global AI bootcamp

Agentic AI with AutoGen

A framework for building AI agents and applications

Surasuk Oakkharaamonphong

Microsoft MVP AI Platform & Developer Technologies

Technical Coach at Arise & Infinitas by Krungthai

MVP Microsoft Most Valuable Professional

Arise by INFINITAS

INFINITAS by Krungthai

global AI community

#GlobalAIBootcamp

PRE-TEST

ASTER SHARING: AGENTIC AI WITH AUTOGEN

Arise Learning
Learning.px.group@arise.tech

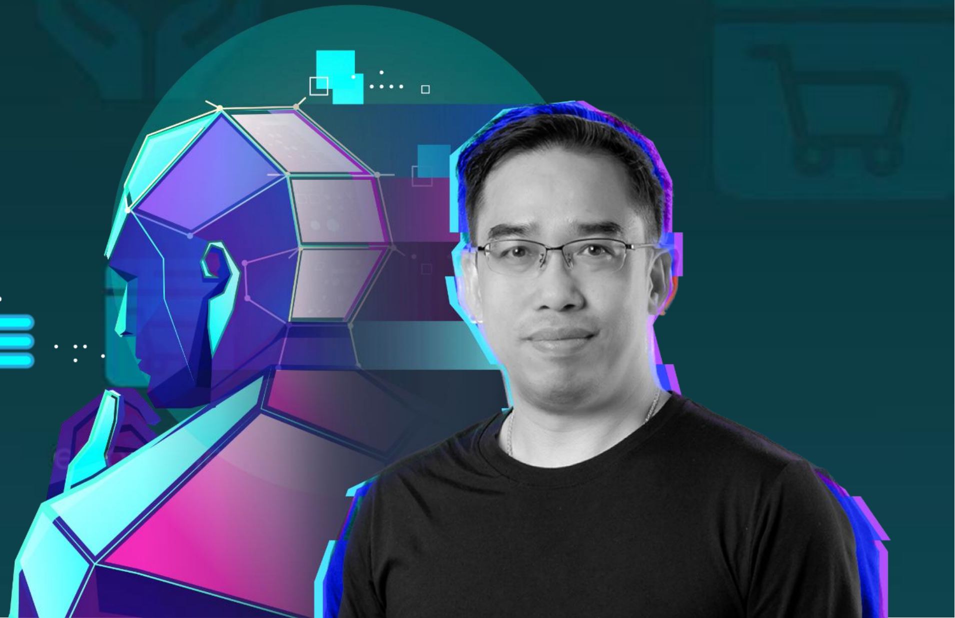
#GlobalAIBootcamp

Agentic AI with AutoGen

A framework for building AI agents and applications

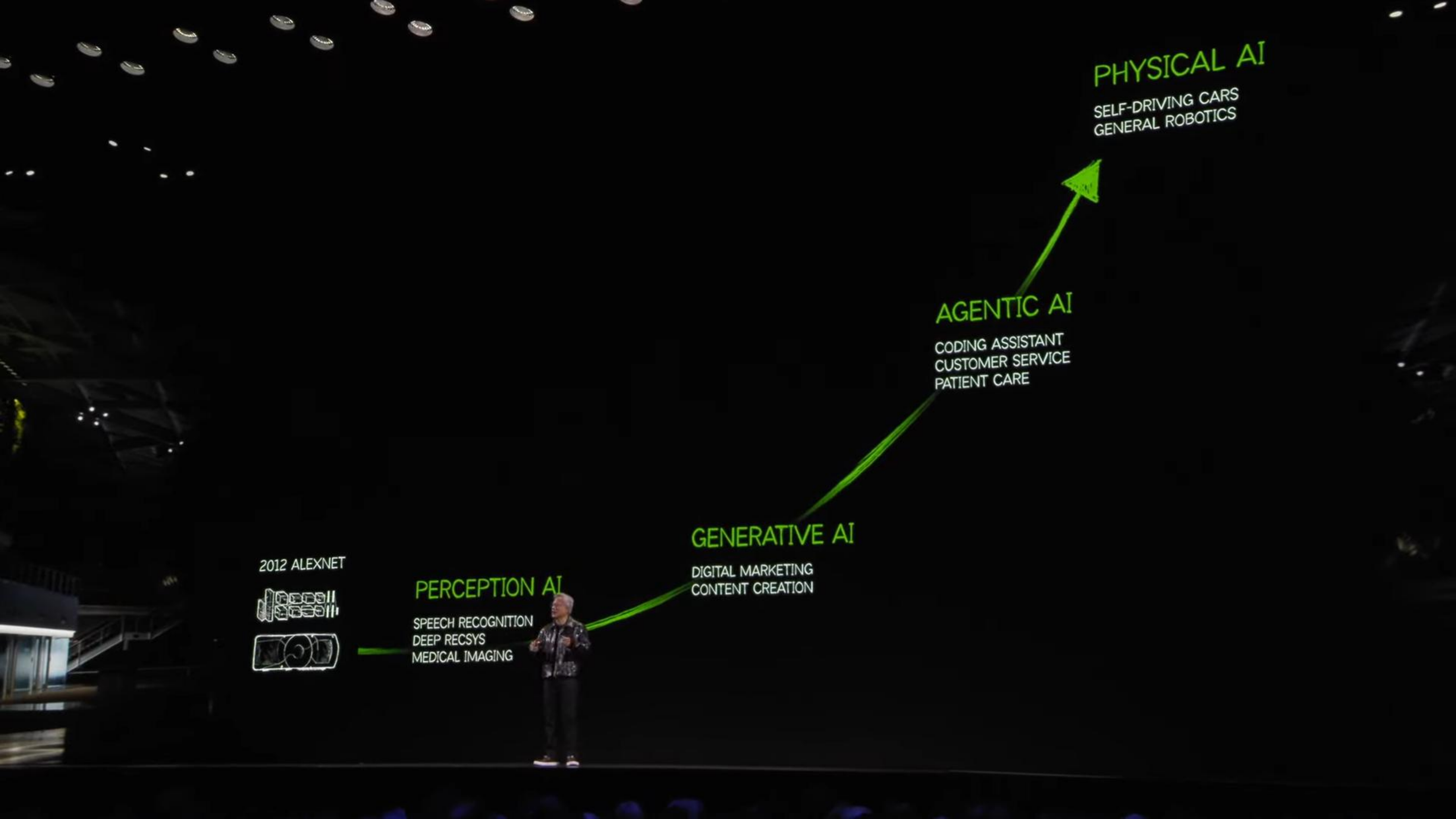
Surasuk Oakkharaamonphong

Microsoft MVP AI Platform & Developer Technologies
Technical Coach at Arise & INFINITAS by Krungthai



