

# Building Smart AI Agents with LangChain







## Speakers:

Siddharth Mehta

Harsh Tripathi



# What will we cover today?

-  Intro to AI Agents
-  Tech Behind Modern Agents
-  LangChain Core Concepts
-  Group Activity: Design an Agent
-  Live Demo: Smart Research Agent
-  Bonus: Model Context Protocol



# Before Agentic Workflows: Rule-Based & Static Logic

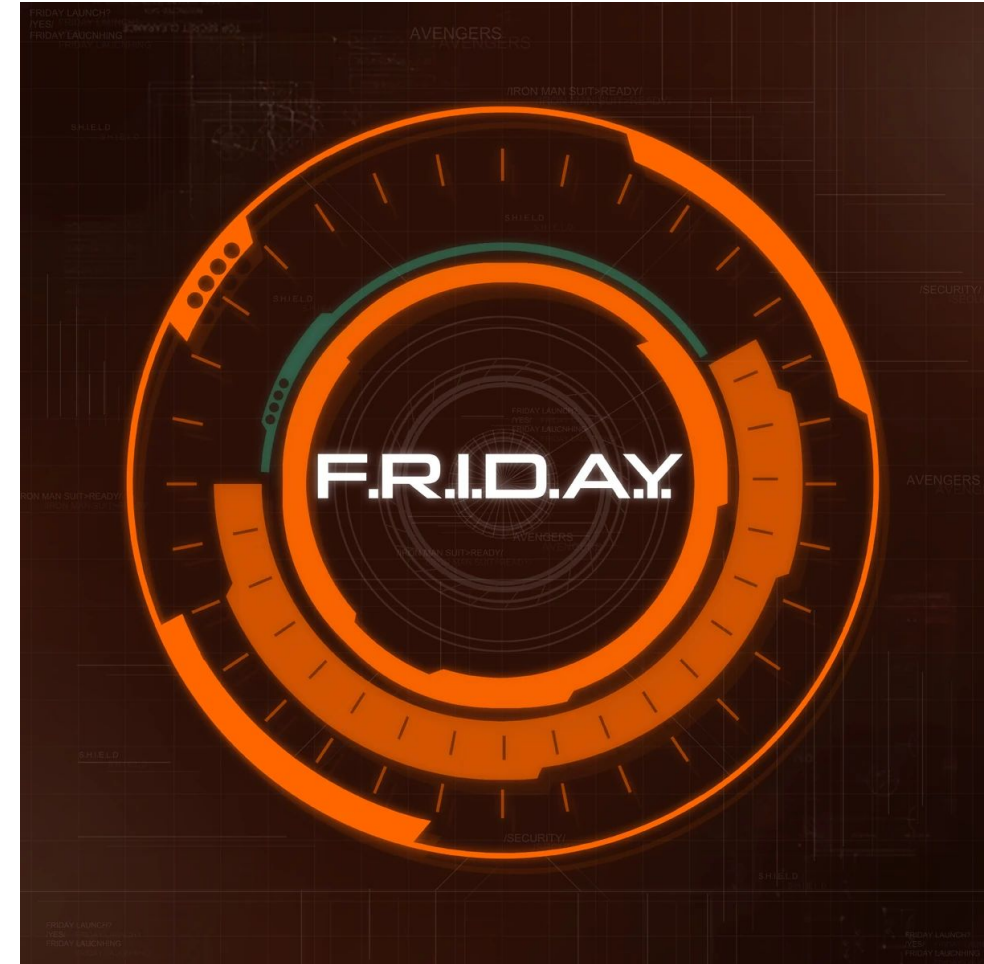
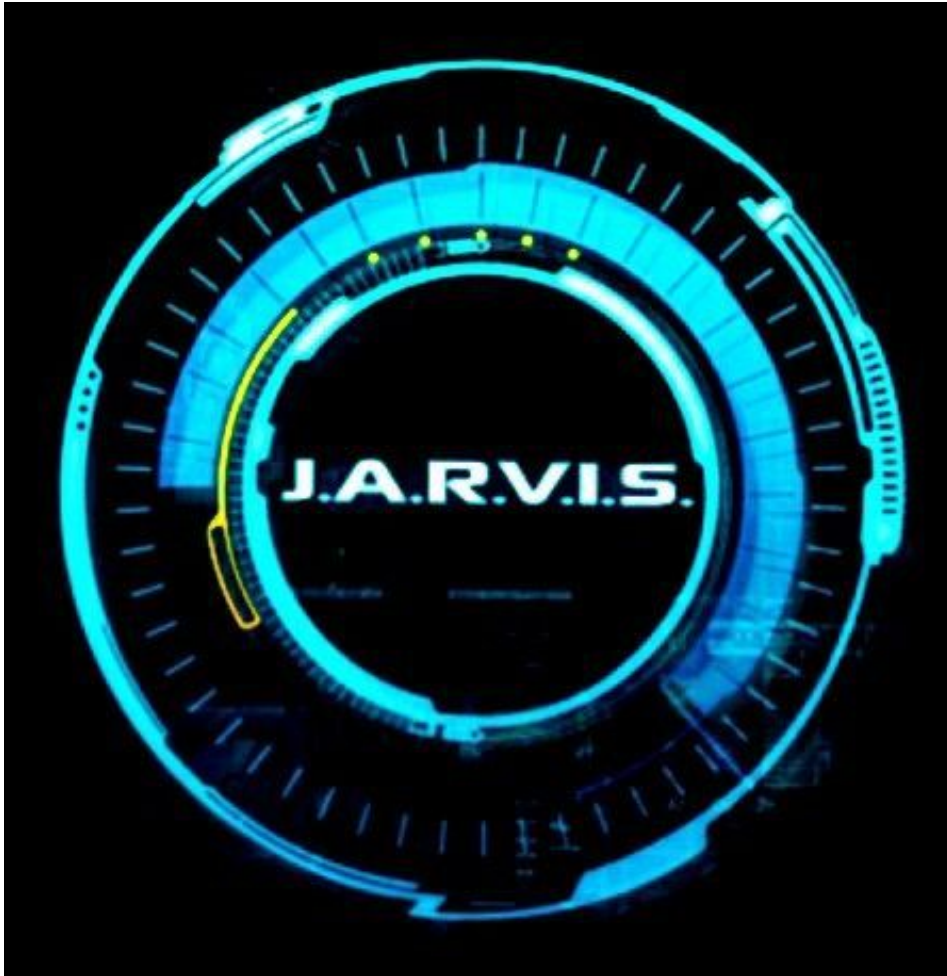
```
def chatbot():  
    print("Hello! I am a simple chatbot.")  
    print("You can ask me about weather, time, or say bye.")  
  
    while True:  
        user_input = input("You: ").lower()  
  
        if "weather" in user_input:  
            print("Bot: It's sunny and warm today.")  
        elif "time" in user_input:  
            from datetime import datetime  
            print(f"Bot: The current time is {datetime.now().strftime('%H:%M:%S')}")  
        elif "bye" in user_input:  
            print("Bot: Goodbye! Have a great day.")  
            break  
        elif "name" in user_input:  
            print("Bot: I'm ChatBot3000. Nice to meet you!")  
        else:  
            print("Bot: I'm not sure how to respond to that.")  
  
# Run the chatbot  
chatbot()
```

