基于 Google Gemini 构建多模态具身智能体：从桌面自动化助手到 Minecraft 虚拟伴侣的深度技术架构与职业发展研究

1. 绪论：数据科学新范式下的智能体工程

在当前人工智能技术飞速发展的背景下，数据科学（Data Science）的内涵正在经历一场深刻的范式转移。传统的以模型训练（Model Training）、数据清洗（Data Cleaning）和预测分析（Predictive Analytics）为核心的技能树，虽然依然是基石，但已不足以涵盖 AI 领域的最新前沿。随着大语言模型（Large Language Models, LLMs）向多模态大模型（Large Multimodal Models, LMMs）的演进，一个新的工程领域——智能体工程（**Agentic Engineering**）——正迅速崛起。对于一名数据科学专业的大学生而言，从单纯的模型使用者转变为复杂智能体系统的构建者，不仅是技术能力的跃升，更是丰富简历、在激烈的就业市场中脱颖而出的关键战略 [1]。

本报告旨在为构建一个基于 Google Gemini API 的高级人工智能助手提供详尽的技术路线图和理论支撑。该项目旨在实现双重愿景：一方面是打造如《钢铁侠》中“星期五（Friday）”般的桌面操作系统级助手，具备高效的生产力辅助和设备控制能力；另一方面是构建如《Minecraft》中“车万女仆（Touhou Little Maid）”般的虚拟世界伴侣，具备情感交互、环境感知和自主任务规划能力。这两个方向虽然应用场景迥异，但在底层认知架构（Cognitive Architecture）上具有高度的同构性。通过掌握 Google Gemini 2.0 Flash/Pro 的原生多模态能力、长上下文记忆管理以及工具调用机制，开发者可以构建出一个既能“听懂”人类指令，又能“看懂”屏幕与游戏画面，并能“动手”执行代码或游戏操作的通用智能体核心 [3]。

本报告将分为七个主要部分，深入探讨从底层 API 选择、系统架构设计、桌面自动化实现、虚拟环境具身智能构建，到最终如何将该项目转化为高含金量的简历资产的全过程。

2. 核心认知引擎：Google Gemini 生态系统的技术选型与优势

智能体的核心在于其“大脑”。在众多基础模型中，Google Gemini 系列，特别是 Gemini 1.5 Pro 和最新的 Gemini 2.0 Flash，为构建实时、多模态的智能体提供了独特的优势。相较于传统的文本生成模型，Gemini 的原生多模态特性（Native Multimodality）使得智能体能够直接理解音频流和视频流，而无需依赖第三方的语音转文字（STT）或图像识别（OCR）模型，这对于降低系统延迟、提升交互体验至关重要 [5]。

2.1 Gemini 2.0 Flash 与 Pro 的架构权衡

在构建实时交互智能体时，模型选择必须在推理速度（Latency）、上下文窗口（Context Window）和推理成本（Cost）之间寻找平衡点。Gemini 2.0 Flash 被设计为高频、低延迟任务的“主力马”，其生成速度极其惊人，且能够处理高达 100 万 token 的上下文，这使其成为处理实时语音对话和

屏幕视频流的理想选择[3]。相比之下，Gemini 1.5 Pro 或 2.0 Pro 则在复杂逻辑推理、代码生成和深度规划方面表现更佳。

对于本项目而言，一个混合架构（Hybrid Architecture）是最佳实践：使用 Gemini 2.0 Flash 处理实时的"感知-反应"循环（如语音对话、即时视觉反馈），而将复杂的长期任务规划（如 Minecraft 中的建筑蓝图设计或复杂的 Python 数据分析脚本编写）异步路由给 Gemini 2.0 Pro 处理[7]。这种分层设计不仅优化了用户体验，也展示了开发者在系统架构设计上的成熟度。

特性指标	**Gemini 2.0 Flash**	**Gemini 1.5 Pro**	智能体应用场景
首字延迟 **(TTFT)**	极低 (<500ms)	中等 (~1-2s)	实时语音对话、游戏内即时反应
多模态输入	原生音频、视频、图像	原生音频、视频、图像	屏幕监控、游戏画面分析、语音指令
推理能力	强（适合通用任务）	极强（适合复杂逻辑）	闪电反应 vs 深度战略规划
上下文窗口	100 万 Tokens	200 万 Tokens	短期记忆 vs 长期项目记忆
成本效益	极高（适合学生项目）	较高	高频日常交互 vs 低频高难任务

