

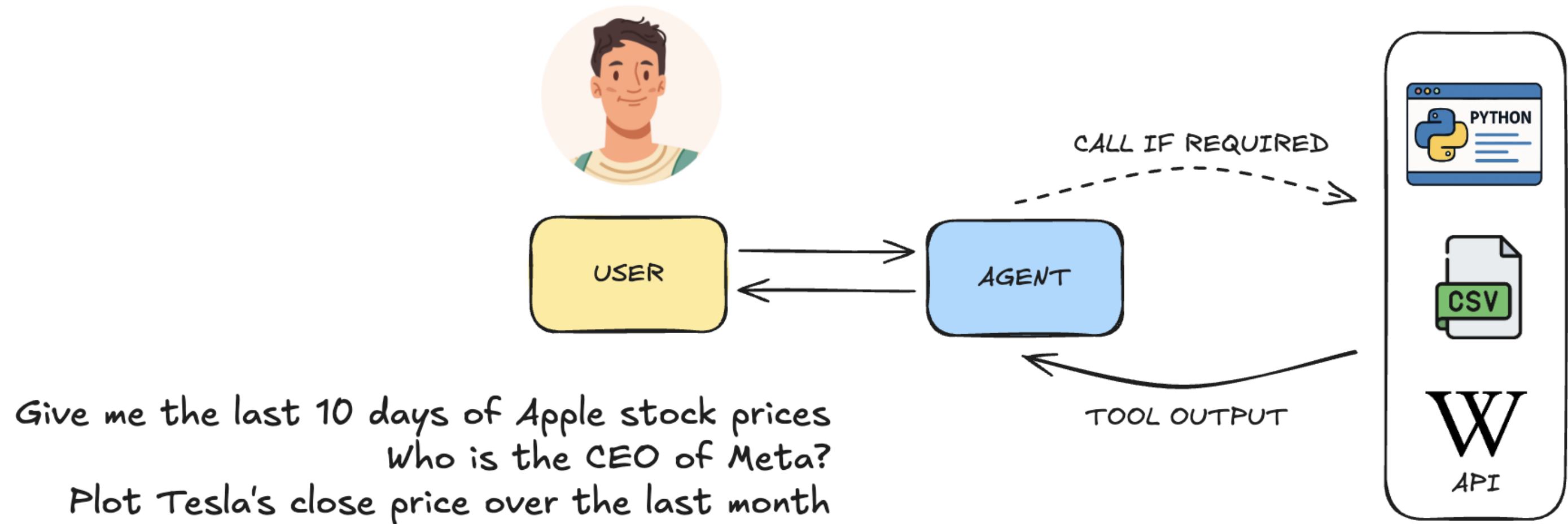
# Building agents the LangGraph way!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