Agentic AI with AutoGen

A framework for building AI agents and applications



2012 ALEXNET

PERCEPTION AI

SPEECH RECOGNITION
DEEP RECSYS
MEDICAL IMAGING

GENERATIVE AI

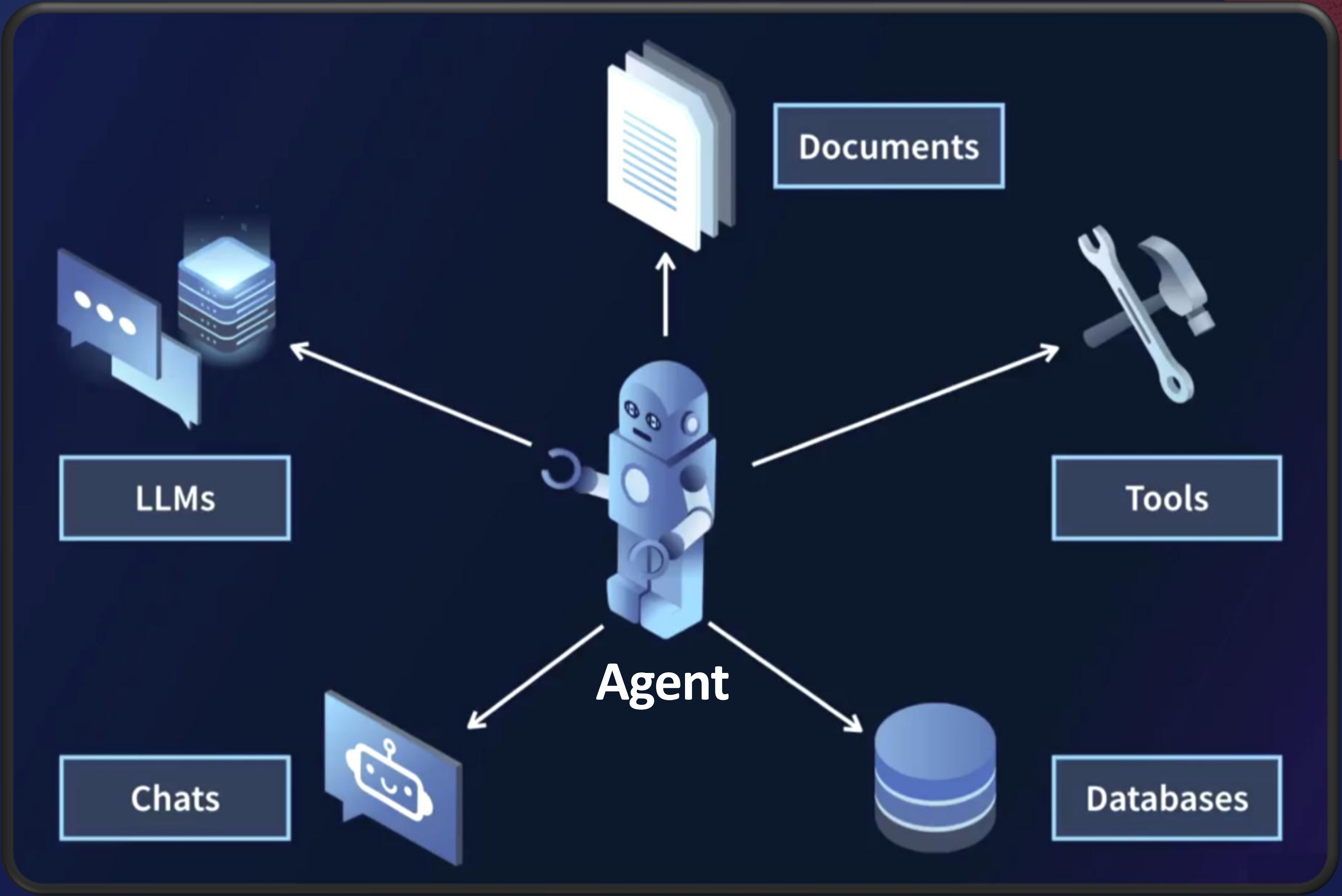
DIGITAL MARKETING
CONTENT CREATION

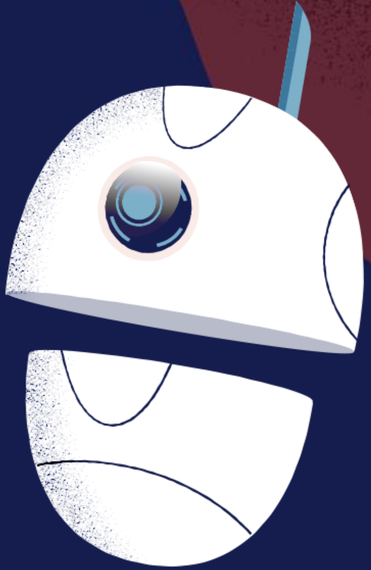
AGENTIC AI

CODING ASSISTANT
CUSTOMER SERVICE
PATIENT CARE

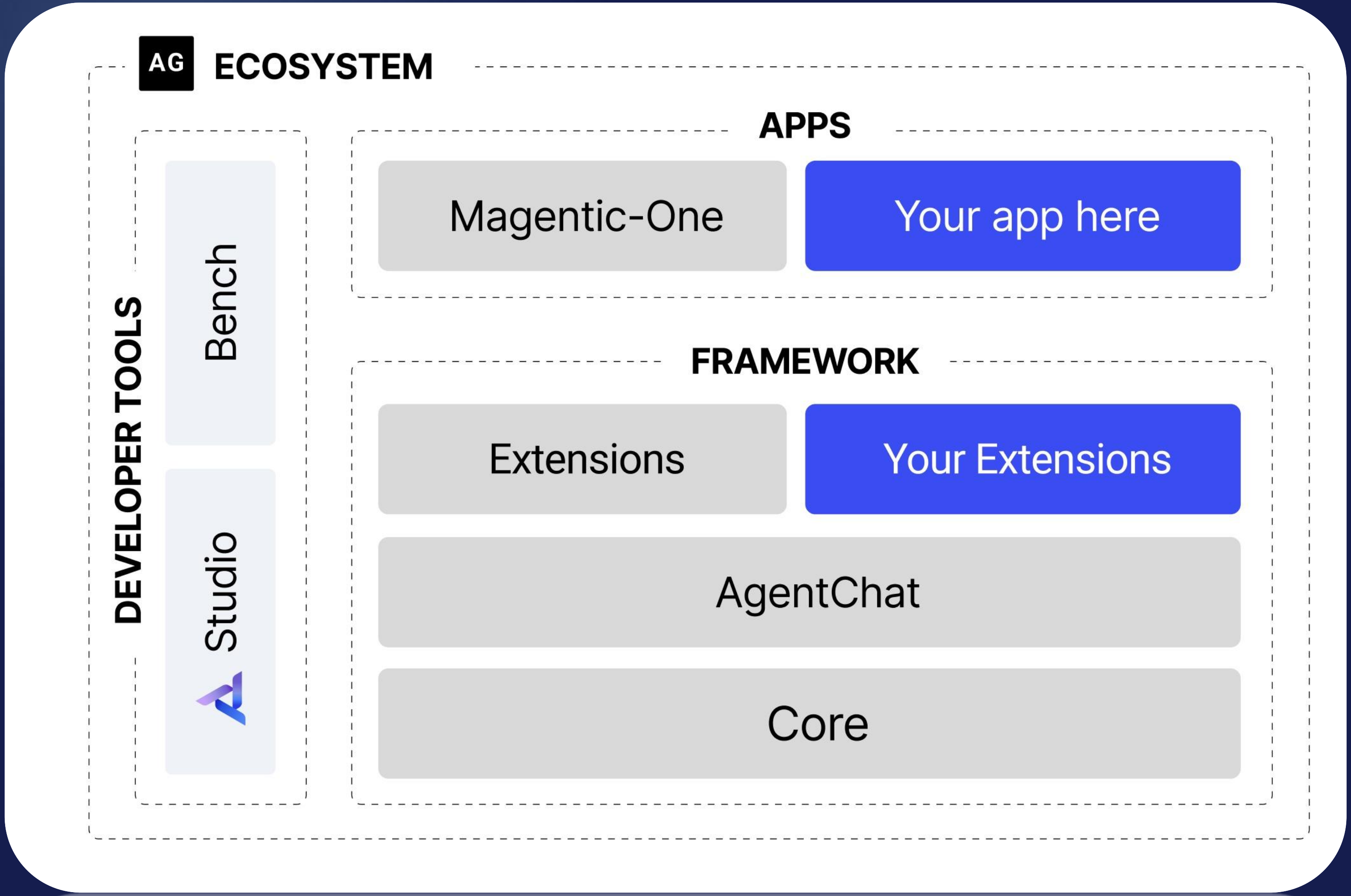
PHYSICAL AI

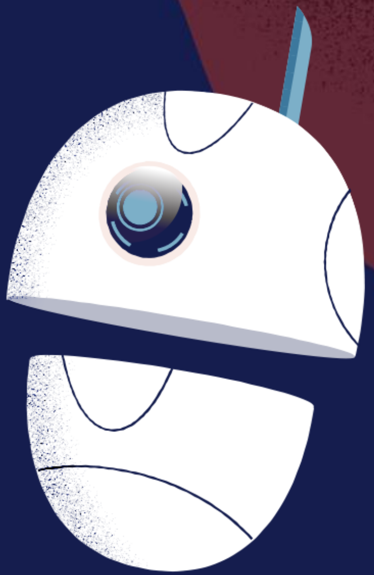
SELF-DRIVING CARS
GENERAL ROBOTICS


Documents
LLMs
Tools
Agent
Chats
Databases



The AutoGen ecosystem





global AI community

#GlobalAIBootcamp

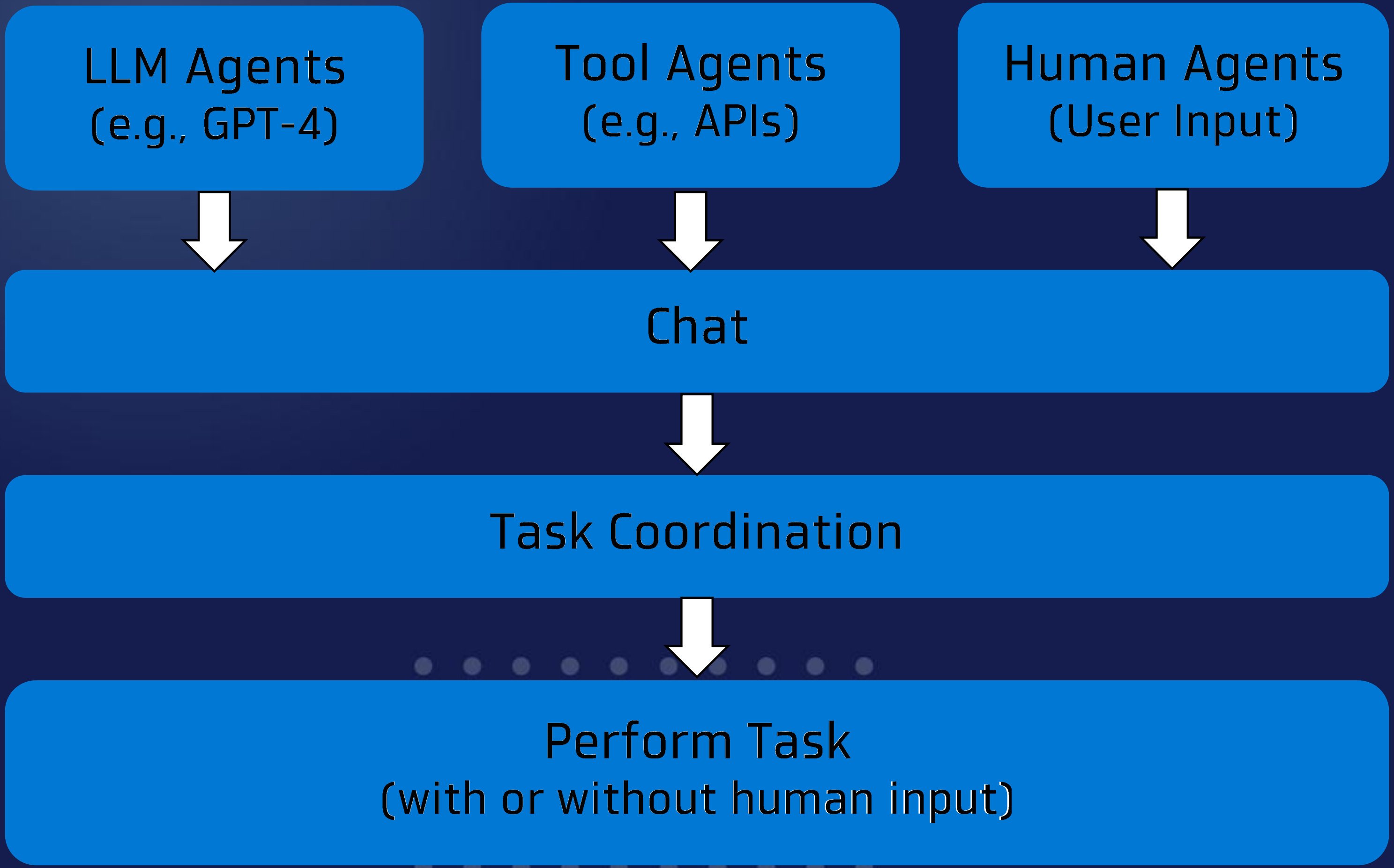
Developer tools

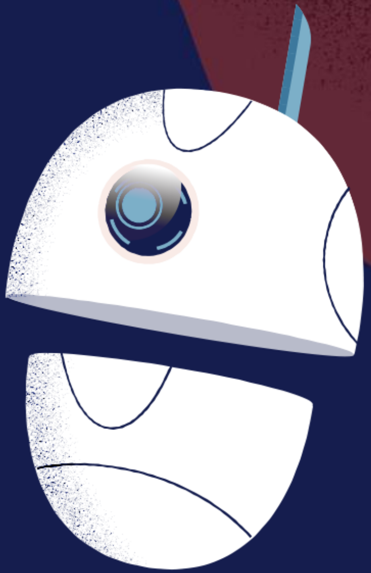
- AutoGen Studio - provides a no-code GUI for building multi-agent applications.
- AutoGen Bench - provides a benchmarking suite for evaluating agent performance.

AutoGen framework

- Core API - Implements message passing, event-driven agents, and local and distributed runtime for flexibility and power. It also support cross-language support for .NET and Python.
- AgentChat API - This API is built on top of the Core API and supports familiar multi-agent patterns such as two-agent chat or group chats.
- Extensions API - Enables first- and third-party extensions continuously expanding framework capabilities. It support specific implementation of LLM clients (e.g., OpenAI, AzureOpenAI), and capabilities such as code execution.

AugoGen framework





Installation

- Python 3.10 or later is required.
  - python3 –m venv .venv
  - source .venv/bin/activate

Python Packages

Agent Chat

- autogen-agentchat

Extension: OpenAI Model Client

- autogen-ext[openai]

Extension: Azure OpenAI with AAD authentication

- autogen-ext[azure]

Environment Parameters

- OPENAI_API_KEY
- AZURE_OPENAI_API_KEY
- AZURE_OPENAI_ENDPOINT
- OPENAI_API_VERSION
- OLLAMA_BASE_URL

AutoGen AgentChat