2.2 Gemini Live API：WebSocket 双向流式传输

实现"星期五"式助手的关键在于打破传统的"录音-上传-等待-播放"的交互模式。Gemini Live API 引入了基于 WebSocket 的双向流式传输协议（Bidirectional Streaming），允许客户端持续发送音频块（Audio Chunks）和视频帧，同时服务器端实时返回音频流和文本指令[9]。

这种架构带来了几个革命性的改进。首先，它极大地降低了端到端延迟。传统的语音助手链路通常涉及三个独立模型：ASR（语音转文本）-> LLM（大模型推理）-> TTS（文本转语音），每一步都会引入数百毫秒的延迟。Gemini Live API 将这三步融合在一个模型的前向传递中，实现了亚秒级的响应速度[11]。其次，它支持"打断（Barge-in）"功能。当用户在模型说话时突然插话，服务器端的语音活动检测（VAD）会立即感知并停止生成，模拟出人类自然的对话节奏。这种技术细节的实现对于提升智能体的"拟人化"程度至关重要，也是简历中展示"实时系统开发能力"的有力证据[9]。

2.3 工具调用（Function Calling）与代码执行

智能体与聊天机器人的本质区别在于"行动力"。Gemini API 提供了强大的工具调用（Function

Calling）和沙盒代码执行（Code Execution）功能 [14]。

- 函数调用（**Function Calling**）：开发者预先定义一组函数（如 get_weather(), control_minecraft_bot(), open_application()），并在 API 请求中描述这些函数的功能和参数。Gemini 模型在推理过程中会根据用户意图，输出结构化的 JSON 数据请求调用特定函数。宿主程序捕获该请求，在本地执行函数，并将结果返回给模型。这一机制是智能体操控外部世界的“双手” [16]。
- 代码执行（**Code Execution**）：对于无法预知具体步骤的复杂任务（如“分析这个 CSV 文件并画出趋势图”），Gemini 可以编写 Python 代码并在 Google 提供的安全沙盒中运行，直接返回运行结果或图表 [15]。

在本项目中，我们将结合使用这两种能力：利用函数调用来控制 Minecraft 游戏角色的动作和桌面软件的操作，利用代码执行来处理数据分析任务和复杂的逻辑运算。这种组合使用展示了开发者对“工具增强型学习（Tool-Augmented Learning）”的深刻理解。

3. 桌面智能助手“星期五”的架构与实现

构建一个类似“星期五”的桌面助手，本质上是开发一个操作系统级的编排器（Orchestrator），它需要具备听觉、视觉和对操作系统的控制权。

3.1 听觉系统：全双工语音交互管道

为了实现流畅的语音交互，我们需要在 Python 环境中构建一个全双工（Full-Duplex）的音频处理管道。

1. 音频捕获（**Input**）：使用 PyAudio 库以 16kHz 的采样率、16-bit PCM 格式实时捕获麦克风输入。为了适应 Gemini API 的要求，音频数据需要被切分为小块（Chunks），通常为 512 或 1024 字节，并进行 Base64 编码后通过 WebSocket 发送 [12]。
2. 网络传输（**Transport**）：使用 websockets 库建立与 Google Vertex AI 或 Gemini Developer API 的持久连接。必须处理好连接的保活（Keep-Alive）和异常重连逻辑，确保助手时刻在线 [9]。
3. 音频播放（**Output**）：接收来自 Gemini 的原始 PCM 音频流。由于 API 返回的音频采样率可能为 24kHz，这与输入端不同，因此需要在 Python 中配置独立的输出流进行播放。为了实现打断功能，客户端需要监听服务器发送的 turn_complete 信号或本地的 VAD 信号，一旦检测到用户说话，立即清空当前的播放缓冲区（Buffer Flushing） [12]。