# Introduction to the world of AI Agents





So, what comes to your mind  
when you think of an AI Agent?



Fictional: AI Assistants from the movie - Iron Man



Amazon Alexa

Apple Siri



Google Home

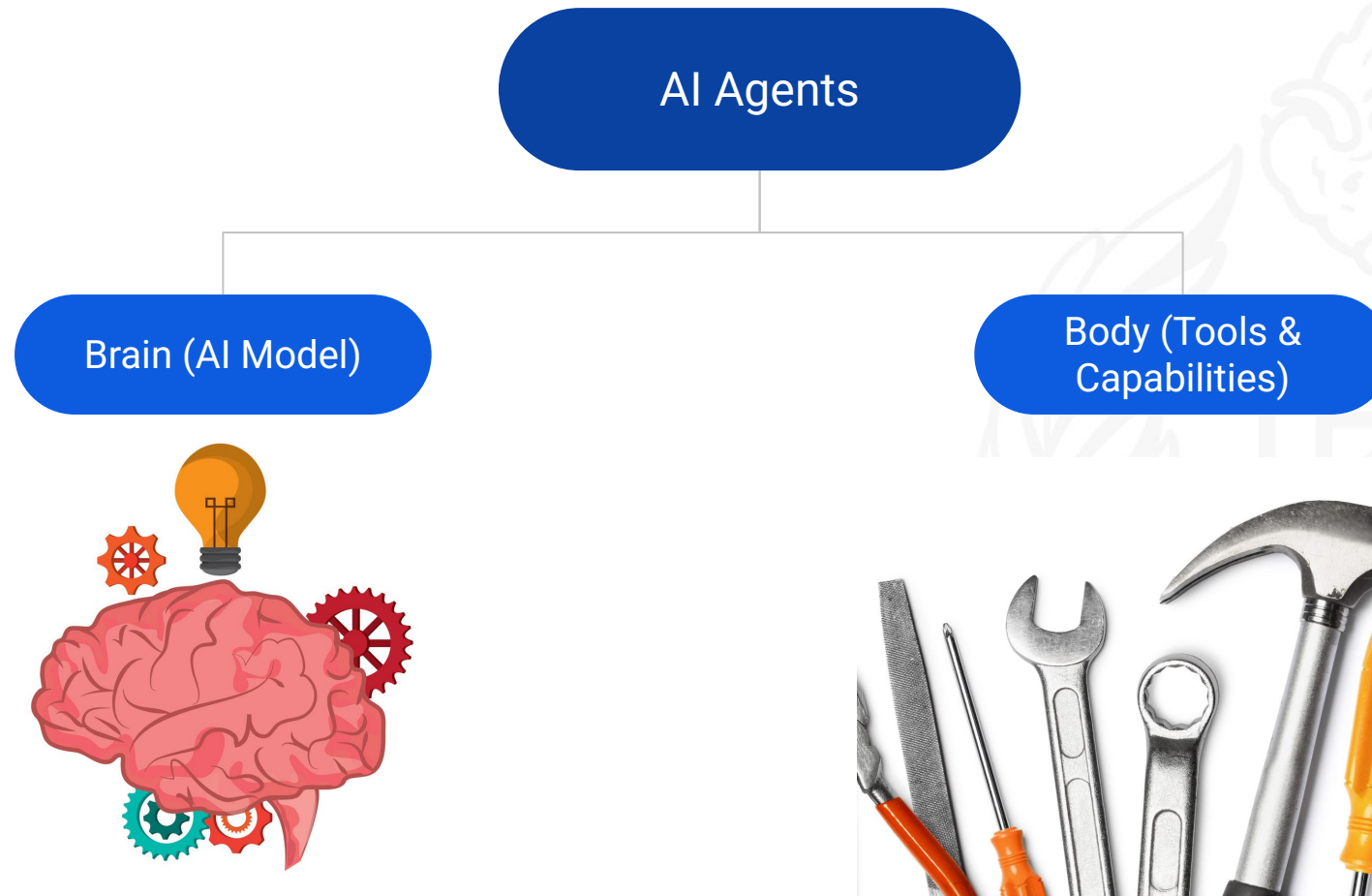


# Let's understand what are Agents...

## ***IBM Definition:***





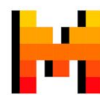
