```
"""



tool decorated func [search_api] -> agent which parses the docs of the tool func

-> injected into prompt -> agent will output json containing tool usage -> agent output will be parsed

-> tool executed

-> terminal response can be returned to agent for self-healing



"""

import os

from dotenv import load_dotenv

# Import the OpenAIChat model and the Agent struct
from swarms import Agent
from swarm_models import OpenAIChat

# Load the environment variables
load_dotenv()


# Define a tool
def search_api(query: str, description: str):
    """Search the web for the query
    Args:
        query (str): _description_

    Returns:
        _type_: _description_
    """
    return f"Search results for {query}"


def weather_api(
    query: str,
):
    """_summary_

    Args:
        query (str): _description_
    """
    print(f"Getting the weather for {query}")


def rapid_api(query: str):
    """_summary_

    Args:
        query (str): _description_
    """
    print(f"Getting the weather for {query}")



# Get the API key from the environment
api_key = os.environ.get("OPENAI_API_KEY")

# Initialize the language model
llm = OpenAIChat(
    temperature=0.5,
)



## Initialize the workflow
agent = Agent(
    agent_name="Research Agent",
    llm=llm,
    max_loops=3,
    dashboard=True,
    tools=[search_api, weather_api, rapid_api],
    interactive=True,
    execute_tool=True,
)

# Run the workflow on a task
out = agent.run("Use the weather tool in Miami")
```

```
print(out)
```