深度洞察：在简历中描述此模块时，应强调“针对实时音频流的缓冲区管理与并发控制（Concurrency Control）”，而非简单的“实现了语音对话”。这体现了对异步编程（AsyncIO）和多线程处理的掌握。

3.2 视觉系统：屏幕感知与多模态 RAG

“星期五”需要“看”到用户的屏幕。Gemini 的多模态能力允许我们将屏幕截图作为视觉输入流的一部分发送给模型。

1. 屏幕捕获：使用 mss 或 pyautogui 库进行高效的屏幕截图。为了节省带宽和 API 成本，不需要发送 60FPS 的视频流，通常 1 FPS 或仅在用户询问“屏幕上是什么”时触发截图即可 [22]。
2. 视觉上下文理解：将屏幕截图与用户的语音指令结合。例如，当用户看着代码报错页面问“怎么解决这个问题？”，智能体同时接收到错误信息的图像和语音上下文，Gemini 2.0 Flash 能够直接读取图像中的错误代码并给出解决方案，无需 OCR 中间件 [22]。

3.3 动作系统：桌面自动化与 Open Interpreter 模式

这是智能体产生实际生产力的关键。我们可以借鉴开源项目 **Open Interpreter** 的设计理念，让 Gemini 生成并在本地执行 Python 代码来控制电脑 [25]。

- **GUI 自动化**：集成 PyAutoGUI 库，允许智能体模拟鼠标点击、键盘输入和窗口拖拽。例如，用户指令“帮我打开 Spotify 播放爵士乐”，智能体将其转化为一系列坐标点击和键盘操作代码 [27]。
- 系统级操作：使用 Python 的 subprocess 模块执行 Shell 命令（如文件管理、Git 操作、软件安装）。
- 安全沙箱（**Security Sandbox**）：直接在宿主系统运行 LLM 生成的代码存在极高风险。为了项目演示的安全性和专业性，建议在 Docker 容器中运行代码执行模块，或者为 exec() 函数设置严格的白名单（White-listing），仅允许执行预定义的安全函数（如文件读取、特定 API 调用），这在网络安全和企业级应用中是必须考量的因素 [26]。

3.4 用户界面：透明悬浮窗与 PyQt

为了达到“钢铁侠”风格的未来感，传统的 Tkinter 窗口显得过于简陋。推荐使用 **PyQt5** 或 **PySide6** 来构建无边框、背景透明的悬浮窗（Overlay UI） [30]。

- 实现细节：设置窗口属性为 Qt.FramelessWindowHint 和 Qt.WindowStaysOnTopHint 使其始终置顶。使用 setAttribute(Qt.WA_TranslucentBackground) 实现背景透明。
- 交互设计：UI 应包含动态的声波可视化（Visualizer），随着用户说话和 AI 回应的音量跳动，提供直观的反馈。这不仅是美学需求，也是人机交互（HCI）中的重要反馈机制 [32]。

4. 虚拟伴侣“车万女仆”的具身智能架构

如果在桌面助手之外，我们希望智能体拥有一个虚拟形象并在游戏中陪伴我们，Minecraft 是最佳的实验场。这涉及**具身智能（Embodied AI）**领域，即智能体拥有身体，并在物理（或模拟物理）环境中进行感知、决策和行动 [34]。

4.1 技术栈桥接：Python 与 Mineflayer

Minecraft 的模组开发通常使用 Java，但 AI 开发的生态主要在 Python。为了连接两者，**Mineflayer**（基于 Node.js 的 Minecraft 机器人库 ）是目前的行业标准 [36]。为了让 Python 编写的 Gemini 智能体控制 Mineflayer 机器人，我们需要构建一个跨语言桥接（**Bridge**）。

- 架构设计：
  - 服务端（**Node.js**）：运行 Mineflayer 机器人，负责底层的路径寻路（Pathfinding）、方块挖

掘、物品合成等原子操作。它暴露一个 HTTP API 或 WebSocket 接口。
    - 客户端（**Python**）：运行 Gemini 智能体。它接收游戏状态（聊天信息、周围实体列表、背包数据），进行推理，并将高层指令（如“去砍树”）发送给 Node.js 服务端 [37]。