*AI agents are software programs designed to interact with their environment, collect data, and autonomously take actions to achieve predefined goals.*



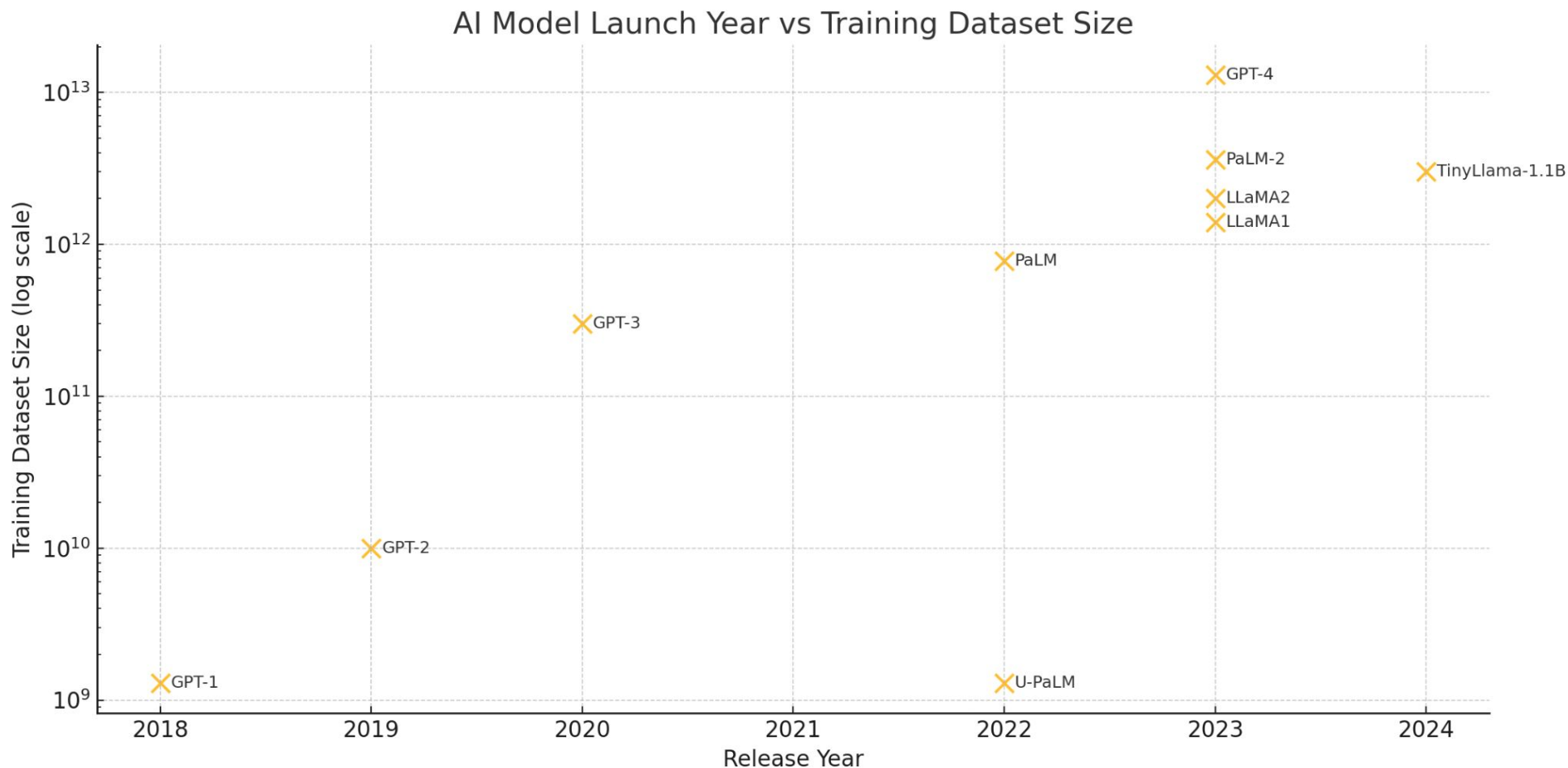


# Large Language Models (Agent's Brain)

- Next-Token Prediction models
- Trained on Huge datasets
- Use Transformer architecture (multi-head-attention mechanism)
- Few Shot and Zero shot learning

