AI Tutorial 101

Rajdeep Chatterjee, Ph.D.
Amygdala AI, Bhubaneswar, India *

January 2025

Agentic AI with Phidata

1 Setting up and using Python project management with `uv`

Below is an example showing an image of the AI Agent along with its AI Tools:

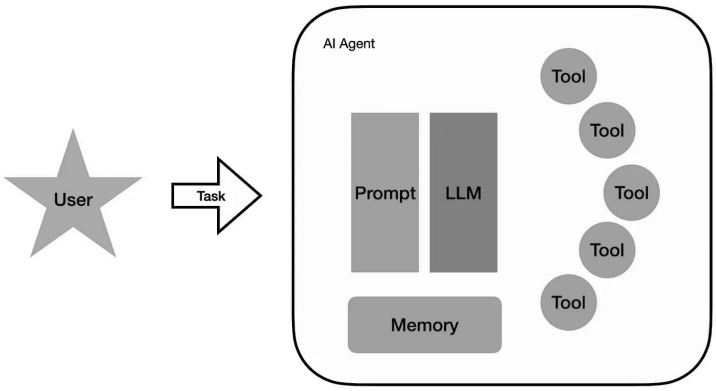

Figure 1: Block Diagram of AI Agent and Its Tools

1. **Open Windows PowerShell as Administrator.**
2. **Change the default path to the desired folder where the agent program will be saved:**

```
cd <desired-folder-path>
```

*Amygdala AI, is an international volunteer-run research group that advocates for *AI for a better tomorrow* `http://amygdalaai.org/`.

3. **Install the `uv` Python project management package:** Open `https://flocode.substack.com/p/044-python-environments-again-uv` in your browser and copy the following command into PowerShell:

```
Invoke-WebRequest -Uri https://astral.sh/uv/install.ps1 -OutFile install.ps1;
powershell -ExecutionPolicy Bypass -File ./install.ps1
```

4. **Install Python using `uv`:**

   - Install the latest version:

```
uv python install 3.12
```

   - Verify the installation:

```
uv python --version
```

5. **Initialize a new project:**

```
uv init -p 3.12 --name <project-name>
code .
```

   This opens the project folder in VS Code.

6. **Structure of the project:**

   - `hello.py`: Demo program to print "Hello, World!"
   - `pyproject.toml`: Project configuration file with installed dependencies.

7. **Run Python files using `uv`:**

```
uv run hello.py
```

8. `uv` **automatically generates a virtual environment:** Alternatively, you can manually create and activate a virtual environment.

9. **Install dependencies:**

```
uv add phidata duckduckgo-search groq
```

10. **Set up API keys:**

    (a) Log in to `Phidata.app` and copy the Phidata API key.

    (b) Log in to Groq Cloud, copy the Groq API key, and download a suitable open-source LLM model (e.g., `"llama-3.3-70b-versatile"`).

    Create a `.env` file in the project workspace:

```
PHI_API_KEY="phidata api key"
GROQ_API_KEY="groq api key"
```

11. **Create a web search agent:** Write the following code in `search_agent.py`:

```
from phi.agent import Agent
from phi.model.groq import Groq
from phi.tools.duckduckgo import DuckDuckGo
from dotenv import load_dotenv

load_dotenv()

SimpleSearchAgent = Agent(
    name="Web Agent",
    description="This is the agent for searching content
        from the web",
    model=Groq(id="llama-3.3-70b-versatile"),
    tools=[DuckDuckGo()],
    instructions="Always include the sources",
)
SimpleSearchAgent.print_response(
    message="What is the capital of India?", stream=True
)
```

Run the agent:

```
uv run search_agent.py
```

12. **Install Yahoo Finance tool:**

```
uv add yfinance
```

13. **Create a finance agent:** Write the following code in `finance_agent.py`:

```
from phi.agent import Agent
from phi.model.groq import Groq
from phi.tools.yfinance import YFinanceTools
from dotenv import load_dotenv

load_dotenv()

finance_agent = Agent
description="Your task is to find finance information",
    model=Groq(id="llama-3.3-70b-versatile"),
    tools=[
        YFinanceTools(
            stock_price=True,
            analyst_recommendations=True,
            company_info=True,
            company_news=True
        )
    ],
    instructions=["Use tables to display data"],
    show_tool_calls=True,
    markdown=True,
    debug_mode=True
)

finance_agent.print_response(
```

```
        "Summarize analyst recommendations for TSLA", stream=
            True
    )
```