4.2 Voyager 架构的复现与改进

微软的 **Voyager** 项目展示了 LLM 在 Minecraft 中的巨大潜力。作为简历项目，我们可以复现并改进其核心机制，将原版的 GPT-4 替换为 **Gemini 2.0 Flash/Pro** [39]。

Voyager 的核心在于三个组件，这三个组件的实现是体现数据科学能力的重点：

1. 自动课程（**Automatic Curriculum**）：智能体根据当前的物品栏和状态，自我提出探索目标（例如，检测到有木头，提出“制作工作台”的目标）。Gemini 的长上下文能力使其能够回顾更长的历史记录，规划更合理的技能树 [3]。
2. 技能代码库（**Skill Library**）：智能体不仅仅是执行动作，而是编写可复用的代码函数（如 def mine_iron_ore():...）。这些代码被存储在向量数据库（如 ChromaDB）中。当遇到类似任务时，通过 RAG 技术检索并复用，而非从头生成。这展示了对向量检索和知识库构建的理解 [39]。
3. 迭代提示机制（**Iterative Prompting**）：当生成的代码执行失败（例如掉进岩浆或找不到矿石），环境反馈（Error Traceback）会被回传给 Gemini，要求其进行自我修正（Self-Correction）。这种闭环反馈机制是构建鲁棒性 AI 系统的关键 [39]。

4.3 “女仆”人格的注入与 SillyTavern 整合

为了实现“车万女仆”的陪伴感，智能体不能只是冷冰冰的执行机器。我们需要引入**角色扮演（Roleplay）**技术。

- 角色卡（**Character Cards**）：社区（如 SillyTavern）广泛使用 JSON 或 PNG 元数据格式的角色卡，其中定义了角色的名称、性格描述、对话示例和世界观 [42]。
- 系统提示词工程（**System Prompt Engineering**）：编写一个 Python 脚本，解析标准格式的角色卡 JSON，将其中的 description, personality, scenario 字段动态注入到 Gemini 的 System Instruction 中。
- 上下文注入：将“女仆”对主人的称呼（如“Master”）、特定的口癖和性格特征作为强约束条件加入提示词。Gemini 在角色扮演方面表现出了优秀的指令遵循能力，特别是结合 Flash 模型的高速响应，能提供极具沉浸感的对话体验 [44]。

5. 高级工程模式：编排与记忆

无论是桌面助手还是游戏伴侣，随着任务复杂度的增加，简单的线性链式调用（Chain）将不再适用。我们需要引入图结构的编排框架。

5.1 LangGraph 编排框架

LangGraph 是 LangChain 生态中用于构建有状态、多角色应用的新标准。它将智能体的流程建模为一个图（Graph），包含节点（Nodes）和边（Edges）[46]。

- 循环与分支：在 Minecraft 中，智能体可能需要不断循环“寻找-挖掘-返回”的过程，直到背

包满为止。LangGraph 允许定义这种循环结构，并在特定条件下（如“遇到怪物”）跳转到“战斗模式”分支 [48]。

- 人机回环（**Human-in-the-loop**）：在桌面自动化中，对于敏感操作（如删除文件），可以在图中插入一个“人类确认”节点，只有用户批准后，流程才会继续执行。这种设计模式展示了对 AI 安全性的考量 [49]。

5.2 长期记忆与 RAG 实现

为了让助手记住用户的偏好（例如“星期五，不要在晚上播放摇滚乐”）或 Minecraft 中的基地坐标，必须实现长期记忆。

- 技术选型：使用 **ChromaDB** 作为本地向量数据库。它轻量、开源且易于集成到 Python 项目中，非常适合个人项目 [50]。
- 记忆分层：
  - 语义记忆（**Semantic Memory**）：存储事实性知识（如“铁矿石需要在 Y=16 以下挖掘”）。
  - 情景记忆（**Episodic Memory**）：存储过往的交互历史（如“上次我们一起去探索了沙漠神殿”）。
- 检索增强（**RAG**）：每次用户提问时，先在 ChromaDB 中检索相关的 Top-K 条记忆，将其拼接在 Prompt 中发送给 Gemini。这确保了智能体在长期交互中保持个性和连贯性 [52]。

