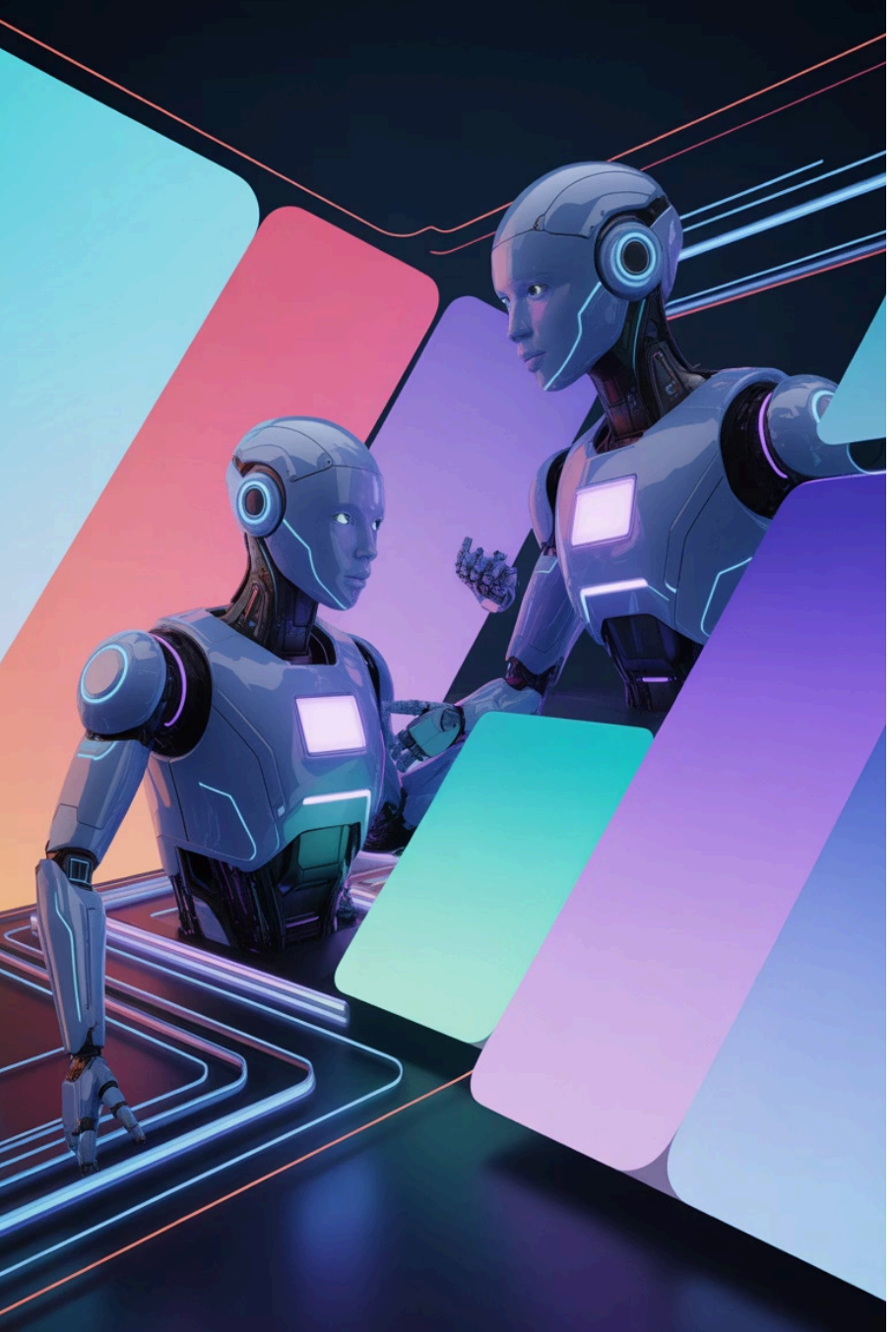


# AI Agents

A comprehensive exploration of how leading AI companies conceptualize autonomous intelligent systems

# What Are AI Agents? Industry Perspectives

## OpenAI Definition

AI agents are autonomous systems that can perceive their environment, make decisions, and take actions to achieve specific goals. They leverage large language models to understand context, reason through problems, and execute tasks with minimal human intervention.