Run the agent:

```
uv run finance_agent.py
```

14. **Create a multi-agent system:** Write the following code in `multi_agent.py`:

```
from phi.agent import Agent
from phi.model.groq import Groq
from phi.tools.duckduckgo import DuckDuckGo
from phi.tools.yfinance import YFinanceTools
import groq
import time
from dotenv import load_dotenv

load_dotenv()

web_search_agent = Agent(
    name="Web Agent",
    description="This is the agent for searching content
        from the web",
    model=Groq(id="llama-3.3-70b-versatile"),
    tools=[DuckDuckGo()],
    instructions="Always include the sources",
    show_tool_calls=True,
    markdown=True,
    debug_mode=True
)

finance_agent = Agent(
    name="Finance Agent",
    description="Your task is to find finance information"
        ,
    model=Groq(id="llama-3.3-70b-versatile"),
    tools=[
        YFinanceTools(
            stock_price=True,
            analyst_recommendations=True,
            company_info=True,
            company_news=True
        )
    ],
    instructions=["Use tables to display data"],
    show_tool_calls=True,
    markdown=True,
    debug_mode=True
)

agent_team = Agent(
    team=[web_search_agent, finance_agent],
    model=Groq(id="llama-3.3-70b-versatile"),
    instructions=["Always include sources", "Use tables to
         display data"],
    show_tool_calls=True,
```

```
        markdown=True,
        debug_mode=True
    )

    def rate_limited_response(agent, query):
        try:
            return agent.print_response(query, stream=True)
        except groq.APIStatusError as e:
            if "rate_limit_exceeded" in str(e):
                time.sleep(60)  # Wait 1 minute
                return agent.print_response(query, stream=True
                    )
            raise e

    # Use the multi-agent system like this:
    rate_limited_response(
        agent_team,
        "Summarize analyst recommendations and share the
            latest news for TSLA"
    )
```

Run the multi-agent system:

```
uv run multi_agent.py
```

15. **Step 20: Run the Image Agent.**

16. **Step 21: Create** `image_agent.py` **and write the following code:**

```
from phi.agent import Agent
from phi.model.groq import Groq
from phi.tools.duckduckgo import DuckDuckGo
from dotenv import load_dotenv
import os

load_dotenv()

# Initialize agent with Groq
agent = Agent(
    model=Groq(
        id="mixtral-8x7b-32768",  # Mixtral model through
            Groq
    ),
    tools=[DuckDuckGo()],
    instructions="Always provide detailed analysis and
        include sources for information",
    show_tool_calls=True,
    markdown=True,
    debug_mode=True
)

def analyze_image_and_search(image_url: str, query: str):
    """
    Analyze image and search for additional context using
        Groq and DuckDuckGo.

```

```
        Args:
            image_url (str): URL of the image to analyze.
            query (str): User's query about the image.
        """
        try:
            # Construct messages in the correct format
            messages = [
                {"role": "system", "content": "You are an AI
                    agent. Please analyze images and provide
                    detailed responses."},
                {"role": "user", "content": f"Analyze the
                    image at this URL: {image_url} and answer
                    the query: {query}"}
            ]

            # Run the agent with the formatted messages
            response = agent.run(messages=messages, stream=
                True)
            for chunk in response:
                print(chunk)

        except Exception as e:
            print(f"Error processing image: {str(e)}")

# Example usage
if __name__ == "__main__":
    # Test with an image
    image_url = "https://en.wikipedia.org/wiki/
        Kalinga_Institute_of_Industrial_Technology#/media/
        File:Kiit_library_building.jpg"
    query = "Tell me about the location and purpose of the
         building. Include any recent news or developments
        ."

    # Analyze image and get response
    analyze_image_and_search(image_url, query)
```

17. **Step 22: Run the agent:**

```
uv run image_agent.py
```