Build an AI Agent for Weather & Search Queries Using Langchain, OpenWeather, and Tavily APIs (Notebook-based)

1. Objective

- Learn to build a Langchain-based AI agent that handles real-time weather and web search queries.
- Integrate the OpenWeather API and Tavily Search API as custom tools within the agent.
- Simulate a React-style conversational interface inside a Jupyter Notebook using Langchain's agent loop.

2. Problem Statement

- You are tasked to build an AI assistant agent that can answer questions like:
    - "What's the weather in Paris today?"
    - "Search for the latest news on AI in healthcare."

- The agent should:
    - Route queries to OpenWeather API for weather-related questions.
    - Route queries to Tavily Search API for real-time web search.
    - Use Langchain's tool routing capabilities to select the appropriate tool for each query.

3. Inputs / Shared Artifacts

- Azure OpenAI Resource: API key, endpoint URL, and deployment name (to be set as environment variables).
- API Keys required:

    - OpenWeather API key (https://openweathermap.org/api)
    - Tavily API key (https://app.tavily.com/)
- Python 3 Jupyter Notebook environment
- Required libraries (install with pip):

```
pip install langchain openai requests tavily-api tiktoken
```

4. Expected Outcomes

- A working notebook simulating a conversational agent with user input and AI responses.
- Langchain agent correctly routes queries to weather or search tools.
- Well-formatted, user-friendly responses for weather and search results.
- (Optional) Maintain chat history or simulate a chat interface using notebook cells.

5. Concepts Covered

- Langchain tools and agent setup
- API integration for OpenWeather and Tavily Search
- Tool routing and decision-making within Langchain agents
- Interactive agent loop simulation in Jupyter Notebook

6. Example: Step-by-Step Instructions with Code

```
# pip install langchain-openai langchain-community langchain-tavily langgraph
pyowm
import os
from langchain.tools import tool
from langchain_openai import AzureChatOpenAI
from langchain_community.utilities import OpenWeatherMapAPIWrapper
from langchain_tavily import TavilySearch
from langgraph.prebuilt import create_react_agent

# Step 1: Setup API keys
os.environ["AZURE_OPENAI_ENDPOINT"] = ""
os.environ["AZURE_OPENAI_API_KEY"] = ""
os.environ["AZURE_DEPLOYMENT_NAME"] = "GPT-4o-mini"

os.environ["OPENWEATHERMAP_API_KEY"] = ""
os.environ["TAVILY_API_KEY"] = ""

# Step 2: Define weather tool using Langchain wrapper
```

```
weather = OpenWeatherMapAPIWrapper()

@tool
def get_weather(city: str) -> str:
    """Get the current weather for a given city.
    Args: city (str): The name of the city to get the weather for.
    Returns: str: A string describing the current weather in the specified
city.
    """
    print(f" get_weather tool caliing: Getting weather for {city}")

    return weather.run(city)

# Step 3: Initialize Tavily search tool

tavily_search_tool = TavilySearch(
    max_results=1,
    topic="general",
)

# Step 4: Initialize Azure OpenAI LLM

llm = AzureChatOpenAI(
    azure_deployment=os.getenv("AZURE_DEPLOYMENT_NAME"),
    azure_endpoint=os.getenv("AZURE_OPENAI_ENDPOINT"),  # or your deployment
    api_version="2024-07-01-preview",  # or your api version
    api_key=os.getenv("AZURE_OPENAI_API_KEY"),
)

# Step 5: Setup Langchain agent with both tools

tools = [get_weather, tavily_search_tool]

agent = create_react_agent(
    model=llm,
    tools=tools,
)

print("Welcome to the AI assistant. Type 'exit' to stop.")
messages = []

# Mock user questions for automatic input
mock_questions = [
    "What's the weather in Hanoi?",
    "Tell me about the latest news in AI.",
    "Who won the last World Cup?",
    "exit",
]

for user_input in mock_questions:
    print("User: ", user_input)
    if user_input.lower() == "exit":
        print("Goodbye!")
```

```
        break
    messages.append({"role": "user", "content": user_input})

    response = agent.invoke({"messages": messages})
    messages.append({"role": "assistant", "content": response["messages"][-
1].content})

    print("AI: ", response["messages"][-1].content)
```

7. Final Submission Checklist

- Submit your .py file containing:
  - API key setup (with placeholders)
  - Tool definitions for OpenWeather and Tavily
  - Langchain agent initialization and conversation loop
- Use a dummy input list (auto input). Do not use manual input (input() built-in).
- Document your approach and any challenges faced.
- (Optional) Include enhancements such as response formatting or chat history.