F-22 - GitHub Copilot Tools in Agents

SUMMARY

In this lesson, we explored the integration of GitHub Copilot extensions with agents. We walked through the process of utilizing agent capabilities and demonstrated the significant enhancement when combined with web search. This integration allows us to harness the full power of up-to-date tools and APIs. Initially, we set up agents without any tools and observed limitations due to outdated SDKs. By enabling web search, we could access the latest SDKs and APIs, enhancing our agent's capabilities.

Here's a detailed step-by-step guide of what we covered:

1. **Setting Up an Agent:**
    - Navigate to GitHub Copilot chat and select the agent.
2. **Selecting Tools:**
    - Initially start with no tools and check agent behavior.
3. **Creating an Agent Example:**
    - Request the agent to create an example using the OpenAI agents SDK stack.
    - Without web search, it relied on outdated SDKs, generating incorrect code.
4. **Enabling Web Search:**
    - Go to settings (gear icon) and enable web search for Copilot.
    - Rerun the task allowing web search which resulted in the agent finding the latest SDK and generating correct code.
5. **Testing Complex Queries:**
    - We tasked the agent with creating a Telegram bot using the latest APIs.
    - Enabled multiple web searches to retrieve both telegram and weather API documentation.
    - Demonstrated the agent's ability to perform multiple searches before code generation.
6. **Understanding Extensions and MCP Server:**
    - Highlighted limitations of current extensions and introduced the more versatile MCP server option for tool integration.

WHAT WE LEARNED

- How to integrate and use GitHub Copilot agents.
- The importance of web search to obtain the latest tools and SDKs.
- Performing multiple searches to improve the functionality and correctness of code.
- Understanding the differences between basic extensions and more robust MCP server integrations.

HOW WE CAN APPLY IT

- Developing applications that require the latest libraries and tools.
- Automating processes that rely on external data sources, like APIs.
- Building chatbots or integration bots for platforms like Telegram.
- Enhancing project capabilities by switching from basic extensions to MCP server.

TIPS AND TRICKS

- **Always Enable Web Search:** To ensure that the agent uses the latest SDK or API, turn on web search and specify to use the "latest SDK" at the end of your prompt.
- **Use MCP Server for More Tools:** If current extensions don't meet your needs, consider integrating an MCP server for a broader range of applications.
- **Incremental Testing:** Start with basic functionality, then incrementally add web search and other tools to test the effects on outputs.

EXAMPLES

Basic Agent Setup

```
# Initial setup without web search
# This will likely use outdated SDKs
agent = OpenAI.agent()

# Prompting it to perform a task
response = agent.execute("Create an agent example")
```

Advanced Setup with Web Search

```
# After enabling web search
# Ensures accessing latest resources
```

```
search_settings = {
    "web_search": True
}

# Task prompting enhanced by web search
response = agent.execute("Use the latest OpenAI SDK to create an agent example", search_settings)
```

Telegram Bot Example

```
# Telegram bot code generation
# Automatically fetches necessary API details

def create_telegram_bot():
    api_key = "your_weather_api_key"
    chat_id = "your_chat_id"
    bot_token = "your_telegram_bot_token"

    # Functionality to send weather info every morning
    def get_weather_and_notify():
        # Implement API calls here
        pass
```

By the end of this lesson, we've laid the groundwork for more advanced integrations using GitHub Copilot with a focus on the advantages offered by enabling web search and extending functionalities through MCP servers.