- autogen_agenchat.agents
  - AssistantAgent - assistance with tool use
  - BaseChatAgent - base class for a chat agent
  - CodeExecutorAgent - executes code and returns the output
  - SocietyOfMindAgent - inner team of agents to generate responses
  - UserProxyAgent - represent a human user through an input function
- autogen_agentchat.ui
  - Console
  - UserInputManager

AutoGen AgentChat

- autogen_agentchat.teams
  - BaseGroupChat - base class for group chat teams.
  - MagenticOneGroupChat - participants managed by the MagenticOneOrchestrator.
  - RoundRobinGroupChat - participants taking turns in a round-robin to publish a message to all.
  - SelectorGroupChat - participants takes turn to publish a message to all,
  - Swarm - selects the next speaker based on handoff message only.
- autogen_agentchat.conditions
  - ExternalTermination
  - HandoffTermination
  - MaxMessageTermination
  - SourceMatchTermination
  - StopMessageTermination
  - TextMentionTermination
  - TimeoutTermination
  - TokenUsageTermination

AutoGen AgentChat

- autogen_agentchat.messages
  - BaseMessage
  - ChatMessage
  - HandoffMessage
  - MultiModalMessage
  - StopMessage
  - TextMessage
  - ToolCallSummaryMessage
  - AgentEvent
  - MemoryQueryEvent
  - ModelClientStreamingChunkEvent
  - ToolCallExecutionEvent
  - ToolCallRequestEvent
  - UserInputRequestedEvent

AutoGen Core

- autogen_core.models
  - AssistantMessage
  - ChatCompletionClient
  - ChatCompletionTokenLogprob
  - CreateResult
  - FunctionExecutionResult
  - FunctionExecutionResultMessage
  - ModelCapabilities
  - ModelFamily
  - ModelInfo
  - RequestUsage
  - SystemMessage
  - TopLogprob
  - UserMessage
- autogen_core.code_executor
  - Alias
  - CodeBlock
  - CodeExecutor
  - CodeResult
  - FunctionWithRequirements
  - FunctionWithRequirementsStr
  - ImportFromModule
  - with_requirements