Model	Provider
Claude 3.7	
Deepseek-R1	
GPT-o3	
Gemini-2.5 Pro	
Mistral	

# Large Language Models (Agent's Brain)



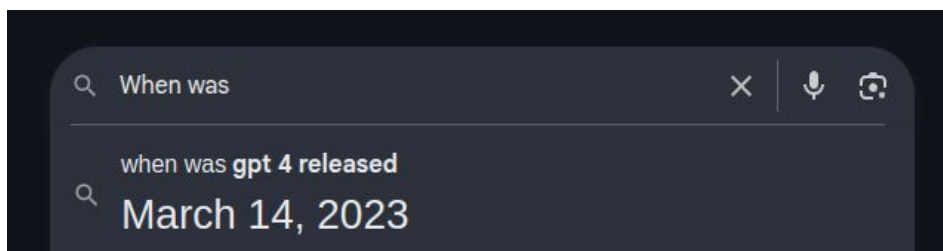
# Memory & Tools (Agent's Body)

Tool	Description
Web Search	Allows the agent to fetch up-to-date information from the internet.
Image Generation	Creates images based on text descriptions.
Retrieval	Retrieves information from an external source.
API Interface	Interacts with an external API (GitHub, YouTube, Spotify, etc.).
Memory	Stores previous interactions or facts to maintain context over time. Enables coherent, stateful conversations.



# Memory & Tools (Agent's Body)

Tool	Description
Web Search	Allows the agent to fetch up-to-date information from the internet.
Image Generation	Creates images based on text descriptions.
Retrieval	Retrieves information from an external source.
API Interface	Interacts with an external API (GitHub, YouTube, etc.).