## Google Gemini Definition

AI agents represent autonomous systems capable of perceiving, reasoning, and acting in complex environments. They understand context, adapt to changing conditions, and execute tasks by breaking down goals into actionable steps while learning from interactions.

## Anthropic Definition

AI agents are systems that can perceive information, process it to make decisions, and take actions toward achieving defined goals. They demonstrate autonomy in task execution while maintaining alignment with human values and intentions.

## Business Perspective

AI agents are software entities designed to automate complex tasks, optimize operational efficiencies, and drive strategic outcomes within an enterprise context. They function by autonomously interpreting business objectives, engaging with various systems and data sources, and executing a sequence of actions to achieve measurable value, thereby augmenting human capabilities and transforming workflows.

## Technical Perspective

From a technical standpoint, AI agents are computational systems orchestrated around a core reasoning engine, typically powered by large language models, that interact with an environment through sensors (perception) and effectors (actions). They are characterized by their ability to maintain an internal state, engage in planning, learn from experience, and adapt their behavior to achieve predefined goals, often incorporating modular architectures for specialized functions.

# Evolving from Traditional Automation to AI Agents

## Automation/RPA/Traditional ML Workflow

- Fixed rule-based processes with limited adaptability
- Requires extensive pre-programming for each scenario
- Cannot handle exceptions or unexpected situations
- Limited learning capabilities from new data
- Brittle when business processes change
- Requires constant human oversight and maintenance
- Struggles with unstructured data and complex reasoning

## AI Agents

- Dynamic decision-making with contextual understanding
- Self-adapting to new scenarios and requirements
- Handles exceptions and edge cases intelligently
- Continuous learning from interactions and feedback
- Resilient to process changes and variations
- Operates autonomously with minimal supervision
- Excels at processing unstructured data and complex reasoning

Yesterday workflows decided the output; AI Agents decide the decision

# Taxonomy of AI Agents


### Level 0: Core Reasoning System

The foundational language model, isolated.


### Level 1: Connected Problem-Solver

Language model augmented with tools for real-world interaction.


### Level 2: Strategic Problem-Solver

Adds context engineering and advanced planning capabilities.


### Level 3: Collaborative Multi-Agent Systems

Teams of specialized agents coordinating to achieve goals.


### Level 4: Self-Evolving Systems

Agents capable of autonomously building new tools or spawning new agents.