AutoGen Extensions

- autogen_ext.models.openai
  - OpenAIChatCompletionClient
  - AzureOpenAIChatCompletionClient
  - BaseOpenAIChatCompletionClient
  - AzureOpenAIClientConfigurationConfgModel
  - OpenAIClientConfigurationConfigModel
  - BaseOpenAIClientConfigurationConfigModel
  - CreateArgumentsConfigModel
- autogen_ext.code_executors.local
  - LocalCommandLineCodeExecutor
- autogen_ext.code_executors.docker
  - DockerCommandLineCodeExecutor

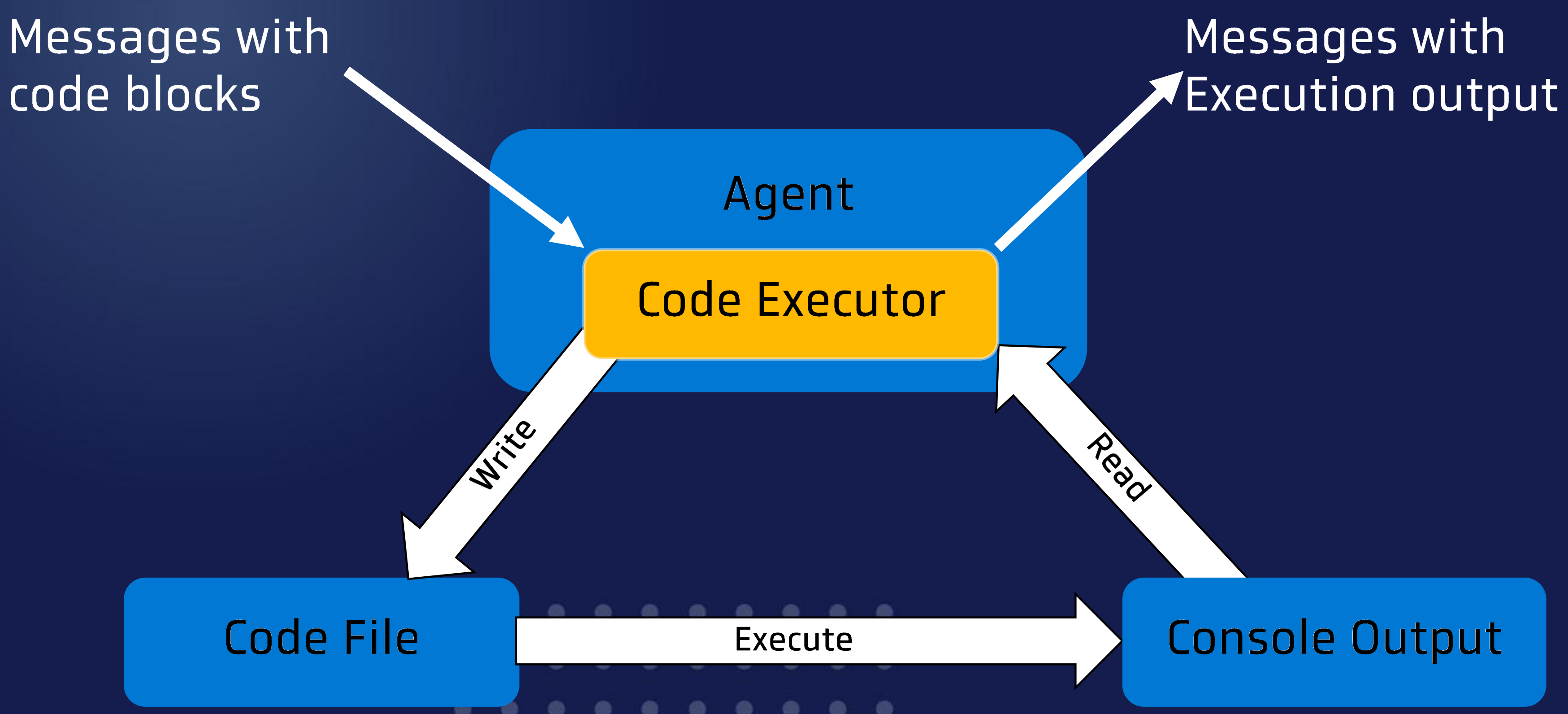
Code Executors

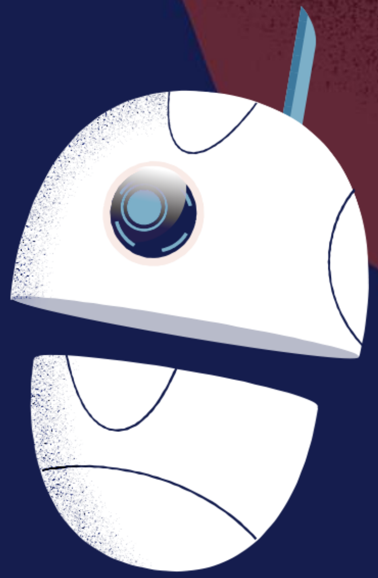
- Code executor processes messages containing code:
  - Executes the code
  - Returns results

- There are two built in:
  - Command Line Code Executor: runs code in cmd env (Unix shell)
  - Jupyter Executor runs in a Jupyter env.

- AutoGen can execute code:
  - Locally - runs code directly on the host system (development)
  - Docker container - runs code in an isolated container

