异构工具联动系统 – 工具模块解释性报告

一、整体说明

本次提交包含四个 Python 工具模块，分别对应：**数值计算、文档生成、新闻检索、股票行情查询。**
四个模块遵循统一的工具规范输出格式：

```
{
  "success": bool,          # 是否成功
  "data": Any | None,       # 成功时的结果数据
  "error": str | None,      # 失败时的错误信息
  "metadata": {             # 工具元信息
      "tool_name": str,
      "duration": "xxms",
      "timestamp": "YYYY-MM-DD HH:MM:SS",
      "is_mock": bool,      # 是否使用 Mock/降级数据
      ...
  }
}
```

这种设计保证了工具可被上层 Agent/调度器一致调用，并且便于在联动过程中进行日志记录、错误恢复与结果汇总。

二、data.py —— 数值计算工具

2.1 功能定位

`data.py` 实现一个轻量的**数值表达式计算工具**，用于处理用户或 Agent 提供的算术表达式，例如：

- `"1 + 2 * 3"`
- `" (10 - 4) / 2 "`
- 支持变量注入后计算（如表达式中含 `x`、`y`）

该工具适合作为 Agent 的基础"计算器/数据处理"能力。

2.2 核心接口

```
def calculate(params: Dict[str, Any]) -> Dict[str, Any]:
```

输入 `params` 支持字段：

- `expression`（必填）：字符串形式的计算表达式
- `variables`（可选）：变量字典，用于替换表达式中的变量

2.3 关键实现逻辑

1. **参数校验**
    - 若 `expression` 为空，直接返回失败结构，并给出明确错误原因。
2. **变量替换**

- 若提供 `variables`，逐个把变量名替换为数值字符串。

3. **安全控制**
   - 通过正则 `^[0-9+\-*/().\s]+$` 限制表达式只能包含数字、四则运算符、括号和空格。
   - 再使用 `eval(expr, {"__builtins__": {}}, {})` 在"无内建函数"的受控环境执行，避免代码注入风险。
4. **结果与步骤返回**
   - 除了返回 `result`，还返回 `steps` 记录计算过程（原表达式 → 替换变量后的表达式 → 最终结果），便于解释型输出。

2.4 输出示例

成功时：

```
{
  "success": true,
  "data": {
    "expression": "1+2*3",
    "result": 7,
    "steps": ["1+2*3", "结果: 7"]
  }
}
```

失败时会捕获异常并返回 `"计算失败：xxx"`。

三、document.py —— 文档生成工具

3.1 功能定位

`document.py` 实现一个**可用大模型生成文档的工具**，当没有模型 Key 时自动降级为 Mock 文本。
该工具用于生成结构化 Markdown 文档，支持不同文档类型模板。

3.2 核心接口

```
def generate_document(params: Dict[str, Any]) -> Dict[str, Any]:
```

输入字段：

- `template`：文档模板类型（如 `"report" / "email" / "summary"`）
- `content`：文档主题/核心内容描述（必填）
- `data`：上下文补充信息（可选，dict）
- `format`：输出格式（默认 `"markdown"`）

3.3 关键实现逻辑

1. **提示词构建** `_build_document_prompt()`
   - 根据模板类型，先给出用途描述（报告/邮件/总结等）。
   - 拼接主题内容 `content`。
   - 若存在上下文 `data`，逐项写入提示词（dict/list 会 dump 为 JSON）。

- 针对 `"report"` 模板追加固定结构要求：
    1. 标题
    2. 摘要
    3. 主要内容（分点）
    4. 结论/建议
2. **优先使用大模型**
    - 如果配置了 `settings.LLM_API_KEY`：
        - 调用 `_generate_with_llm()` 通过 `LLMService` 生成正文。
        - 使用 `asyncio.run()`；若运行环境已有事件循环，则手动创建新 loop 运行，增强兼容性。
        - 统计字数 `word_count` 并返回。
3. **失败降级**
    - 若超时（>30s）或异常：
        - 直接降级到 Mock 文本（确保工具始终可用，不中断链路）。
        - 在 metadata 中标记 `"is_mock": True`。