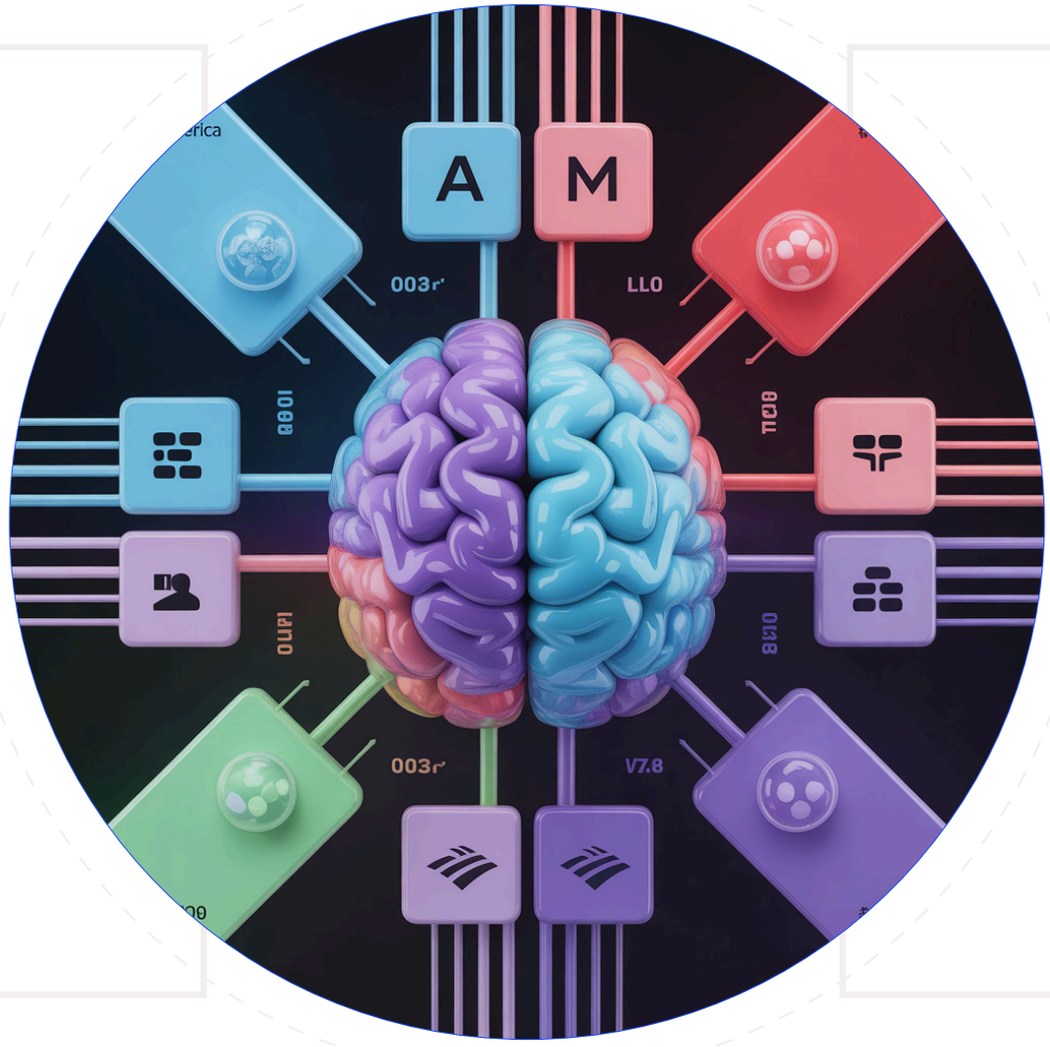
# Agent Building Blocks

### Brain / LLM

Reasoning core with neural patterns

### Tools / Hands

APIs, interfaces, action capabilities



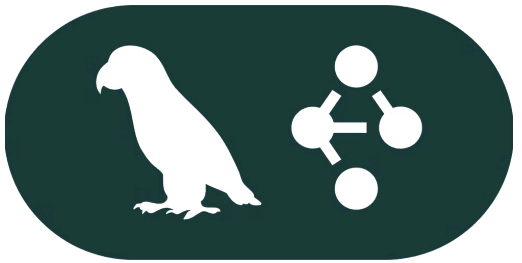

### Memory

Knowledge stores, retrieval, learning

### Body / Orchestra

Coordination and system management

# Common AI Frameworks (Pro Code)



LangGraph


AutoGen

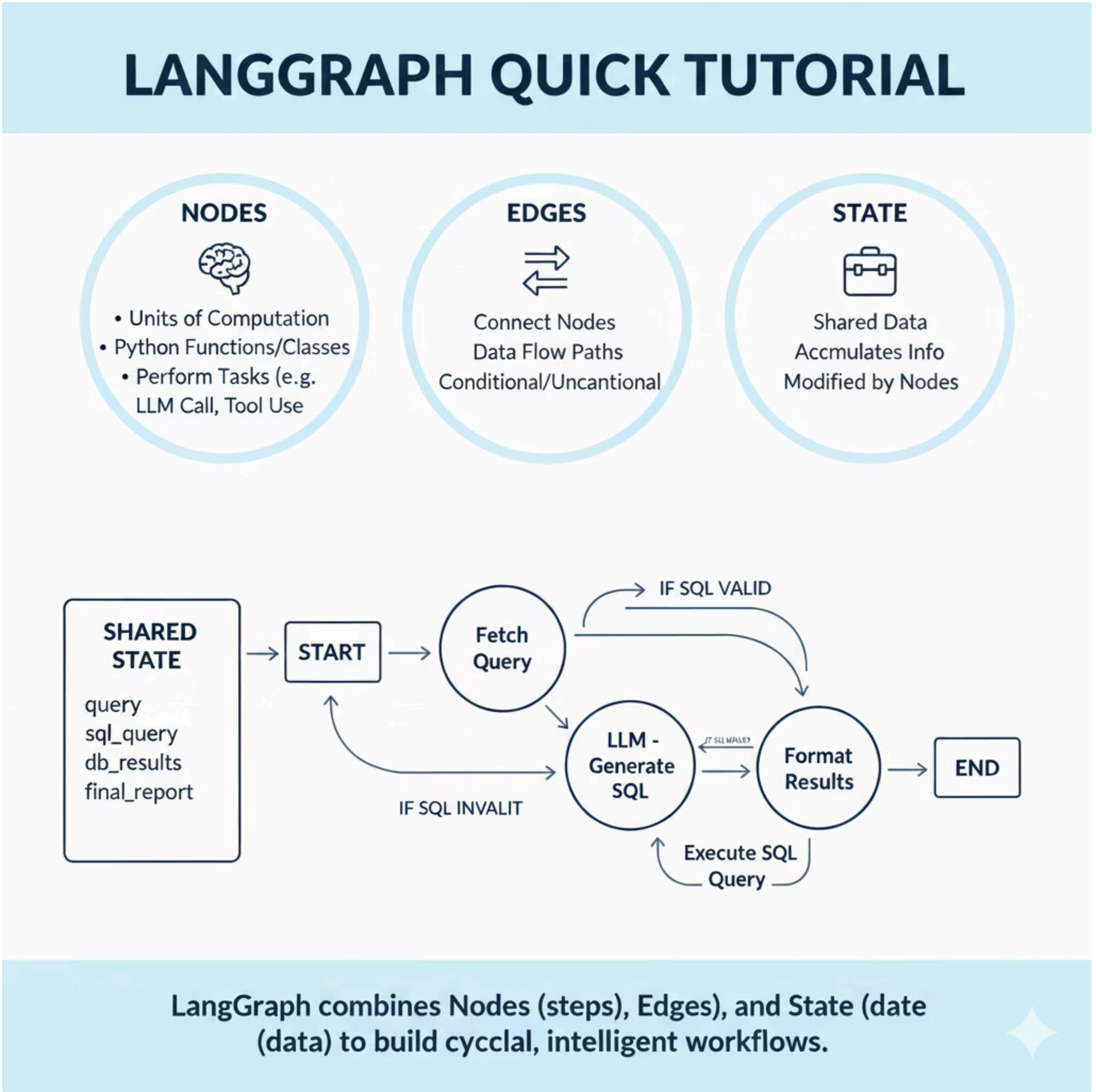

CrewAI


LlamaIndex

Haystack


Semantic Kernel

# LangGraph


LANGGRAPH QUICK TUTORIAL
NODES
• Units of Computation
• Python Functions/Classes
• Perform Tasks (e.g. LLM Call, Tool Use
EDGES
Connect Nodes
Data Flow Paths
Conditional/Uncantional
STATE
Shared Data
Accmulates Info
Modified by Nodes
SHARED STATE
query
sql_query
db_results
final_report
START
Fetch Query
IF SQL VALID
LLM - Generate SQL
Format Results
END
IF SQL INVALIT
Execute SQL Query
LangGraph combines Nodes (steps), Edges), and State (date (data) to build cycclal, intelligent workflows.