Code Executors – Local

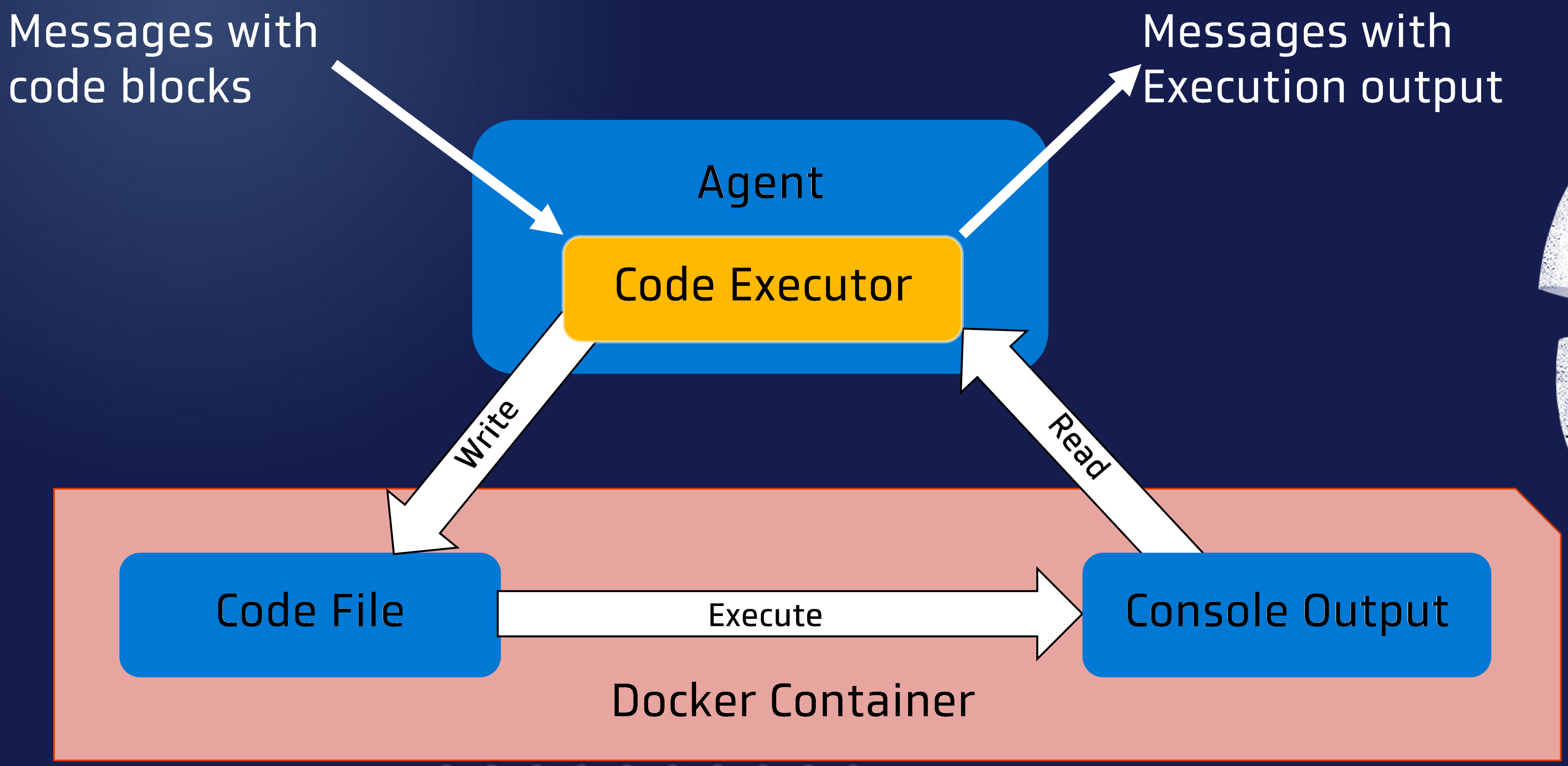




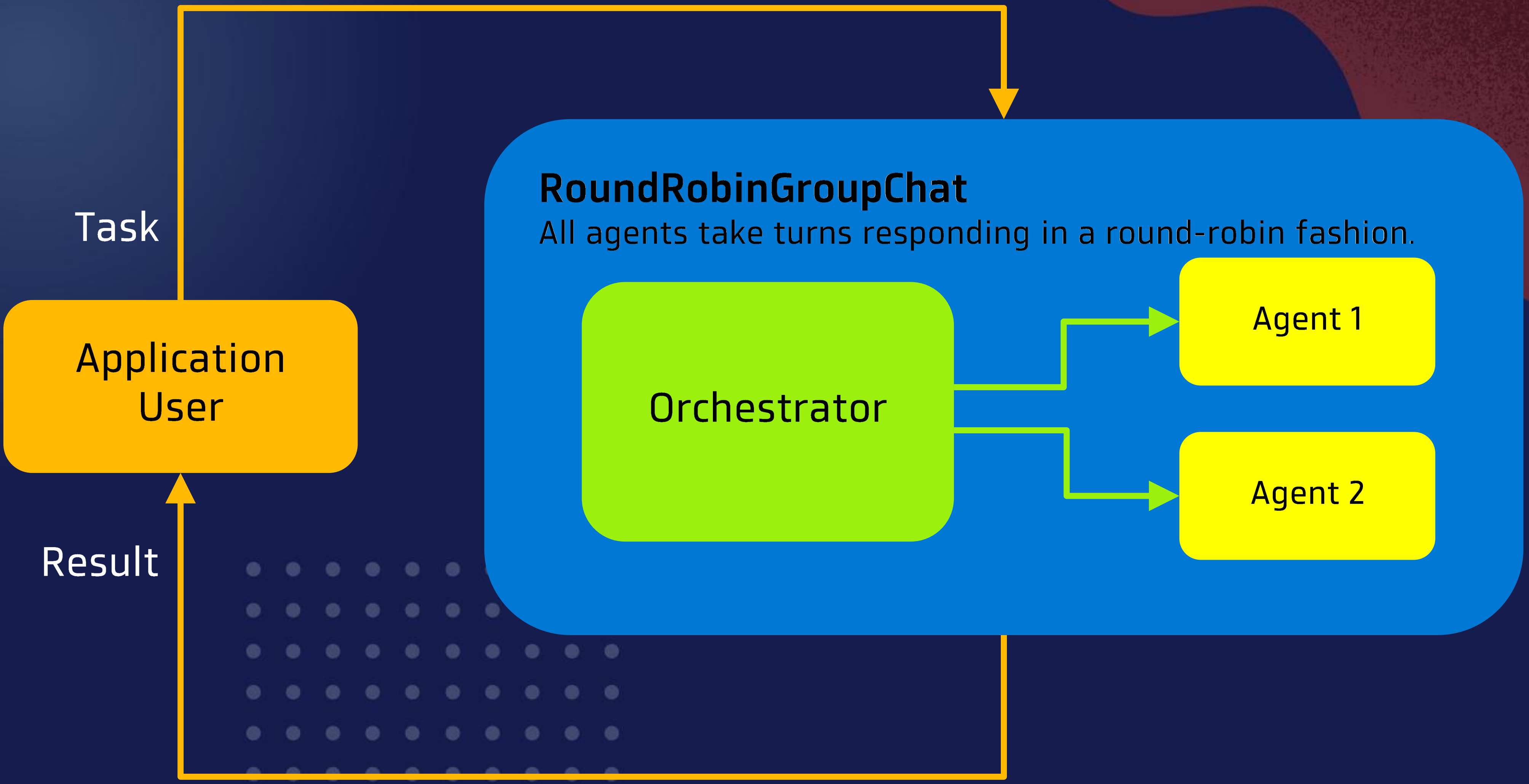
global AI community

#GlobalAIBootcamp

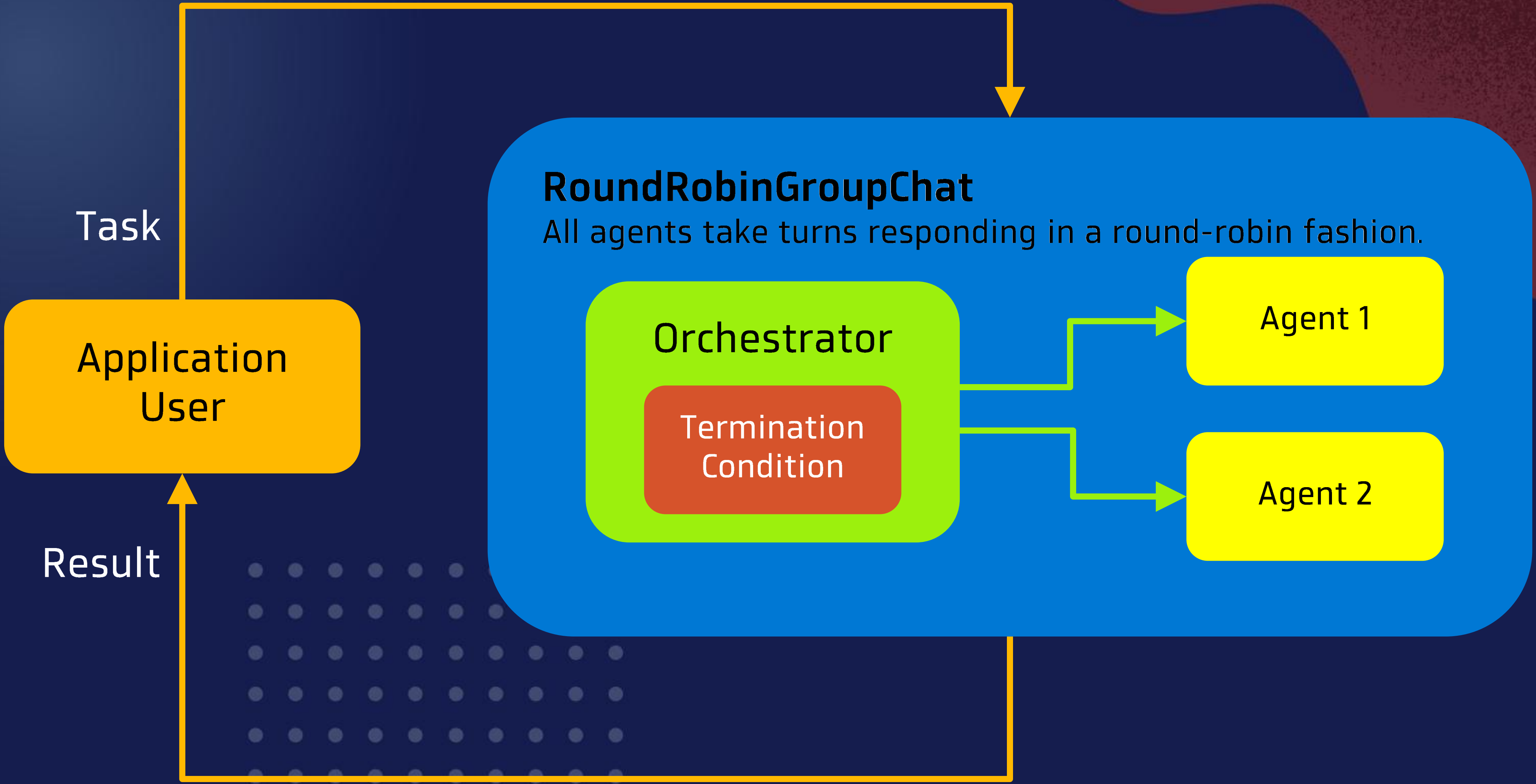