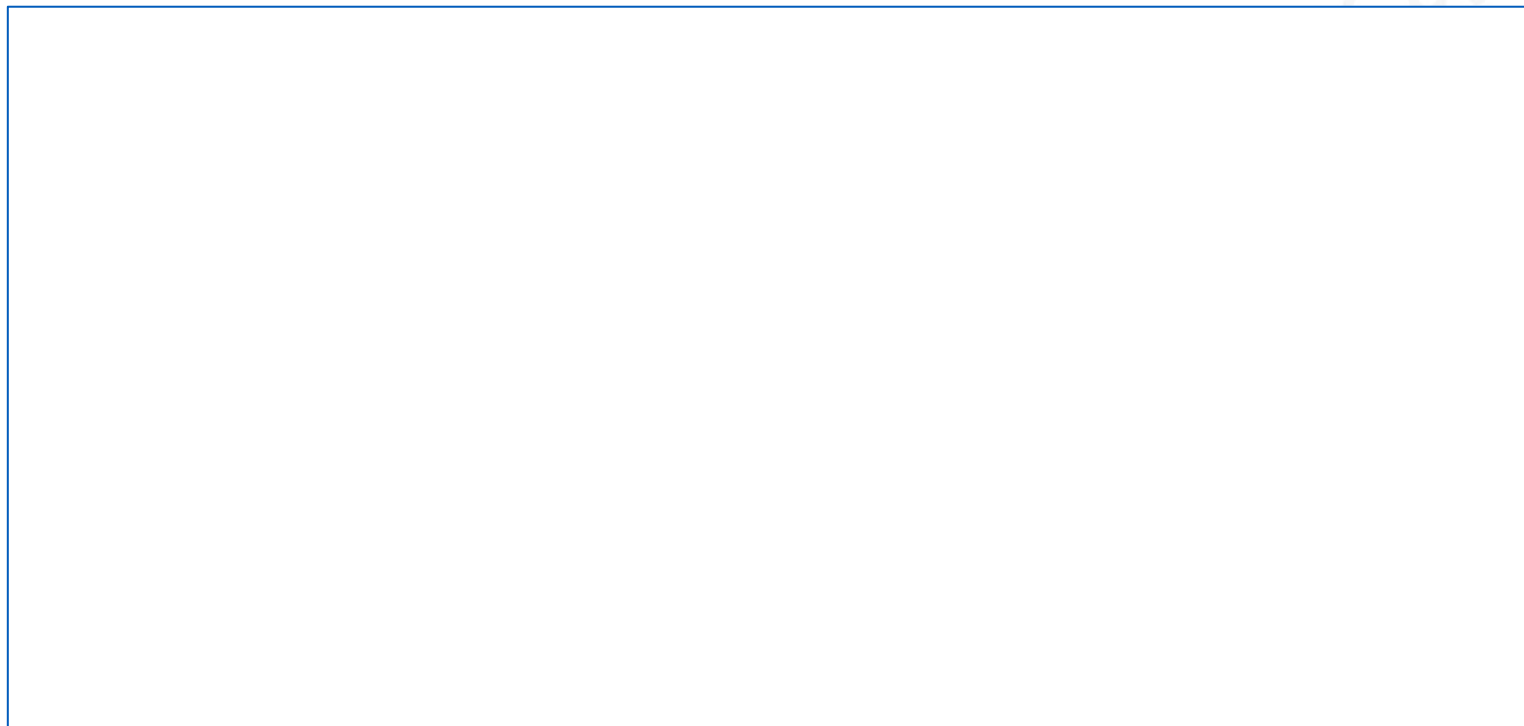


Web search Tools



Image Generation tools

# Agentic Workflow: How does it all come together?



GIF showcasing the orchestration of all the components

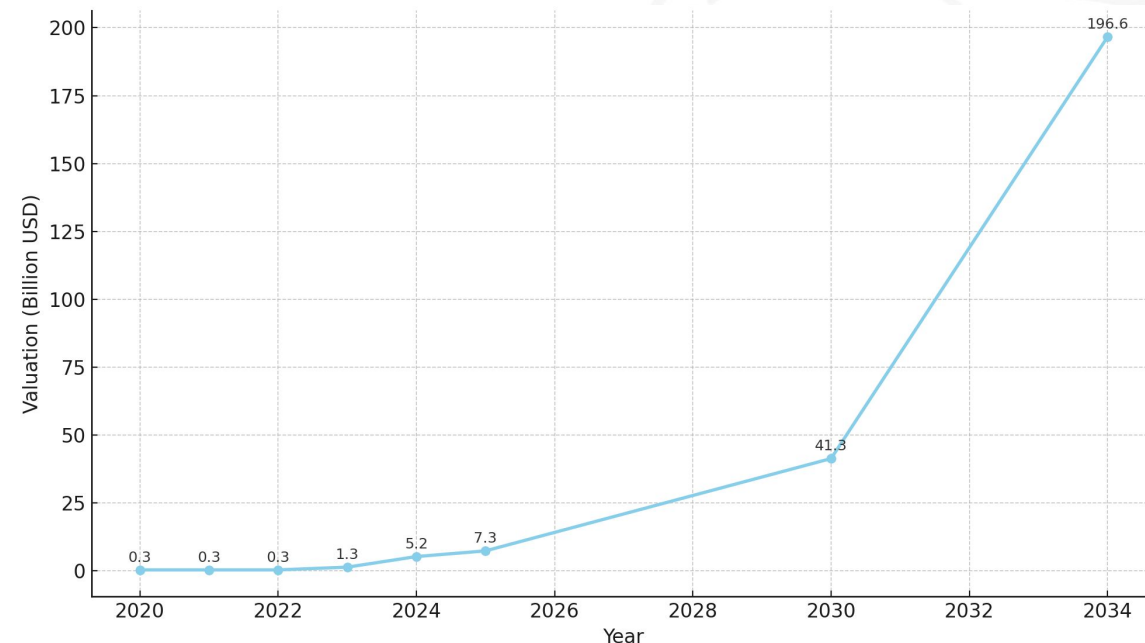
# Types of Agents

- ReAct Agents (Reasoning + Acting)
- Function/Tool-Calling Agents
- Structured Chat Agents



## Why Agents now?

- **Technological Breakthroughs in LLMs**
  - Multi-modal capabilities.
  - Reasoning capabilities
- **Compute Advancements enabling scale**
  - Training Acceleration
  - Distributed Computing
- **Market forces driving adoption**
  - Huge investments in Agentic AI companies



AI Agents Industry Valuation

All these factors enabled the paradigm shift that pushed AI Agents at the forefront of what we do today.