# Basic Implementation

## Installation:

```
pip install langgraph
```

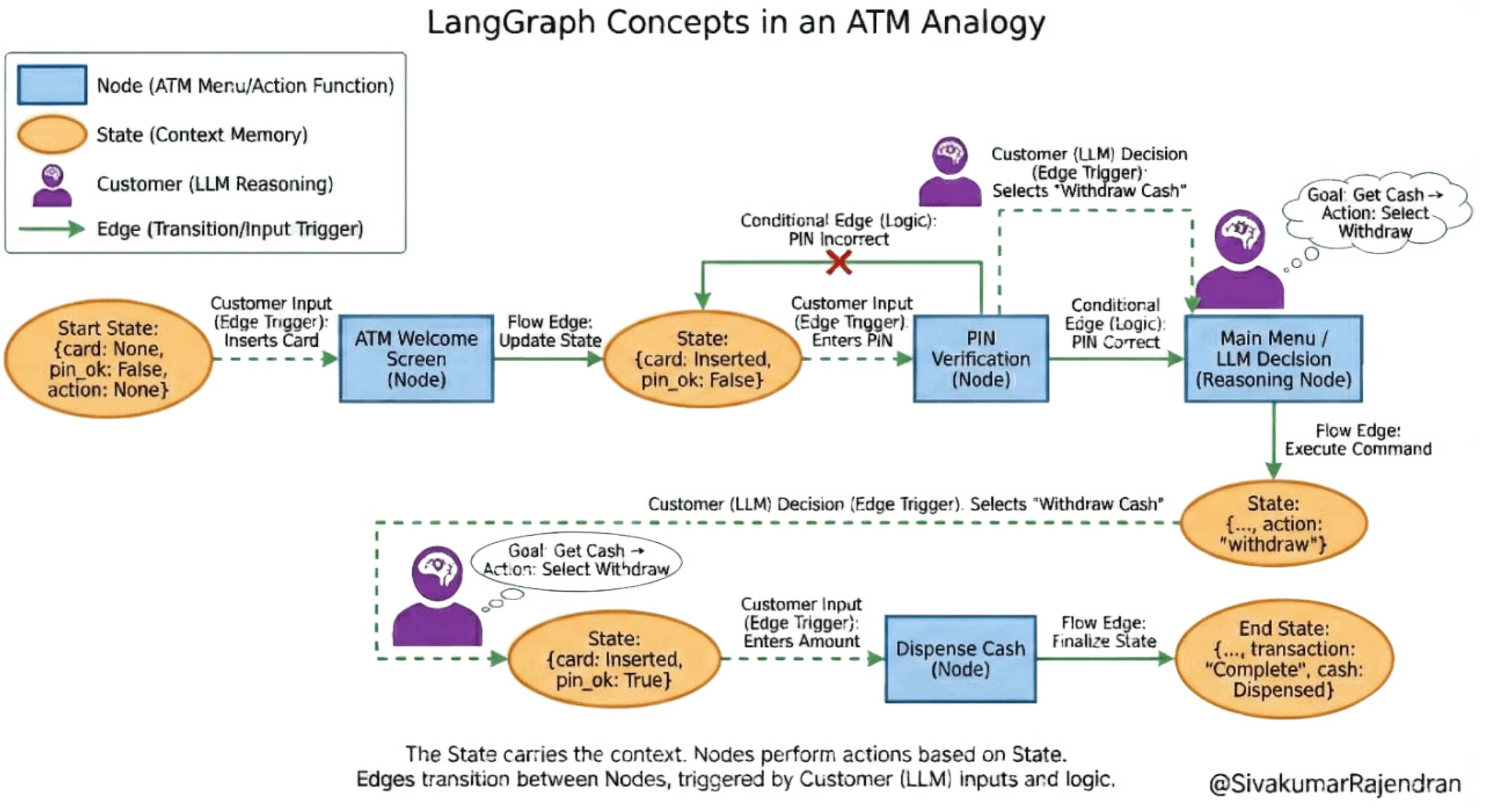
## Simple Example - Basic Chatbot:

```
from langgraph.graph import StateGraph
from typing import TypedDict

# Define state structure
class State(TypedDict):
    messages: list

# Define nodes
def chatbot_node(state: State):
    # Process user message
    response = generate_response(state["messages"][-1])
    return {"messages": state["messages"] + [response]}

# Build the graph
workflow = StateGraph(State)
workflow.add_node("chatbot", chatbot_node)
workflow.set_entry_point("chatbot")
workflow.set_finish_point("chatbot")

# Compile and run
app = workflow.compile()
result = app.invoke({"messages": ["Hello!"]})
```