Code Executors – Docker



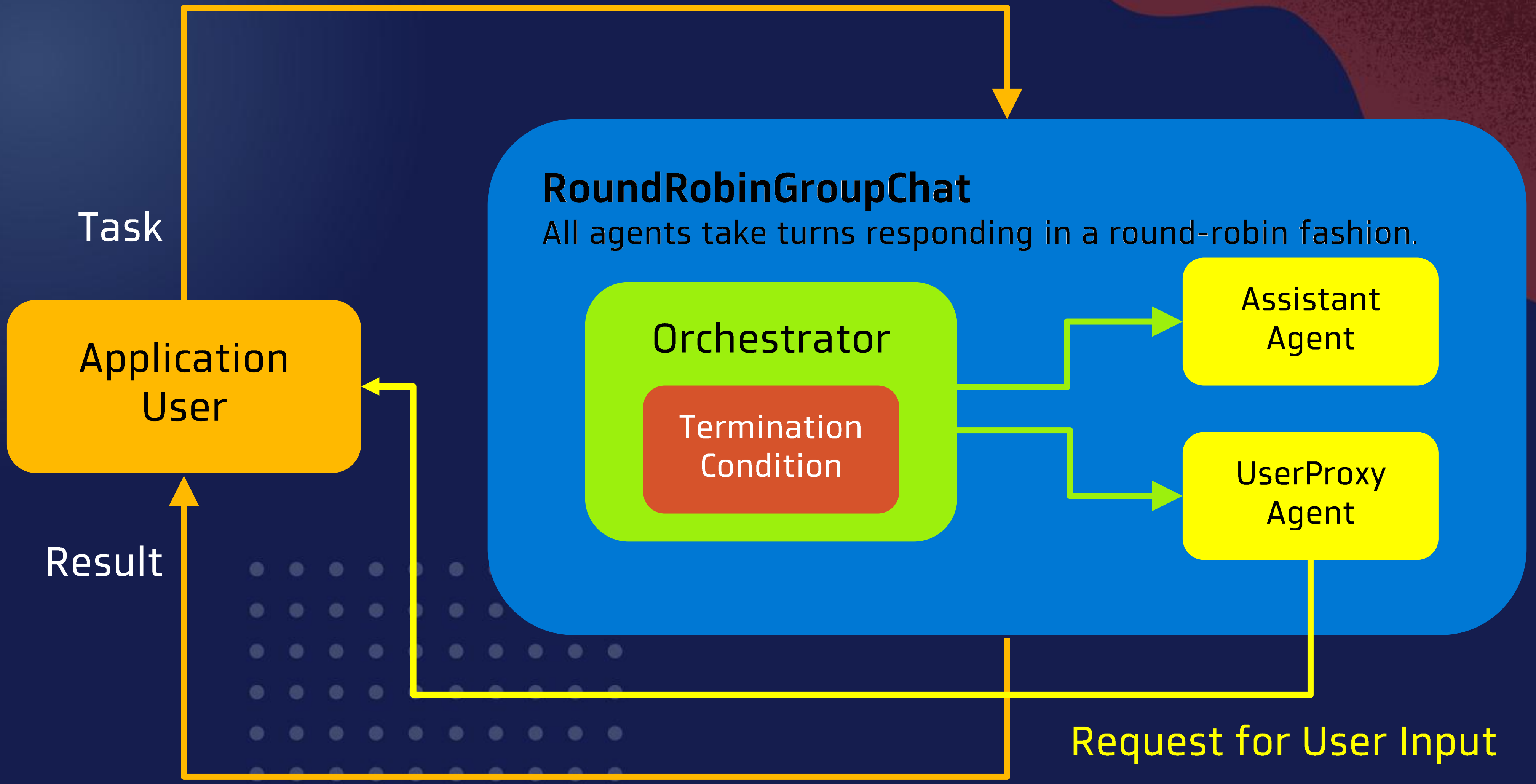
Team is a group of agents that work together



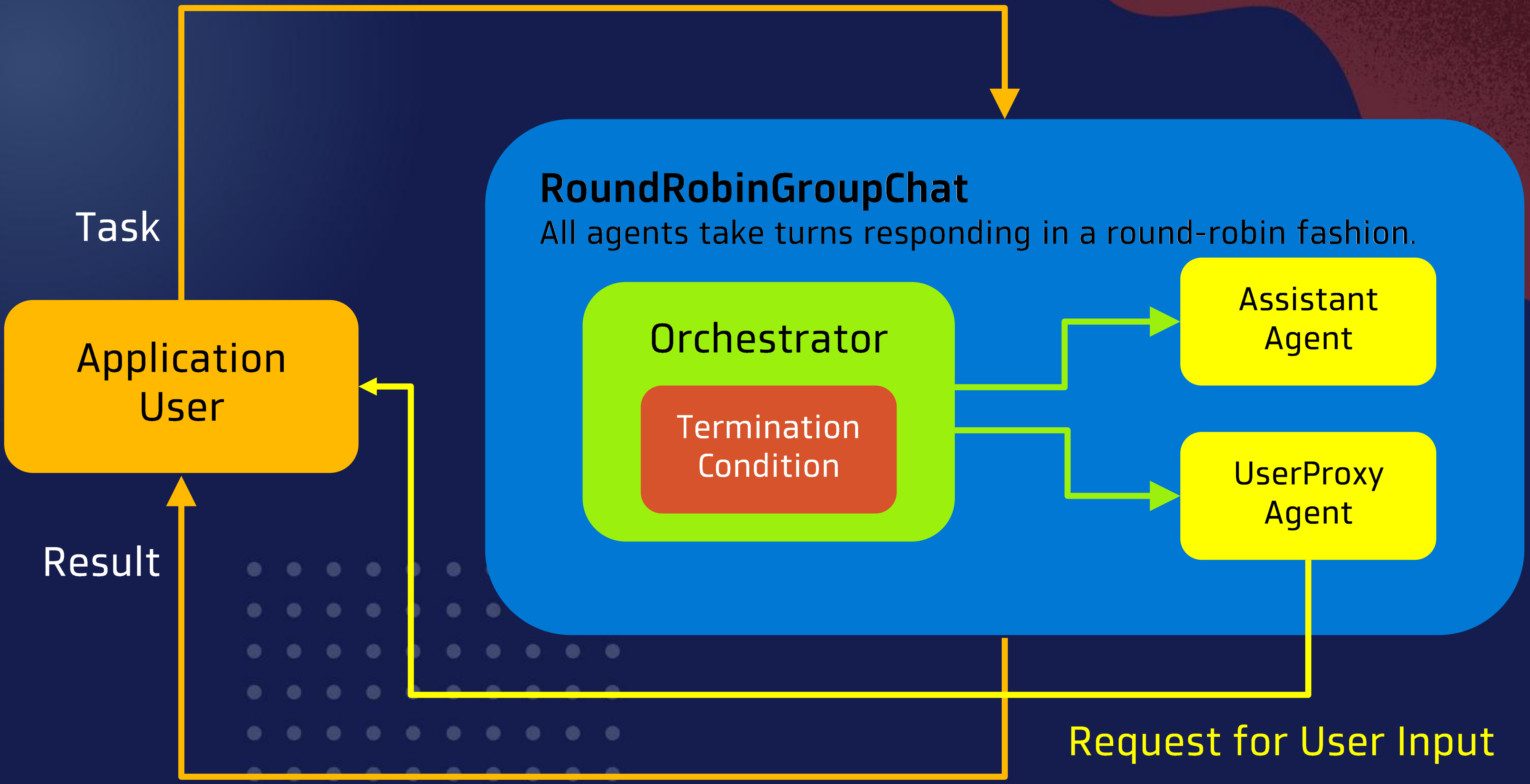
Termination Condition is called after each agent responds