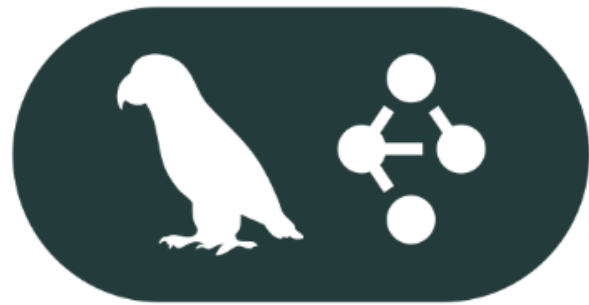


**James Chapman**  
Curriculum Manager, DataCamp

# What you'll build...

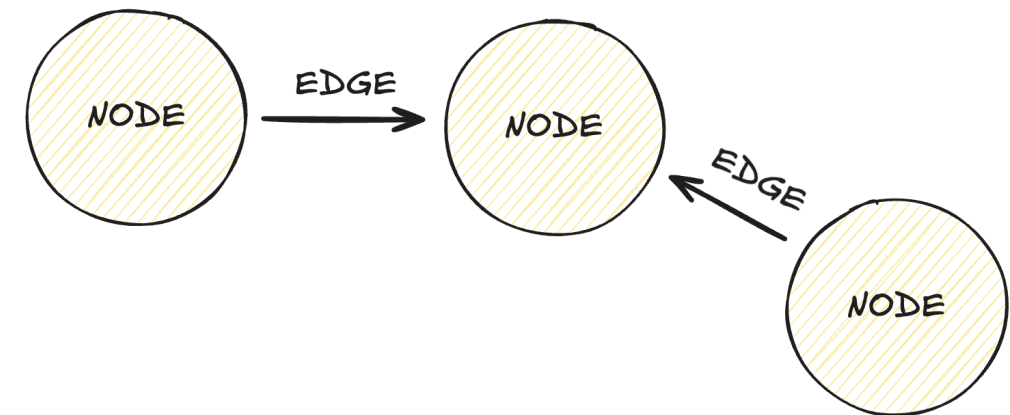


# Agents: the LangGraph way!



# LangGraph

- LangChain  $\square$  LangGraph
- *Build and orchestrate* production-ready agentic workflows