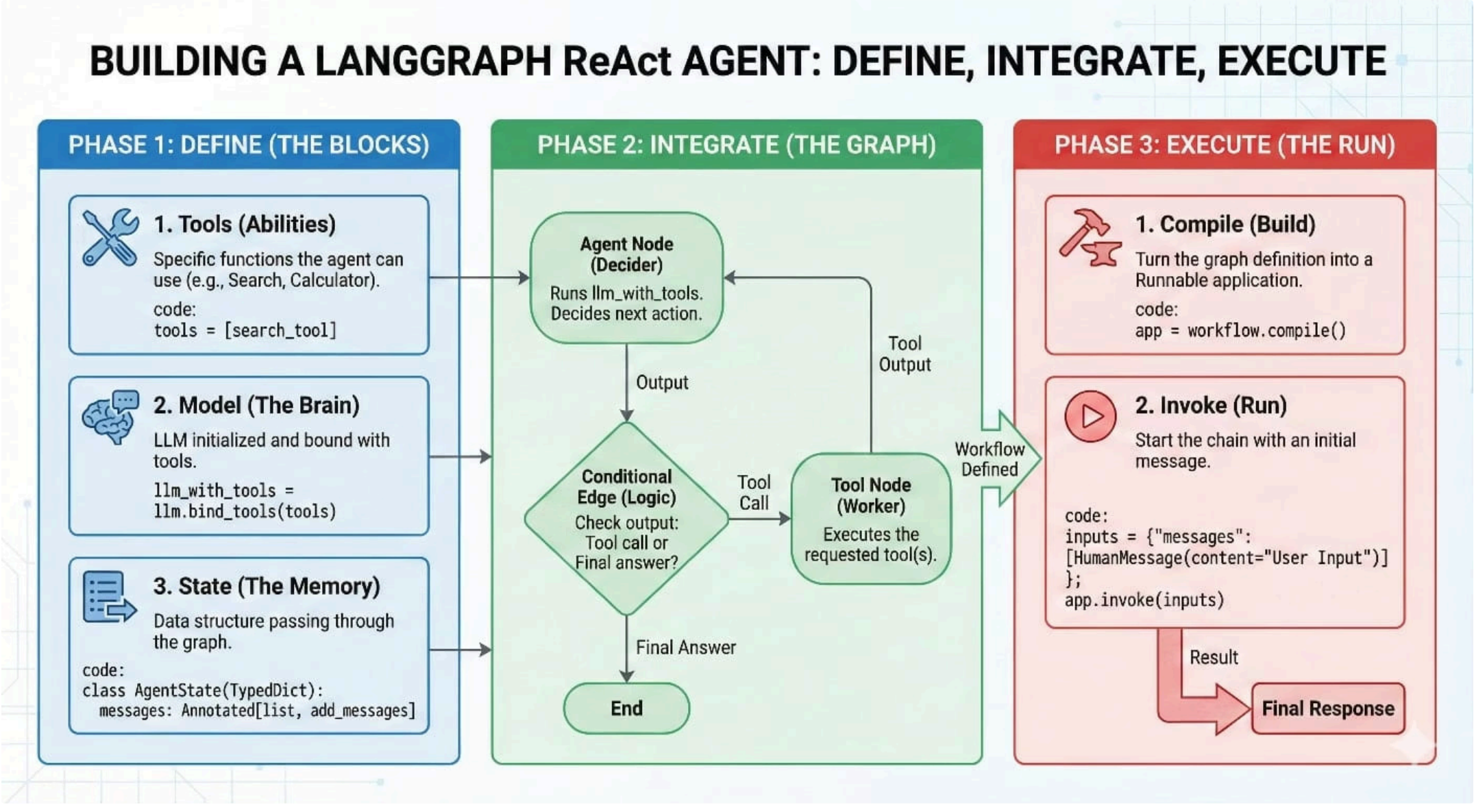
## Key Steps:

1. Define your state structure
2. Create node functions
3. Build the graph with nodes and edges
4. Compile and execute

# LangGraph Concepts in ATM Analogy



# Building a LangGraph ReAct Agent: Define, Integrate, Execute



# Let's create a simple agent