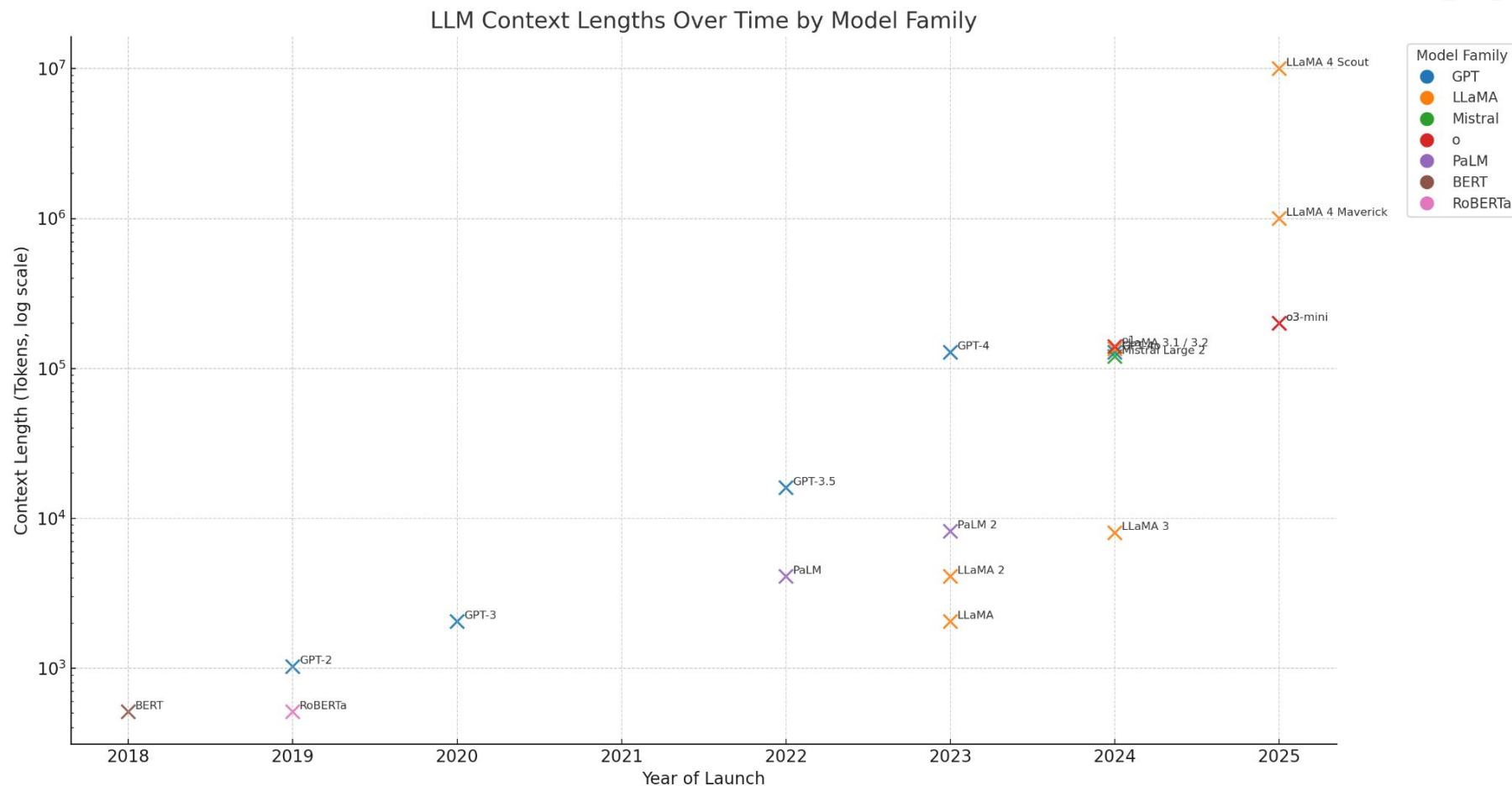
# Trends and Advancements in the field



# Context Length Advancements

- Context Length - Maximum amount of tokens that an LLM can process at an instant, effectively known as “working memory”
- Why is it important ?
  - More Context → More Input → More Information → Potentially more accurate outputs
  - All major AI labs use this as a competing metric for their models.
- Llama 4
  - 10M context window length(for comparison, the first ChatGPT had 4K tokens)
  - Applications unlocked: Document summarization, retrieval-augmented generation (RAG), and long-form conversational AI

# Context Length Advancements



# Agentic AI and Tools Advancements

- Protocol Advancements for Agents to interact with LLM's and inputs
  - MCP, A2A
- Advancements for decentralized use of Agents in the technology space
  - CrewAI
  - Replit Coding Agent
  - Google Firebase Studio