Human-in-the-Loop providing feedback during a run



Termination Condition is called after each agent responds



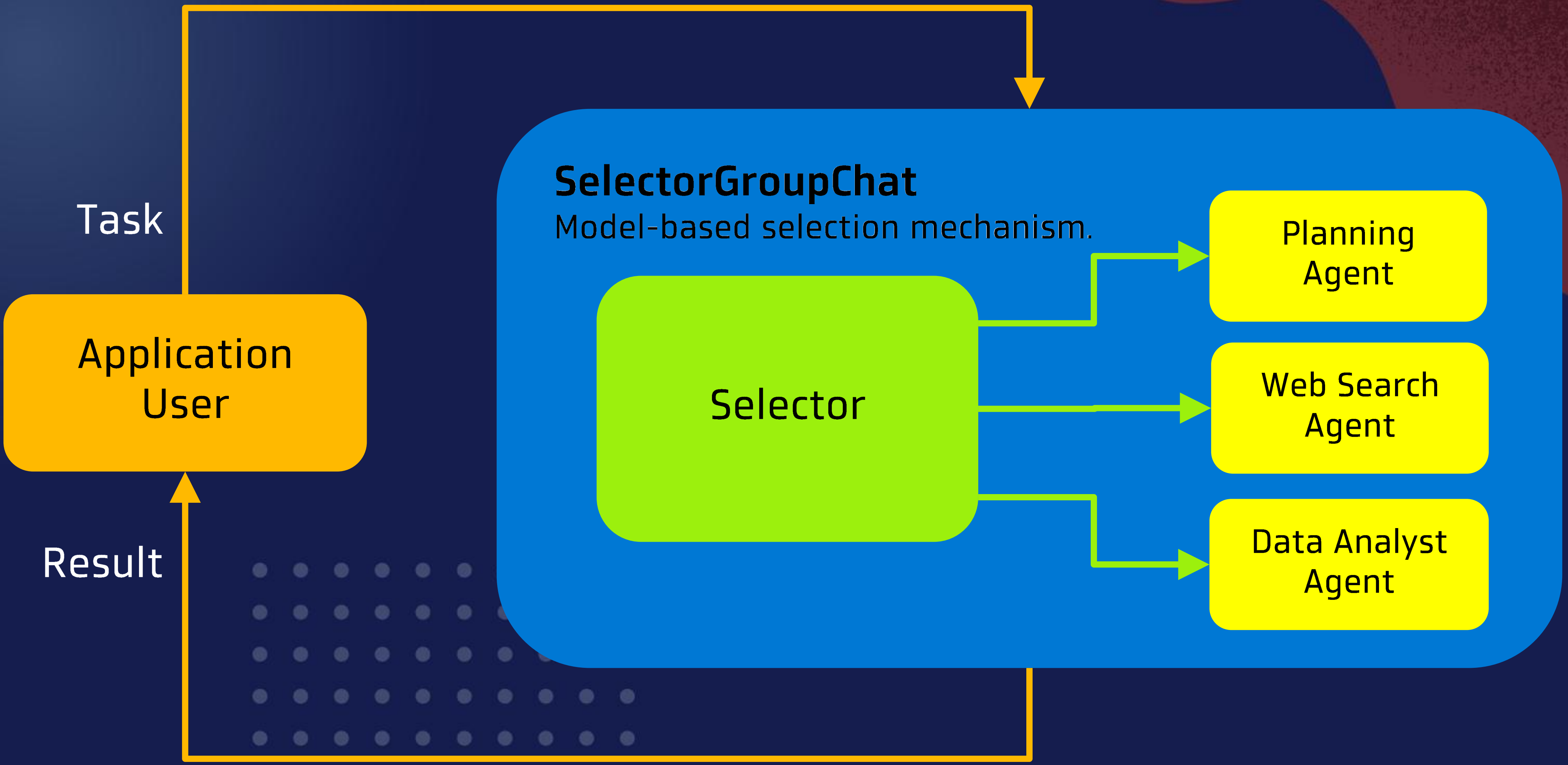
Termination Condition

1. MaxMessageTermination: Stops after a specified number of messages have been produced, including both agent and task messages.
2. TextMentionTermination: Stops when specific text or string is mentioned in a message (e.g., "TERMINATE").
3. TokenUsageTermination: Stops when a certain number of prompt or completion tokens are used. This requires the agents to report token usage in their messages.
4. TimeoutTermination: Stops after a specified duration in seconds.

Termination Condition

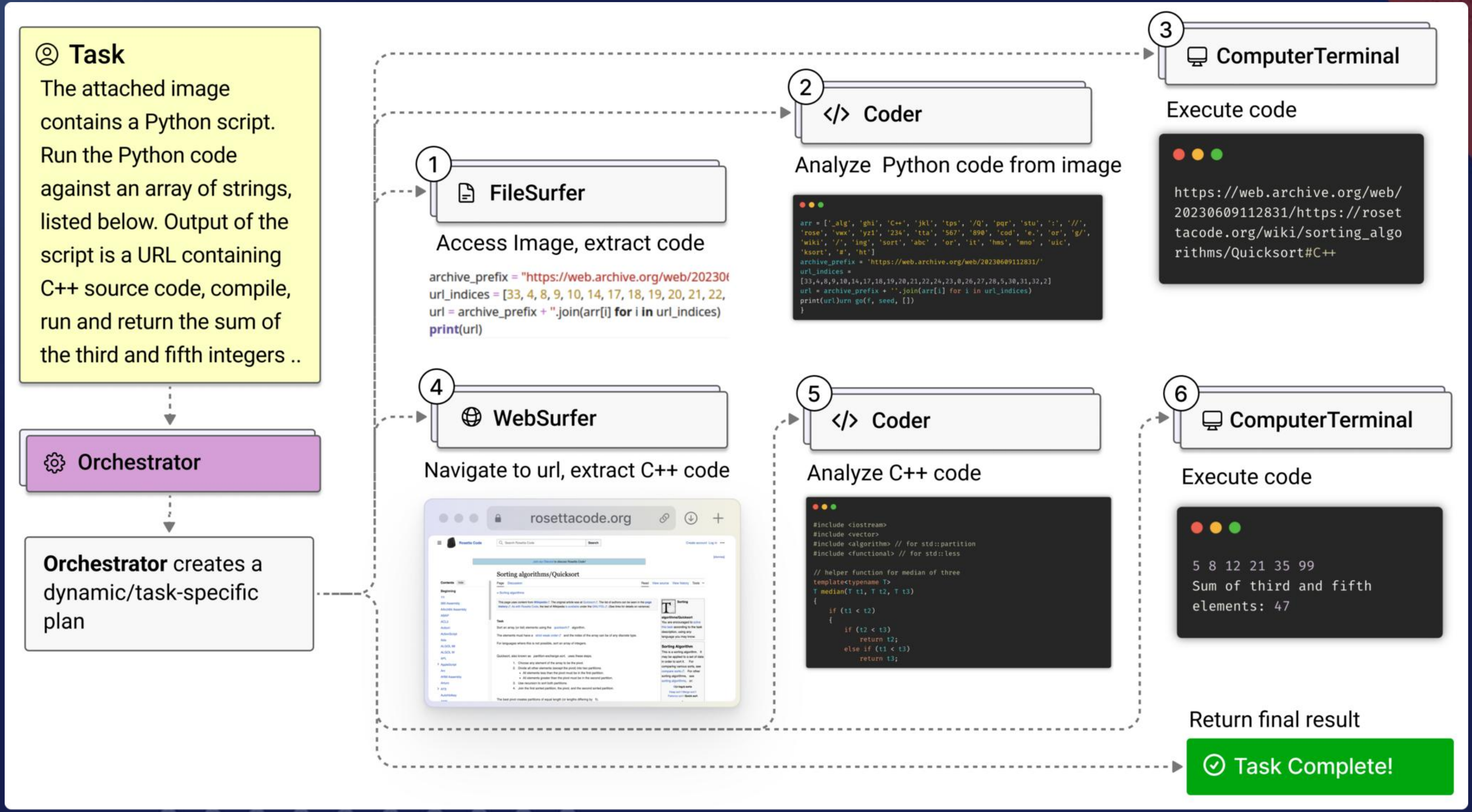
5. HandoffTermination: Stops when a handoff to a specific target is requested. Handoff messages can be used to build patterns such as Swarm. This is useful when you want to pause the run and allow application or user to provide input when an agent hands off to them.
6. SourceMatchTermination: Stops after a specific agent responds.
7. ExternalTermination: Enables programmatic control of termination from outside the run. This is useful for UI integration (e.g., “Stop” buttons in chat interfaces).
8. StopMessageTermination: Stops when a StopMessage is produced by an agent.