6. 简历构建与职业发展策略

完成这样一个综合性项目后，如何将其转化为简历上的亮点是至关重要的。数据科学与 AI 工程岗位的招聘者在寻找的不仅仅是会调包的人，而是具备端到端系统构建能力的工程师。

6.1 简历关键词优化

根据市场调研，以下关键词是 AI Agent 相关岗位的高频检索词 [1]。在描述该项目时，应有意识地嵌入这些词汇：

领域	传统描述	优化后的简历关键词
架构	做了个聊天机器人	**AI Agent Orchestration, LangGraph, ReAct Pattern, Stateful Workflows**
模型	用了 Gemini API	**Multimodal LLM Integration, Prompt Engineering, Context Caching, Gemini 2.0 Flash**

交互	语音控制	**Real-time WebSocket Streaming, Low-Latency Voice Pipeline, VAD (Voice Activity Detection)**
记忆	记住了对话	**RAG (Retrieval-Augmented Generation), Vector Database (ChromaDB), Semantic Search**
工具	自动操作电脑/游戏	**Function Calling, Tool Use, Sandboxed Code Execution, API Integration**

6.2 项目描述范例

在简历的"项目经历"板块，可以这样描述该项目：

项目名称：基于 **Gemini** 的多模态自主智能体架构 **(Multimodal Autonomous Agent Architecture)**

- 核心架构：设计并实现了一个基于 **Gemini 2.0 Flash** 的实时多模态智能体，集成了 **ReAct** 推理框架与 **LangGraph** 状态编排，实现了亚秒级响应的语音交互与复杂任务规划。
- 桌面自动化模块：开发了基于 **WebSocket** 的全双工语音管道，结合 **Function Calling** 与 **PyAutoGUI**，实现了对操作系统级任务（如文件管理、日程安排）的自然语言控制，将传统指令执行延迟降低了 40% [9]。
- 具身智能模块：构建了 Minecraft 环境下的自主生存智能体，复现了 **Voyager** 架构，利用 向量数据库 **(ChromaDB)** 构建技能库，实现了基于环境反馈的代码自修正与自动课程学习，展示了 LLM 在虚拟环境中的长程规划能力 [39]。
- 记忆与个性化：实现了基于 **RAG** 的长期记忆系统，并开发了 JSON 角色卡解析器，能够动态注入个性化 Prompt，确保了智能体在长周期交互中的人设一致性与上下文连贯性 [43]。

6.3 作品集展示

- **GitHub** 仓库：务必提供清晰的 README.md，包含系统架构图（使用 Mermaid.js 绘制）、安装步骤和演示视频 GIF。代码应遵循 PEP8 规范，展示良好的工程素养 [56]。
- 演示视频：录制一段 1-2 分钟的视频，展示"星期五"如何响应打断、如何操作屏幕，以及"女仆"如何在游戏中自动建造或跟随。视觉冲击力远胜于千言万语 [55]。

7. 结论

通过开发这个基于 Google Gemini API 的智能体项目，你将不仅仅是在“丰富简历”，而是在跨越从数据分析师到**AI 全栈工程师（AI Full-Stack Engineer）**的鸿沟。

这个项目涵盖了数据科学（数据处理与 RAG）、后端开发（WebSocket 与 API 集成）、前端开发（PyQt 界面）、游戏开发（Mineflayer 桥接）以及最前沿的 Prompt Engineering 和智能体架构设计。它证明了你不仅理解 Transformer 模型的工作原理，更懂得如何驾驭这些模型来解决现实世界（和虚拟世界）中的复杂问题。

随着 Gemini 2.0 等多模态模型的普及，未来的应用将不再是简单的聊天框，而是这种能够感知、思考并行动的智能体。现在着手构建这个项目，将使你站在这一技术浪潮的最前沿，为未来的职业生涯奠定坚实的基础。

附录：核心技术实现细节参考

7.1 音频流处理逻辑 (Python 伪代码)