Advancements within last quarter

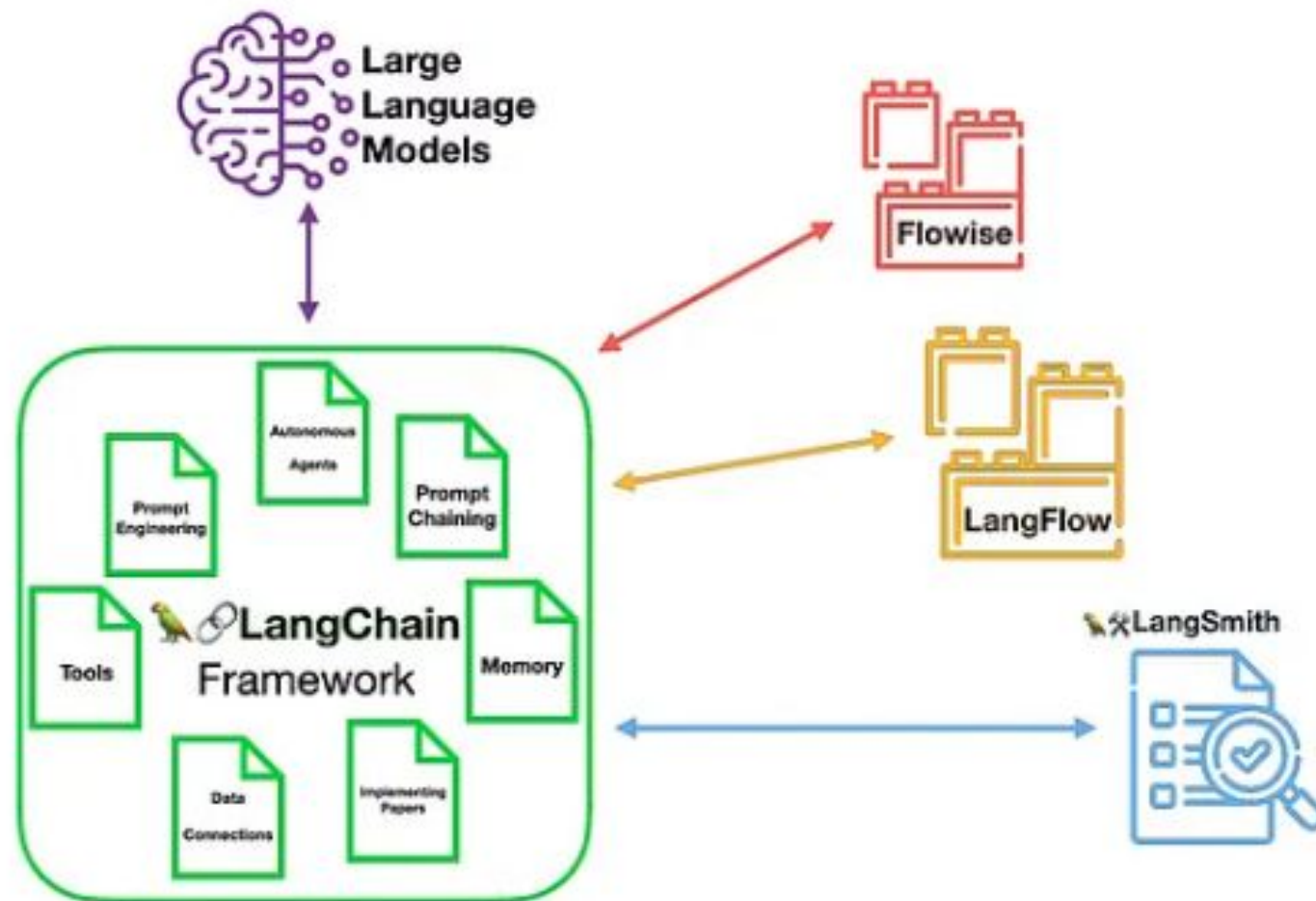


# LangChain



# LangChain

- Framework to build AI agents
- Open-source & modular design
- Connects LLMs with real tools
- Enables reasoning + decision-making
- Supports memory & context handling
- Powers multi-step task execution



# Most Frequently used code snippets

```
from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI(
    model_name="gpt-3.5-turbo",
    temperature=0.3
)
```

**Loads LLM** using APIs by OpenAI, Groq, Claude, etc.

# Most Frequently used code snippets

```
from langchain.tools import Tool

def get_weather(location: str) -> str:
    # Dummy function – replace with real API call
    return f"The weather in {location} is 25°C and sunny."

weather_tool = Tool(
    name="Weather Checker",
    func=get_weather,
    description="Use this to get the weather for a city"
)
```

**Define Tools** which can be any python function to perform variety of tasks



# Most Frequently used code snippets

```
from langchain.agents import initialize_agent, AgentType

agent = initialize_agent(
    tools=[weather_tool],
    llm=llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

**Initiate Agent** and provide it with tools, llm, agent type, and other parameters

# Most Frequently used code snippets