3.4 输出特点

- 输出是标准 Markdown 文本，方便前端直接渲染。
- metadata 中额外附带：
    - `format`
    - `word_count`
    - `template`
    - `api_provider`（真实模型时为 `"llm"`）

四、news.py —— 新闻检索工具

4.1 功能定位

`news.py` 实现新闻搜索工具：
优先使用 **NewsAPI** 获取真实新闻；若无 Key 或调用失败，自动使用 Mock 新闻模板。

4.2 核心接口

```python
def search_news(params: Dict[str, Any]) -> Dict[str, Any]:
```

输入字段：

- `query`（必填）：搜索关键词
- `limit`：返回数量（默认 10）
- `category`：类别筛选（如 `"ai"` / `"科技"` / `"财经"` 等）

4.3 关键实现逻辑

1. **参数校验**
   - `query` 为空直接返回错误。
2. **真实 API 调用**
   - 若 `settings.NEWS_API_KEY` 存在：
     - 请求 `https://newsapi.org/v2/everything`
     - 参数包含关键词、语言、排序方式、返回数量等。
     - 若响应中包含 `articles`，提取标题、摘要、来源、链接、发布时间等结构化字段返回。
3. **错误处理**
   - 网络错误、API 格式错误、NewsAPI 返回错误码都会被捕获并打印日志。
4. **Mock 降级**
   - 无 Key 或真实调用失败时：
     - 按 `category` 选择对应 mock 模板集合（AI/科技/财经...）。
     - 若类别不匹配则默认 AI 类新闻。
     - 从模板中切片取前 `limit` 条返回。
   - metadata 标记 `"is_mock": True`。

4.4 返回结构

返回 `data` 内为新闻列表，每条含：

- `title`
- `description`
- `url`
- `source`
- `publishedAt`

五、stock.py —— 股票行情工具

5.1 功能定位

`stock.py` 实现股票行情查询：

- **美股（字母代码）** → 调用 Alpha Vantage 获取真实日线行情
- **A 股/不支持的代码** → 自动降级 Mock（保证工具可演示与可用）

5.2 核心接口

```python
def get_stock_data(params: Dict[str, Any]) -> Dict[str, Any]:
```

输入字段：

- `symbol`（必填）：股票代码或名称（如 `"AAPL"` / `"苹果"`）

- `days`：返回天数（默认 7）

5.3 关键实现逻辑

1. **参数校验**
   - `symbol` 空则失败返回。
2. **名称映射**
   - 内置 `stock_name_to_code`：
     - 如果输入是股票中文名（如“苹果”），转换成对应代码（如 AAPL）。
3. **真实 API 调用逻辑**
   - 若 `settings.STOCK_API_KEY` 存在：
     - 检测代码格式：
       - 若 `symbol.isalpha()` 且长度 1~5 → 认为是美股 → 调用 Alpha Vantage
       - 若为纯数字且长度≥5（A 股）→ Alpha Vantage 不支持 → 抛错进入 Mock
     - API 成功时解析 `"Time Series (Daily)"`：
       - 提取最近 `days` 天的开/高/低/收/成交量。
4. **Mock 价格生成**
   - 若进入 Mock：
     - 使用哈希构造“可复现”价格序列（同一股票同一天生成一致）。
     - 生成字段：date、open、high、low、close、volume。
   - metadata 中标记 `"is_mock": True`。

5.4 输出特点

- `data` 中包含：
  - `symbol`
  - `name`
  - `prices`（列表，按时间倒序）

六、总结

四个工具模块具备以下共同特点：

1. **接口统一、返回结构一致**
   方便上层 Agent 进行自动化调度与结果整合。
2. **真实能力 + Mock 降级并存**
   在无 API Key 或网络环境受限场景下，仍能稳定演示完整流程。
3. **实现强调可解释与可调试**
   - 输出 metadata（耗时、时间戳、是否 mock）
   - 打印调试日志（便于联调）
4. **可扩展性强**
   未来可按相同规范继续添加新的异构工具（如天气、日程、地图、邮箱等），与 Agent 工具链无缝联动。