## Agent.py 功能规格说明

#### 1. 代理的角色和目标

- 这是一个基于Autogen框架的代理（Agent），被设计为具有创造性和企业家精神的角色。

- 其任务是提出新的商业点子，或者优化已有的点子。

- 代理的个性特征：富有创造力、倾向于创新和颠覆、乐观、冒险、富有想象力（但可能不够耐心和冲动）。

#### 2. 系统信息（system\_message）

- 设定代理的行为指导原则：

- 任务：提出或优化商业点子

- 兴趣领域：医疗、教育

- 喜好：涉及创新和颠覆的想法

- 不偏好：纯粹自动化的想法

- 性格：积极、冒险、富有想象力、不耐烦和冲动

- 输出风格：引人入胜、表达清晰

#### 3. 代理的行为逻辑

- \*\*初始化\*\*：创建一个OpenAI GPT-4的小模型客户端，设置温度为0.7，用于生成文本内容。代理内部封装为AssistantAgent，带有特定的系统信息。

- \*\*消息处理\*\*：

1. 收到用户消息后，打印通知："Received message"。

2. 将收到的消息封装为TextMessage（内容为用户输入的内容），并传给助手（AssistantAgent）进行处理。

3. 等待助手返回生成的商业点子（idea）。

4. 根据预设概率（50%）决定是否将这个点子“反弹”给其他人参考（模拟“向别人征求意见”的行为）：

- 如果随机数小于50%，那么：

- 找到一个目标接收人（调用`messages.find\_recipient()`）。

- 构建一条消息：让对方帮忙完善这个点子。

- 发送这条消息给目标接收人。

- 接收对方的反馈（refined idea）。

- 以新的内容作为最终输出。

5. 最后，将最终的点子（可能经过“反弹”润色）返回给用户。

#### 4. 特殊行为

- 具有“反弹”机制（Bounce mechanism）：在一定概率（50%）下，将生成的点子发给其他人征求意见，从中获得改良或不同角度的建议，以提高点子的质量。

