MCP Inspector is up and running at <http://localhost:5173> 🚀

打开浏览器访问 <http://localhost:5173/>

MCP Inspector v0.7.0

Transport Type: STDIO

Command: uv

Arguments: run --with mcp mcp run txt_counter

> Environment Variables

Connect

● Disconnected

Connect to an MCP server to start inspecting

History
No history yet

Server Notifications
No notifications yet

System

CASE：桌面TXT统计器（MCP SDK使用）

Step1, 点击 Connect

Step2, 选择Tools => count_desktop_txt_files

Step3, 点击 Run Tool

看到结果为 2

Step4, 选择Tools => list_desktop_txt_files

Step5, 点击 Run Tool

看到结果为

在桌面上找到 2 个 .txt 文件：

- 111.txt

- 新建 文本文档.txt

在history中，可以看到对应的 tools请求情况

History

4. tools/call

Request:

```
{
  "method": "tools/call",
  "params": {
    "name": "list_desktop_txt_files",
    "arguments": {},
    "_meta": {
      "progressToken": 1
    }
  }
}
```

Response:

```
{
  "content": [
    {
      "type": "text",
      "text": "在桌面上找到 2 个 .txt 文件: \n- 111.txt\n- 新建 文本文档.txt"
    }
  ],
  "isError": false
}
```

MCP Inspector

MCP Inspector:

专为MCP Server设计的开源调试工具，目的在于测试和优化 MCP Server的功能。

MCP Inspector的核心功能

- 服务器连接管理

支持本地和远程 MCP 服务器连接，可配置不同的传输方式（如 STDIO 或 HTTP+SSE）。

允许传递环境变量和命令行参数，方便调试不同运行环境下的服务器。

提供执行历史记录（History），方便回溯工具调用过程。

- 服务浏览与调用

提供可视化界面（默认访问 <http://localhost:5173>），展示 MCP 服务器提供的工具（Tools）、资源（Resources）、提示（Prompts）等服务。

开发者可以直接在 UI 中调用工具并查看执行记录，例如测试计算器工具的加法运算并实时观察返回结果。

UV工具

在MCP协议中，uv工具常用于启动和管理MCP服务器。例如，在配置MCP服务器时，可以通过uv命令指定运行的Python脚本。

以下是一个典型的MCP服务器配置示例：

```
{
  "mcpServers": {
    "example-server": {
      "name": "example-server",
      "type": "stdio",
      "command": "uv",
      "args": [
        "--directory",
        "/path/to/your/project",
        "run",
        "main.py"
      ]
    }
  }
}
```


在这个配置中，uv被用来运行main.py脚本，从而启动MCP服务器

打卡：Agent + MCP



可以使用Qwen-Agent，或其他具有Agent能力的客户端，比如Cursor, Cherry Studio，集成MCP服务，并通过自然语言进行调用

- 1) 规划从xx到xx的7天自驾游
- 2) 获取某网页内容，并转化为Markdown格式
- 3) 检索最新的关税新闻
- 4) 统计桌面上的txt的文件数量

The image features a blue background with several groups of 3D white cubes. Each cube has a light gray shadow on its bottom face and a teal shadow on the surface below it. The cubes are arranged in various sizes and orientations, creating a sense of depth and perspective. The text is centered in the middle of the image.

Thank You
Using data to solve problems