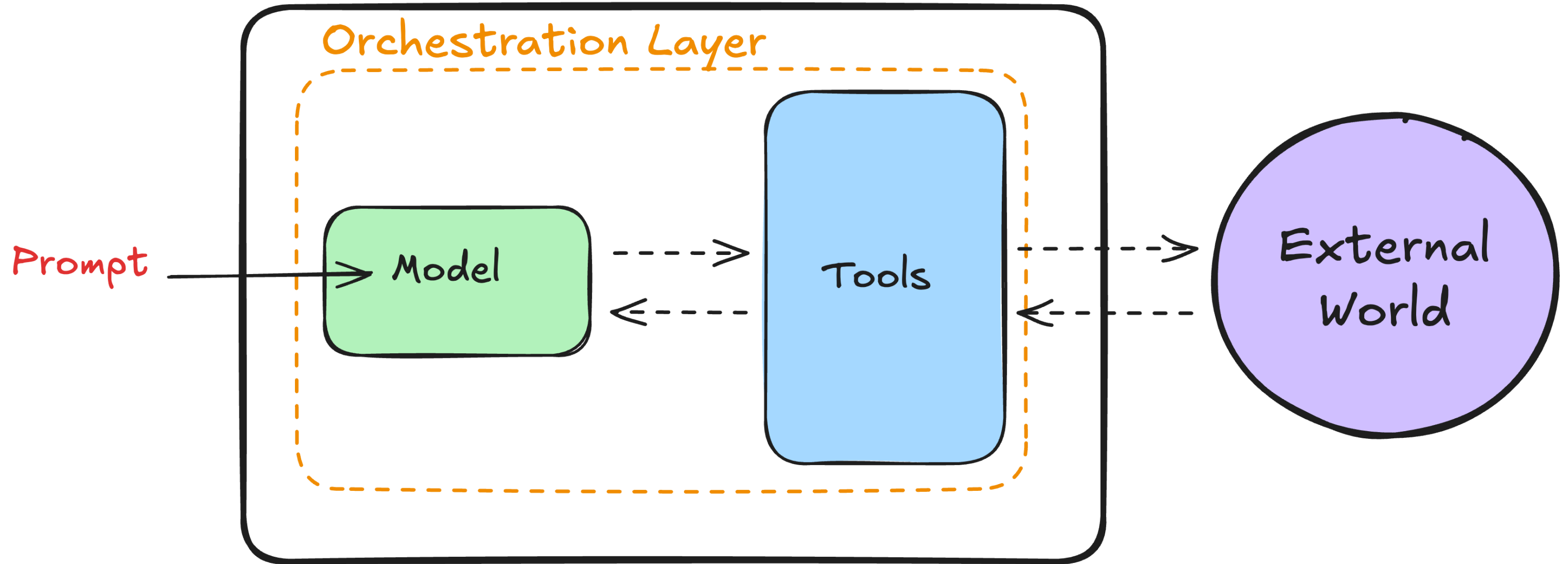
# Limitations of LLMs



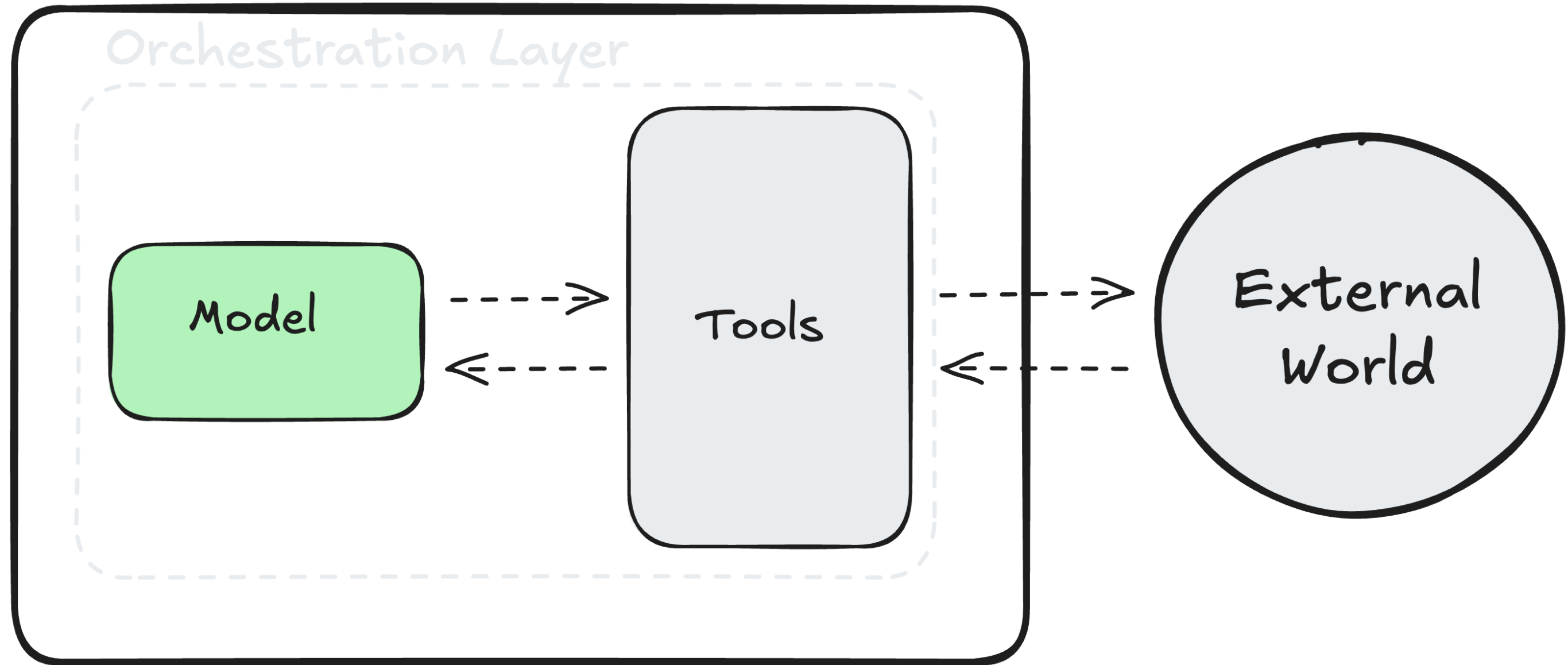
1. Knowledge cutoff based on training data
  2. Trained to generate specific modalities (text, images, etc.)
- LLM  $\neq$  Agent
  - Agents  $\rightarrow$  LLM + *Tools*