```
agent.run("What's the weather in San Francisco?")
```

**Query** the agent with user provided input.

## Setup Checklist:

- Python - 3.10
- Langchain, Numpy, pandas (Python packages)
- Hugging Face endpoint API



Please refer to the setup manual provided along in the readme file (resources provided in the QR)

# Group Activity: Design Your Own Agent

In the next 10 minutes, work in small groups (2-3 students) to design an AI agent for Productivity Boosting.

Think about:

- What problem does your agent solve?  
An example can be a task planner agent to prioritize tasks.
- What tools or data would it need?  
The agent above can use tools like Google Calendar APIs, Python executor, etc
- How will it interact with users?  
The users can chat with the agent to update their calendars, and plan with ease.

( 🕒 At the 5-minute mark, groups are welcome to volunteer and share a quick pitch of their idea with the room. )

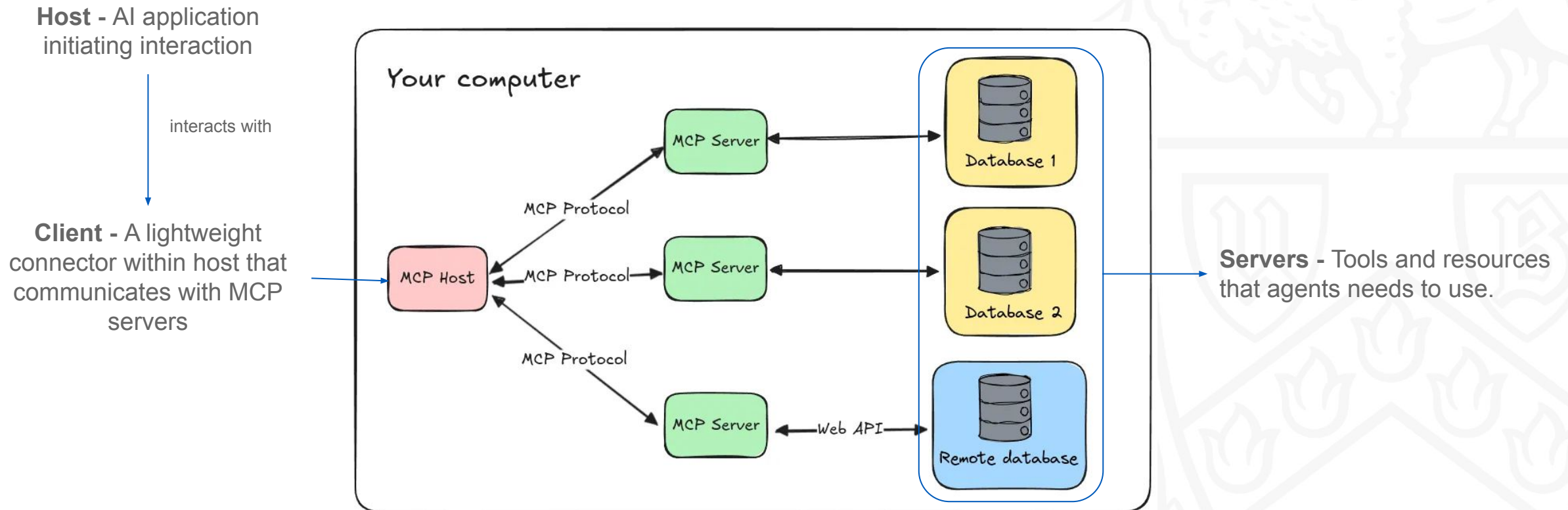


# Let's Code!

**BUT FIRST LET'S TAKE A SELFIE**



# Model Context Protocol



# Challenges with Agents

- **Reliability**

Agent can make errors, in complex environment, this leads to compounded mistakes

- **Transparency**

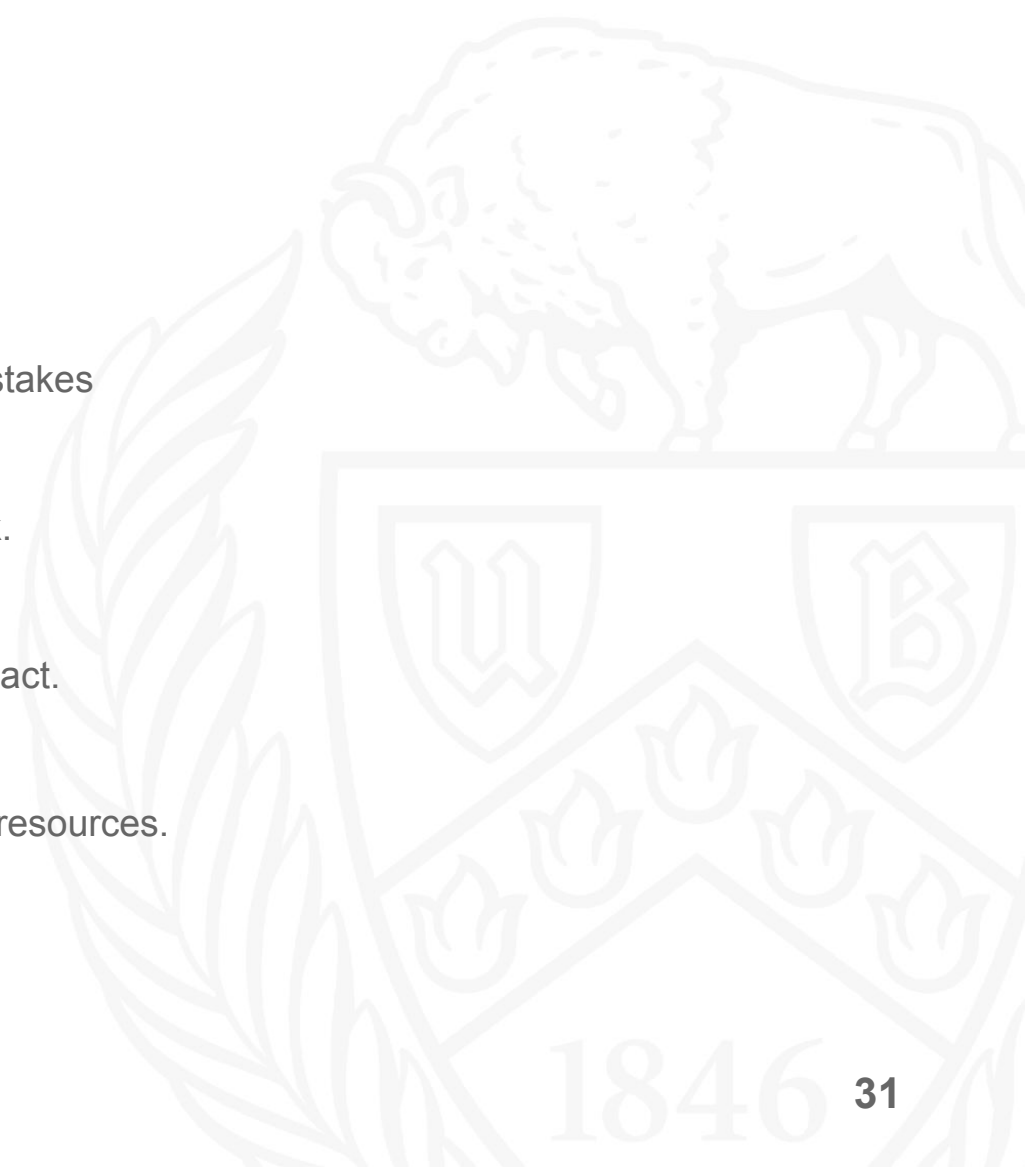
Understanding and explaining the decision making of LLMs can be difficult task.

- **Ethical Implications**

The deployment of AI agents must consider things like bias, privacy and its impact.

- **Resource Requirements**

Training and operating advanced AI agents demand substantial computational resources.



# In Summary

- **What are agents?**

Autonomous AI powered applications, that use reasoning + tools to take actions.

- **Why they matter?**

They transform static LLMs to dynamic and interactive system, and solve complex problems.

- **What's next?**

Now that we know what agents are -

*Let's explore, learn, and build the next generation agents - to provide solutions reliably.*

.... and the bonus quiz password is **1204**