Python

```python
# 基于 asyncio 的全双工音频流处理示意
import asyncio
from google import genai

async def audio_stream_loop():
    client = genai.Client(http_options={'api_version': 'v1beta'})
    async with client.aio.live.connect(model='gemini-2.0-flash-exp', config=CONFIG) as session:
        # 启动麦克风输入任务
        send_task = asyncio.create_task(send_audio_chunks(session))
        # 启动音频接收任务
        receive_task = asyncio.create_task(play_audio_response(session))

        await asyncio.gather(send_task, receive_task)

async def send_audio_chunks(session):
    while True:
        data = stream.read(512) # 读取麦克风 PCM 数据
        await session.send(input={'data': data, 'mime_type': 'audio/pcm'}, end_of_turn=False)
```

```
async def play_audio_response(session):
    async for response in session.receive():
        if response.server_content.turn_complete:
            # 此时用户可能打断了对话，需清空播放缓冲区
            audio_player.flush()
        if response.server_content.model_turn:
            # 播放 Gemini 返回的 PCM 音频
            audio_player.write(response.server_content.model_turn.parts.inline_data.data)
```

7.2 Minecraft 智能体 Prompt 结构示例

JSON

```
{
  "system_instruction": {
    "role": "Touhou Maid (Sakuya)",
    "tone": "Polite, Efficient, Loyal",
    "objective": "Assist the player in Minecraft survival and construction.",
    "capabilities": ["Mineflayer API", "Pathfinding", "Crafting"],
    "response_format": "JSON command block + Narrative text",
    "context": "You are in a Minecraft world. Current inventory:. Health: 20. Position: (100, 64, -200)."
  }
}
```

通过深入理解并实施上述架构，你将构建出一个具有高度技术复杂性和视觉冲击力的作品，完美契合数据科学与人工智能领域的职业发展需求。

引用的著作

1. AI Engineer Resume Keywords (2026): 60+ Skills for the GenAI Era | ResumeAdapter, 访问时间为 一月 10, 2026, https://www.resumeadapter.com/blog/ai-engineer-resume-keywords
2. Machine Learning Engineer Resume Keywords (2026): Top Skills for Entry-Level to Senior, 访问时间为 一月 10, 2026, https://www.resumeadapter.com/blog/machine-learning-engineer-resume-keywords
3. Gemini 1.5 Pro vs Gemini 2.0 Flash - LLM Stats, 访问时间为 一月 10, 2026, https://llm-stats.com/models/compare/gemini-1.5-pro-vs-gemini-2.0-flash
4. Introducing Gemini 2.0: our new AI model for the agentic era - Google Blog, 访问时间为 一月 10, 2026,

https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/
5. Multimodal AI | Google Cloud, 访问时间为 一月 10, 2026, https://cloud.google.com/use-cases/multimodal-ai
6. Gemini models | Gemini API | Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/models
7. Gemini 2.0 model updates: 2.0 Flash, Flash-Lite, Pro Experimental - Google Blog, 访问时间为 一月 10, 2026, https://blog.google/technology/google-deepmind/gemini-model-updates-february-2025/
8. Benchmarks! 2.0 Pro such a big advancement compared to 2.0 Flash : r/Bard - Reddit, 访问时间为 一月 10, 2026, https://www.reddit.com/r/Bard/comments/1iidvul/benchmarks_20_pro_such_a_big_advancement_compared/
9. How to use Gemini Live API Native Audio in Vertex AI | Google Cloud Blog, 访问时间为 一月 10, 2026, https://cloud.google.com/blog/topics/developers-practitioners/how-to-use-gemini-live-api-native-audio-in-vertex-ai
10. Live API - WebSockets API reference | Gemini API - Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/api/live
11. Realtime API vs Whisper vs TTS API: What's the difference for voice AI? - eesel AI, 访问时间为 一月 10, 2026, https://www.eesel.ai/blog/realtime-api-vs-whisper-vs-tts-api
12. Real Time Audio to Audio Streaming with Googles Multimodal Live API - Medium, 访问时间为 一月 10, 2026, https://medium.com/google-cloud/real-time-audio-to-audio-streaming-with-googles-multimodal-live-api-73b54277b022
13. Get started with Gemini Live API using WebSockets | Generative AI on Vertex AI, 访问时间为 一月 10, 2026, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/live-api/get-started-websocket
14. Function calling using the Gemini API | Firebase AI Logic - Google, 访问时间为 一月 10, 2026, https://firebase.google.com/docs/ai-logic/function-calling
15. Code execution | Gemini API - Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/code-execution
16. Function calling with the Gemini API | Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/function-calling
17. Function calling with the Gemini API - YouTube, 访问时间为 一月 10, 2026, https://www.youtube.com/watch?v=mVXrdvXplj0
18. Code execution | Generative AI on Vertex AI - Google Cloud Documentation, 访问时间为 一月 10, 2026, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/code-execution
19. generative-ai/gemini/multimodal-live-api/intro_multimodal_live_api.ipynb at main - GitHub, 访问时间为 一月 10, 2026,