<sup>1</sup> Image created with GPT-4o

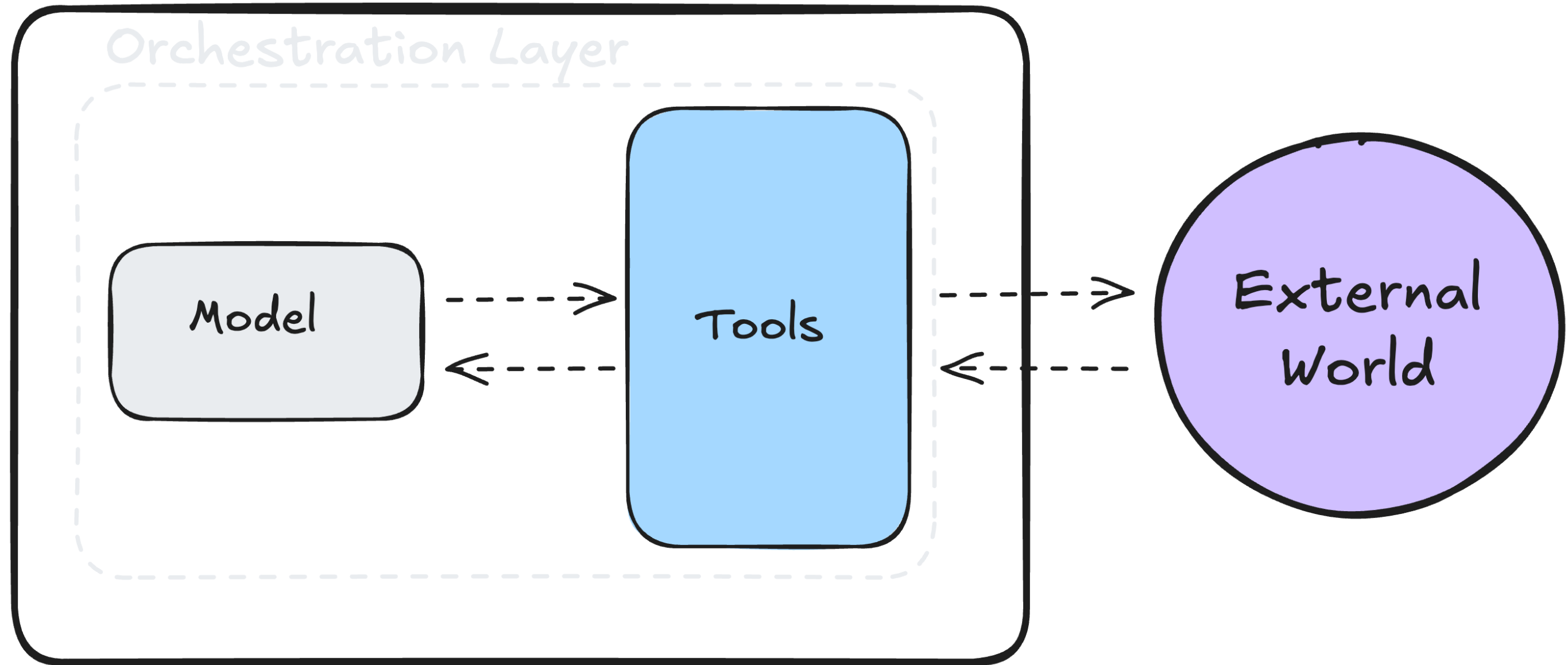
# Agents: the story from the inside



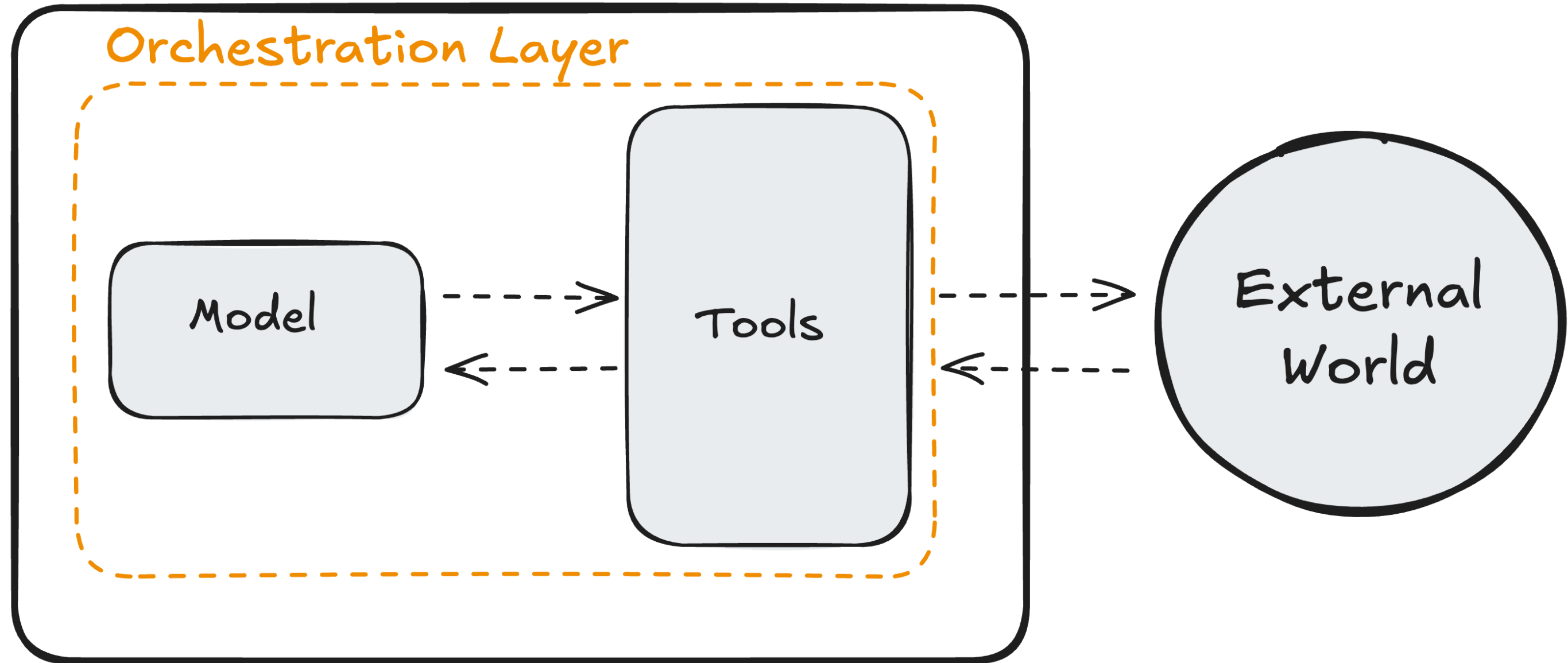
# Agents: the story from the inside



# Agents: the story from the inside

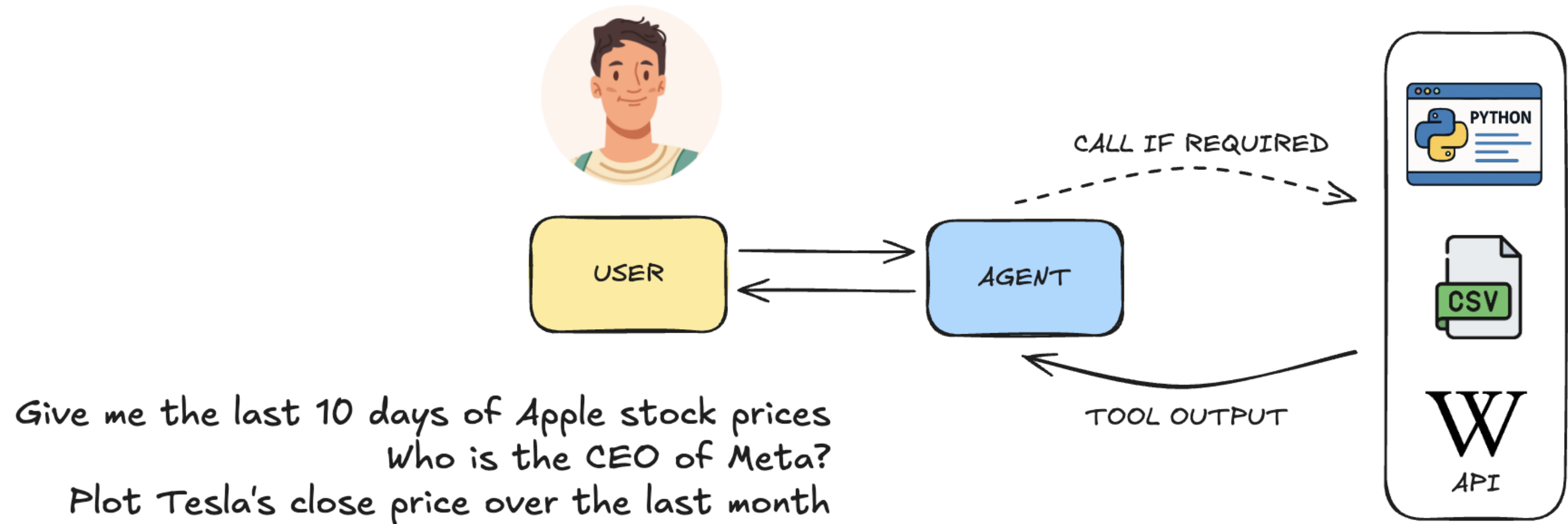


# Agents: the story from the inside

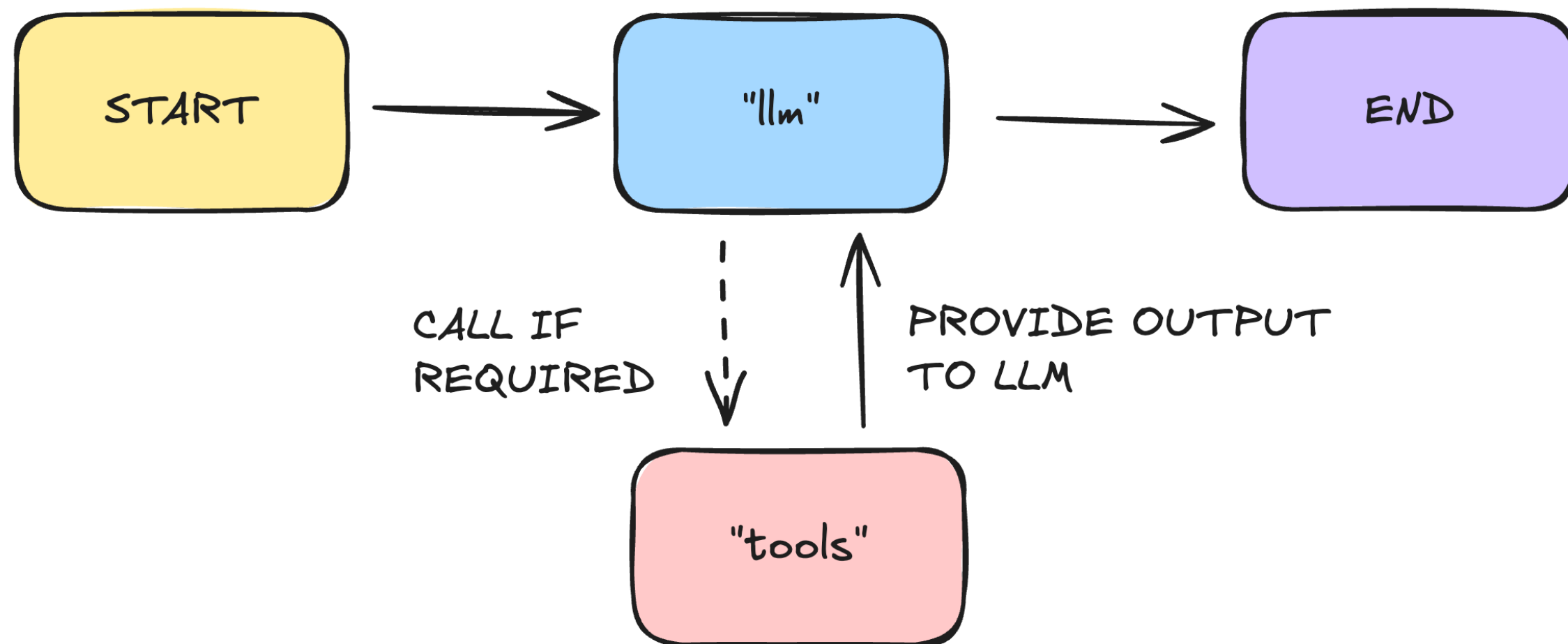


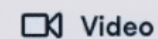


# What you'll build...



# Nodes and edges





Video



```
LangGraph's create_react_agent() function allows you to quickly instantiate a tool-calling agent from an LLM and a list of tools!

1 from langgraph.prebuilt import create_react_agent
2 from langchain_openai import ChatOpenAI
3
4 # Add three tools to the list: wikipedia_tool, stock_data_tool, and python_repl_tool
5 tools = [wikipedia_tool, stock_data_tool, python_repl_tool]
6
7 llm = ChatOpenAI(model="gpt-4o-mini")
8
9 # Create an agent using the create_react_agent function
10 prompt = """
11 You are an assistant for research and analysis of Fortune 500 companies. You have access to three tools:
12 - A Wikipedia tool for retrieving factual summary information about companies
13 - A stock performance data tool for retrieving stock price information from local CSV files
14 - A Python tool for executing Python code, which is to be used for creating stock performance visualizations
15 """
16
17 # Create an agent using the create_react_agent function
18 agent = create_react_agent(
19     llm,
20     tools=tools,
21     name="finance_assistant",
22     prompt=prompt
23 )
```