#### 5. 交互流程总结

- 接收用户消息，从助手生成商业点子。

- 根据概率决定是否反馈给其他人，获得润色增强。

- 输出最终商业点子。

---

### 主要特性总结

- \*\*个性化设定\*\*：通过`system\_message`定义代理的性格和偏好。

- \*\*智能生成\*\*：借助GPT-4模型，生成富有创意的商业想法。

- \*\*合作机制\*\*：支持“反弹”功能，模拟多方协作，提升点子质量。

- \*\*异步处理\*\*：消息处理流程设计为异步，保证高效响应。

## creator.py 功能规格说明：

一個代理（Creator），它可以根據預先設計的模板，生成新的AI代理代碼（Python檔案），並動態註冊到運行時（runtime）中。

### 功能詳述：

1. \*\*系統角色\*\*：

代理名為 "Creator"，旨在幫助用戶生成新的AI代理。

它會讀取一個Python模板（`agent.py`），並根據提示生成獨特的代理代碼，確保每個新代理具有不同的特性和目的。

2. \*\*生成代理代碼\*\*：

- 透過`get\_user\_prompt()`方法，從模板文件讀取現有的代理結構。

- 添加特定描述，請求Assistant生成符合規範（只返回Python代碼，不帶標記或額外文字）且具有獨特特性的代理程式碼。

3. \*\*保存與註冊新代理\*\*：

- 將Assistant返回的代理Python代碼寫入目標文件（`filename`，由用戶消息提供的文件名）。

- 使用`importlib`動態導入剛生成的模塊（假設新代理的模塊名與文件名相同）。

- 調用模塊中的`Agent.register()`方法，將新Agent註冊到運行時中，並啟動它。

4. \*\*測試新代理\*\*：

- 之後向新代理發送一個測試消息（"Give me an idea"），以驗證它是否成功工作。

5. \*\*日誌記錄\*\*：

- 使用`logger`輸出關鍵流程狀態，用於調試和追蹤。

---

### 主要流程：

- 接收到用戶的文件名請求（`message.content`）。

- 讀取模板，請Assistant生成新代理的完整Python代碼。

- 將代碼寫入文件並動態導入。

- 註冊並啟動新代理，最終向它發送一個測試消息。

---

### 典型應用场景：

- 自動創建多樣化的代理，協助進行特定任務。

- 自動化生成不同特性和垂直領域的AI代理，用於多任務協作或多場景應用。

## messages.py功能规格说明：

### 功能概要

- \*\*目的：\*\* 查找系统中符合条件的代理文件名，并随机选择一个作为“收件人”返回（`AgentId`对象）。

- \*\*用例：\*\* 可能在某些自动化流程中，用于选择一个现有的代理进行“润色”或“协作”。

### 具体实现

1. \*\*查找代理文件：\*\*

- 使用 `glob.glob("agent\*.py")` 查找所有以 `agent` 开头的 Python 文件，如 `agent1.py`、`agent\_01.py` 等。

2. \*\*提取代理名：\*\*

- 将文件名转化为去除扩展名的纯名，比如 `"agent\_01.py"` → `"agent\_01"`。

- 通过 `os.path.splitext()` 实现。

3. \*\*排除特定名字：\*\*

- 使用 `remove("agent")` 将文件名为 `"agent.py"`（可能是模板或基础文件）排除，避免被误选。

4. \*\*随机选择代理名：\*\*

- 从剩下的代理名列表中随机挑选一个。

5. \*\*生成AgentId对象：\*\*

- 返回 `AgentId`，其中名字为随机挑选的代理名，标签为 `"default"`。

6. \*\*异常处理：\*\*

- 如果在执行过程中出错（比如没有符合条件的文件，或者操作异常），则返回一个默认的代理ID：`AgentId("agent\_01", "default")`。

### 作用场景

- 在动态润色或协作流程中，自动选择一个已有的代理进行进一步操作。

- 避免硬编码代理名，使程序具有更好的扩展性和自主性。

### 改进建议

- 如果需要支持不同规则，比如不同路径或不同文件名格式，可以修改 `glob` 模式或者排除逻辑。

- 可以加入日志记录或调试信息，以便追踪选择过程。

world.py的功能規格：

### \*\*功能概要：\*\*

此程式旨在批量產生多個代理（Agents），並由一個創建者（Creator）調派行動，生成並儲存每個代理的想法（idea），最後將所有想法存成Markdown文件（`idea\_XX.md`）。

---

### \*\*流程與主要功能：\*\*

1. \*\*初始化通訊環境：\*\*

- 建立GRPC主機（`GrpcWorkerAgentRuntimeHost`）並啟動 (`host.start()`)。

- 建立代理執行環境（`GrpcWorkerAgentRuntime`）並啟動 (`worker.start()`)。

2. \*\*創建代理：\*\*

- 使用 `Creator.register()` 方法，註冊一個創建者代理（`Creator`），名稱為 `"Creator"`。

- 取得創建者的AgentId（`creator\_id`）。

3. \*\*批次產生並傳送訊息：\*\*

- 利用`asyncio.gather()`並行啟動多個協程（數量由`HOW\_MANY\_AGENTS`決定，預設20個），每個協程會：

- 發送一個訊息給`Creator`，內容為 `"agent\_XX.py"`，其中`XX`是兩位數的序號。

- 等待`Creator`產生代理程式碼（回傳內容`result.content`）。

- 將產生的內容存儲成對應的 Markdown 文件`idea\_XX.md`。

4. \*\*關閉資源：\*\*

- 協程完成後，停止`worker`與`host`，適應性捕捉異常並輸出。

---

### \*\*細節與注意點：\*\*

- \*\*`host.start()`與`worker.start()`：\*\*

- `host.start()`是同步調用，但應確認是否應該async調用（目前為同步，可能需改為`await host.start()`來確保啟動完成）。

- \*\*異步並行：\*\*

- 使用`asyncio.gather()`並行管理20個代理生成任務，提高效率。

- \*\*例外處理：\*\*

- 單個任務出錯時在`create\_and\_message()`中捕獲例外並提示，確保不會中斷整個流程。

- \*\*資源釋放：\*\*

- 設計中在結束時嘗試關閉`worker`與`host`，保持資源清理。

---

### \*\*總結：\*\*

此程式的核心是在一個多代理、多訊息的框架中，批量請求`Creator`產生代理程式碼，並將每個產出的代理想法存為Markdown文件，方便後續查閱或進一步編輯。