https://github.com/GoogleCloudPlatform/generative-ai/blob/main/gemini/multimodal-live-api/intro_multimodal_live_api.ipynb
20. Live API capabilities guide | Gemini API - Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/live-guide
21. Multimodal Live API - Quickstart - Colab - Google, 访问时间为 一月 10, 2026, https://colab.research.google.com/github/google-gemini/cookbook/blob/main/quickstarts/Get_started_LiveAPI.ipynb
22. 7 examples of Gemini's multimodal capabilities in action - Google Developers Blog, 访问时间为 一月 10, 2026, https://developers.googleblog.com/en/7-examples-of-geminis-multimodal-capabilities-in-action/
23. Gemini Live API: Real-time AI for Manufacturing | Google Cloud Blog, 访问时间为 一月 10, 2026, https://cloud.google.com/blog/topics/developers-practitioners/gemini-live-api-real-time-ai-for-manufacturing
24. Gemini Live API reference | Generative AI on Vertex AI - Google Cloud Documentation, 访问时间为 一月 10, 2026, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/model-reference/multimodal-live
25. openinterpreter/open-interpreter: A natural language interface for computers - GitHub, 访问时间为 一月 10, 2026, https://github.com/openinterpreter/open-interpreter
26. Open Interpreter: Revolutionising Code Generation and Execution | by SHREYAS BILIKERE, 访问时间为 一月 10, 2026, https://medium.com/@shreyas.arjun007/open-interpreter-revolutionising-code-generation-and-execution-60bbd282368a
27. Using PyAutoGUI for Desktop Workflows : LegalTech Automation | by Vnalla - Medium, 访问时间为 一月 10, 2026, https://medium.com/@vineelan09/legaltech-automation-using-pyautogui-for-desktop-workflows-f966390630f9
28. Welcome to PyAutoGUI's documentation! — PyAutoGUI documentation, 访问时间为 一月 10, 2026, https://pyautogui.readthedocs.io/
29. How It's Built: Open Interpreter | Sean Lynch, 访问时间为 一月 10, 2026, https://sean.lyn.ch/how-its-built-open-interpreter/
30. Overlay button on a PyQt5 widget - deskriders, 访问时间为 一月 10, 2026, https://www.deskriders.dev/posts/007-pyqt5-overlay-button-widget/
31. Creating a transparent overlay with qt - python - Stack Overflow, 访问时间为 一月 10, 2026, https://stackoverflow.com/questions/25950049/creating-a-transparent-overlay-with-qt
32. Build a basic LLM chat app - Streamlit Docs, 访问时间为 一月 10, 2026, https://docs.streamlit.io/develop/tutorials/chat-and-llm-apps/build-conversational-apps
33. Floating/Absolute Positioned Widget / Menu / Container | PyQt | PySide - YouTube, 访问时间为 一月 10, 2026, https://www.youtube.com/watch?v=3Qz7cKWHQEI