Play  
Instructions

100XP

Run the first two cells to install the libraries and re-define the tools

- Re-create your agent built with conditional edges using LangGraph's high-level `create_react_agent()` syntax, calling the agent `"finance_assistant"`.
- Re-run the visualization and chatbot code and compare it to what you found previously.

Take Hint (-30 XP)

File Environment Saving...



```
1 from langchain_openai import ChatOpenAI
2
3 # Add three tools to the list: wikipedia_tool, stock_data_tool, and python_repl_tool
4 tools = [wikipedia_tool, stock_data_tool, python_repl_tool]
5
6 llm = ChatOpenAI(model="gpt-4o-mini")
7
8 # Create an agent using the create_react_agent function
9 prompt = """
10 You are an assistant for research and analysis of Fortune 500 companies. You have access to three
11 tools:
12 - A Wikipedia tool for retrieving factual summary information about companies
13 - A stock performance data tool for retrieving stock price information from local CSV files
14 - A Python tool for executing Python code, which is to be used for creating stock performance
15 visualizations
16 """
17 # Create an agent using the create_react_agent function
18 agent = ----
```

This `agent` object is an already-compiled graph, so it can be visualized and interacted with in the same way as your previous graph to verify its equivalence:

```
# Visualize your graph
```



Run All

Submit Answer




Video

prompt=prompt

This agent object is an already-compiled graph, so it can be visualized and interacted with in the same way as your previous graph to verify its equivalence:

# Visualize your graph

agent



Here are the same prompts you tried with your graph with the conditional edge:

- Tell me about Apple Inc.
- AAPL stock price

100XP

Run the first two cells to install the libraries and re-define the tools

- Re-create your agent built with conditional edges using LangGraph's high-level `create_react_agent()` syntax, calling the agent `"finance_assistant"`.
- Re-run the visualization and chatbot code and compare it to what you found previously.

Take Hint (-30 XP)

File Environment

```
@tool
def python_repl_tool(
    code: Annotated[str, "The python code to execute to generate your chart."],
):
    """Use this to execute python code. If you want to see the output of a value,
    you should print it out with `print(...)`. This is visible to the user. The chart should be
    displayed using `plt.show()`. """
    try:
        result = repl.run(code)
    except BaseException as e:
        return f"Failed to execute. Error: {repr(e)}"
    return f"Successfully executed the Python REPL tool.\n\nPython code
    executed:\n`python`{code}\n\nCode output:\n`{result}`"
```

LangGraph's `create_react_agent()` function allows you to quickly instantiate a tool-calling agent from an LLM and a list of tools!


Define a one-line conditional tool-calling agent using LangGraph's pre-built functionality.

```
from langchain_openai import ChatOpenAI

# Add three tools to the list: wikipedia_tool, stock_data_tool, and python_repl_tool
tools = [wikipedia_tool, stock_data_tool, python_repl_tool]
```

Run All

Submit Answer

 datacamp

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Agent toolbox: Wikipedia tool

MULTI-AGENT SYSTEMS WITH LANGGRAPH



**Full Name**  
Instructor

# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Agent toolbox: stock data tool

MULTI-AGENT SYSTEMS WITH LANGGRAPH



**Full Name**  
Instructor



# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Agent toolbox: Python tool

MULTI-AGENT SYSTEMS WITH LANGGRAPH



**Full Name**  
Instructor

# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Agentic orchestration with LangGraph

MULTI-AGENT SYSTEMS WITH LANGGRAPH



Full Name  
Instructor

# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH

# Your first agent!

MULTI-AGENT SYSTEMS WITH LANGGRAPH



**Full Name**  
Instructor

# Let's practice!

MULTI-AGENT SYSTEMS WITH LANGGRAPH