Team is a group of agents that work together



Magentic-One

A Generalist Multi-Agent System for Solving Complex Tasks


Task
The attached image contains a Python script. Run the Python code against an array of strings, listed below. Output of the script is a URL containing C++ source code, compile, run and return the sum of the third and fifth integers ..
Orchestrator
Orchestrator creates a dynamic/task-specific plan
1
FileSurfer
Access Image, extract code
2
Coder
Analyze Python code from image
3
ComputerTerminal
Execute code
4
WebSurfer
Navigate to url, extract C++ code
rosettacode.org
5
Coder
Analyze C++ code
6
ComputerTerminal
Execute code
Return final result
Task Complete!

```
archive_prefix = "https://web.archive.org/web/20230
url_indices = [33, 4, 8, 9, 10, 14, 17, 18, 19, 20, 21, 22,
url = archive_prefix + ''.join(arr[i] for i in url_indices)
print(url)
```

```
arr = ['_alg', 'ghi', 'C++', 'jkl', 'tps', '/Q', 'pqr', 'stu', ':', '//',
'rose', 'vwx', 'yz1', '234', 'tta', '567', '890', 'cod', 'e.', 'or', 'g/',
'wiki', '/', 'ing', 'sort', 'abc' , 'or', 'it', 'hms', 'mno' , 'uic',
'ksort', '#', 'ht']
archive_prefix = 'https://web.archive.org/web/20230609112831/'
url_indices =
[33,4,8,9,10,14,17,18,19,20,21,22,24,23,0,26,27,28,5,30,31,32,2]
url = archive_prefix + ''.join(arr[i] for i in url_indices)
print(url)urn go(f, seed, [])
}
```

```
https://web.archive.org/web/
20230609112831/https://roset
tacode.org/wiki/sorting_algo
rithms/Quicksort#C++
```

```
#include <iostream>
#include <vector>
#include <algorithm> // for std::partition
#include <functional> // for std::less

// helper function for median of three
template<typename T>
T median(T t1, T t2, T t3)
{
    if (t1 < t2)
    {
        if (t2 < t3)
            return t2;
        else if (t1 < t3)
            return t3;
```

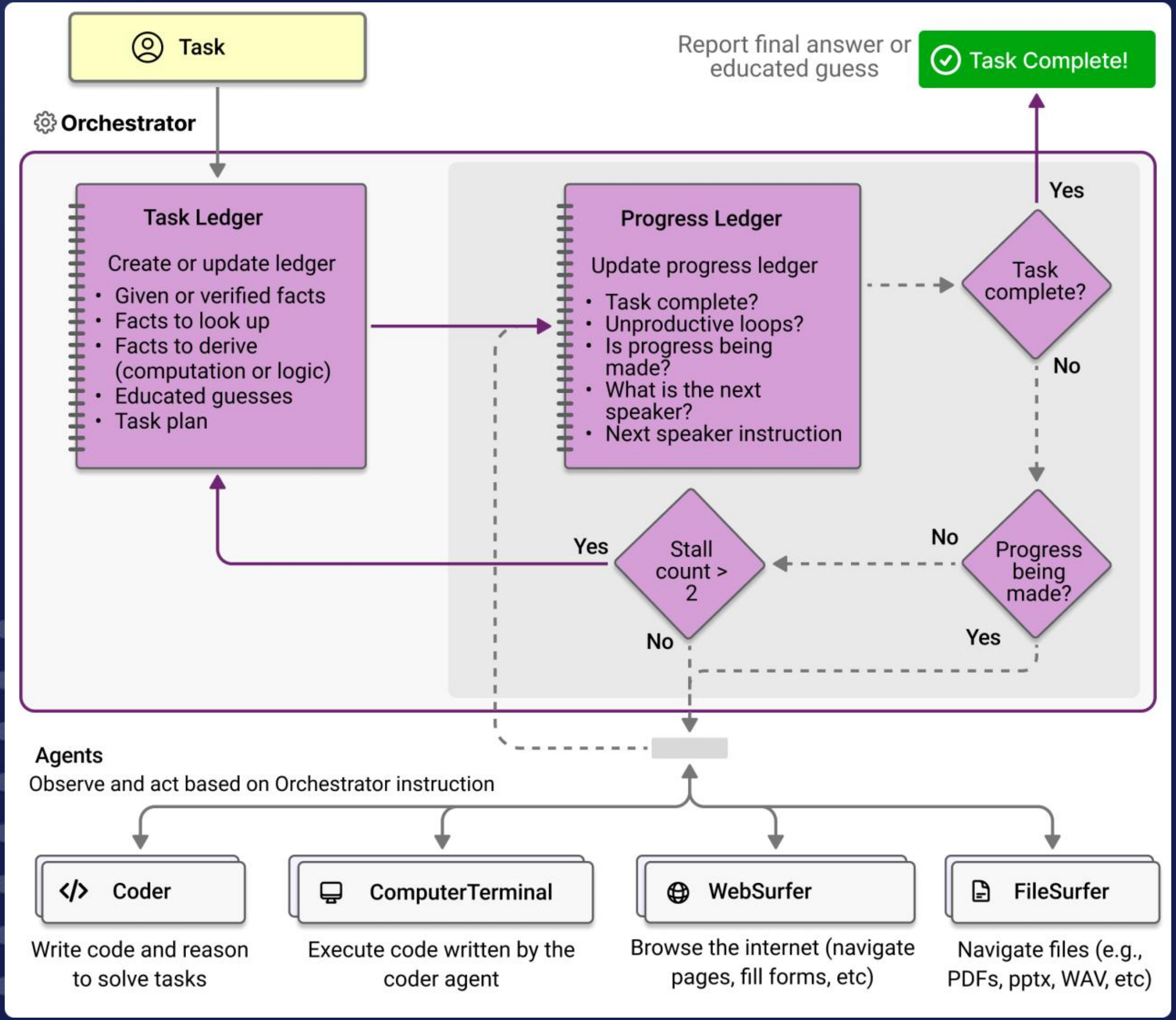
```
5 8 12 21 35 99
Sum of third and fifth
elements: 47
```

global AI community

#GlobalAIBootcamp

Magentic-One

A Generalist Multi-Agent System for Solving Complex Tasks



Task
Orchestrator
Report final answer or educated guess
Task Complete!
Task Ledger
Create or update ledger
• Given or verified facts
• Facts to look up
• Facts to derive (computation or logic)
• Educated guesses
• Task plan
Progress Ledger
Update progress ledger
• Task complete?
• Unproductive loops?
• Is progress being made?
• What is the next speaker?
• Next speaker instruction
Task complete?
Yes
No
Progress being made?
No
Yes
Stall count > 2
Yes
No
Agents
Observe and act based on Orchestrator instruction
Coder
Write code and reason to solve tasks
ComputerTerminal
Execute code written by the coder agent
WebSurfer
Browse the internet (navigate pages, fill forms, etc)
FileSurfer
Navigate files (e.g., PDFs, pptx, WAV, etc)

Travel Planning Example

global AI community

#GlobalAIBootcamp

Company Research Example

global AI community

#GlobalAIBootcamp

Literature Review Example

global AI community

#GlobalAIBootcamp

Agentic AI with AutoGen

"Programming Framework for Agentic AI"

Surasuk Oakkharaamonphong

Arise by INFINITAS

INFINITAS by Krungthai

POST-TEST

ASTER SHARING: AGENTIC AI WITH AUTOGEN

Arise Learning
Learning.px.group@arise.tech

#GlobalAIBootcamp

แบบประเมินผลหลังการเข้าร่วม

ASTER SHARING: AGENTIC AI WITH AUTOGEN
วันที่ 13 กุมภาพันธ์ 2568

Arise Learning
Learning.px.group@arise.tech

#GlobalAIBootcamp