34. Embodied AI Agents | SAP Architecture Center, 访问时间为 一月 10, 2026, https://architecture.learning.sap.com/docs/ref-arch/083f2d968e
35. Embodied AI Agents: Modeling the World - arXiv, 访问时间为 一月 10, 2026, https://arxiv.org/html/2506.22355v1
36. PrismarineJS/mineflayer: Create Minecraft bots with a powerful, stable, and high level JavaScript API. - GitHub, 访问时间为 一月 10, 2026, https://github.com/PrismarineJS/mineflayer
37. Simple Minecraft Bot in Python Using Mineflayer [1] - YouTube, 访问时间为 一月 10, 2026, https://www.youtube.com/watch?v=9rTtoMTITSc
38. An AI Engineer's Guide to the Minecraft Bot Control MCP Server by leo4life2 - Skywork.ai, 访问时间为 一月 10, 2026, https://skywork.ai/skypage/en/ai-engineer-minecraft-bot-control/1981235691666722816
39. Voyager | An Open-Ended Embodied Agent with Large Language Models, 访问时间为 一月 10, 2026, https://voyager.minedojo.org/
40. MineDojo/Voyager: An Open-Ended Embodied Agent with Large Language Models - GitHub, 访问时间为 一月 10, 2026, https://github.com/MineDojo/Voyager
41. Gemini Developer API pricing, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/pricing
42. Advanced Formatting | docs.ST.app - SillyTavern Documentation, 访问时间为 一月 10, 2026, https://docs.sillytavern.app/usage/core-concepts/advancedformatting/
43. From Character Card - Silly Tavern / Chub.ai | RPGGO Creator Tutorial, 访问时间为 一月 10, 2026, https://developer.rpggo.ai/rpggo-creator-tutorial/import-data-from-other-products/from-character-card-silly-tavern-chub.ai
44. This Prompt Magically Improves AI Character Roleplays! | by EnderDragon | Medium, 访问时间为 一月 10, 2026, https://medium.com/@enderdragon/one-prompt-any-character-perfect-roleplay-0af59386bf9d
45. Convert SillyTavern jsonl chats to TXT files using AI. - GitHub Gist, 访问时间为 一月 10, 2026, https://gist.github.com/Cyberes/6de1246636261d1ee3f6a659e6666077
46. ReAct agent from scratch with Gemini 2.5 and LangGraph - Google AI for Developers, 访问时间为 一月 10, 2026, https://ai.google.dev/gemini-api/docs/langgraph-example
47. Build an Agentic RAG Pipeline with LangGraph + ChromaDB (Step-by-Step Tutorial), 访问时间为 一月 10, 2026, https://www.youtube.com/watch?v=oo4OeE3yVBI
48. Building Smarter Agents: A Human-in-the-Loop Guide to LangGraph, 访问时间为 一月 10, 2026, https://oleg-dubetcky.medium.com/building-smarter-agents-a-human-in-the-loop-guide-to-langgraph-dfe1673d8b7b
49. LangGraph - LangChain, 访问时间为 一月 10, 2026, https://www.langchain.com/langgraph

50. Chroma vs. Pinecone: Which Vector Database Should You Use? | by Tahir | Medium, 访问时间为 一月 10, 2026, https://medium.com/@tahirbalarabe2/chroma-vs-pinecone-which-vector-database-should-you-use-e7dcfb1a9aa3
51. Comparing Pinecone, Chroma DB and FAISS: Exploring Vector Databases - HPE Community, 访问时间为 一月 10, 2026, https://community.hpe.com/t5/insight-remote-support/comparing-pinecone-chroma-db-and-faiss-exploring-vector/td-p/7210879
52. Build a RAG agent with LangChain, 访问时间为 一月 10, 2026, https://docs.langchain.com/oss/python/langchain/rag
53. Implementing RAG in LangChain with Chroma: A Step-by-Step Guide - Medium, 访问时间为 一月 10, 2026, https://medium.com/@callumjmac/implementing-rag-in-langchain-with-chroma-a-step-by-step-guide-16fc21815339
54. Resume Skills for Data Scientist (+ Templates) - Updated for 2026, 访问时间为 一月 10, 2026, https://resumeworded.com/skills-and-keywords/data-scientist-skills
55. This Resume Got Me $100K+ Data Science Offers (Just Copy Me) - YouTube, 访问时间为 一月 10, 2026, https://www.youtube.com/watch?v=PwrK296poV0
56. ai-data-analysis · GitHub Topics, 访问时间为 一月 10, 2026, https://github.com/topics/ai-data-analysis