Agents

Introduction

Agents

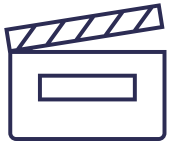
What is an Agent?

Perceives Environment

Takes Actions autonomously

To achieve a defined goal

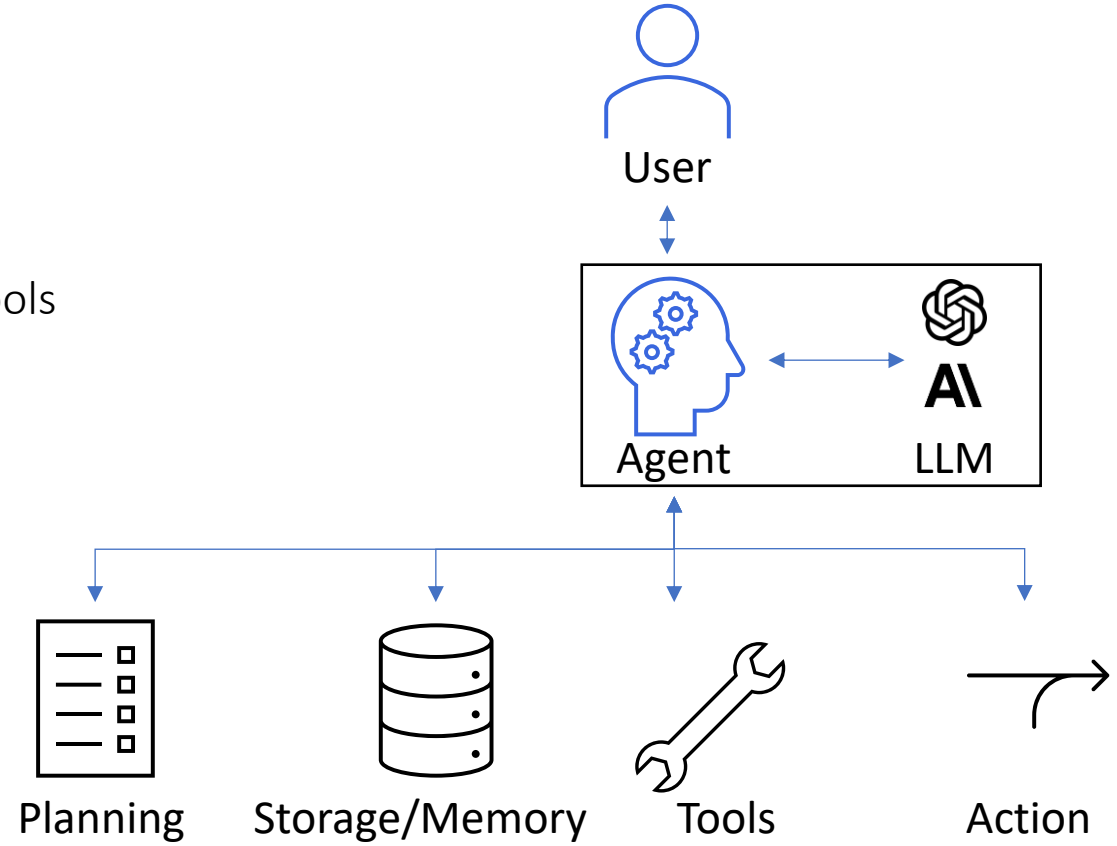
Improve through learning



Agents

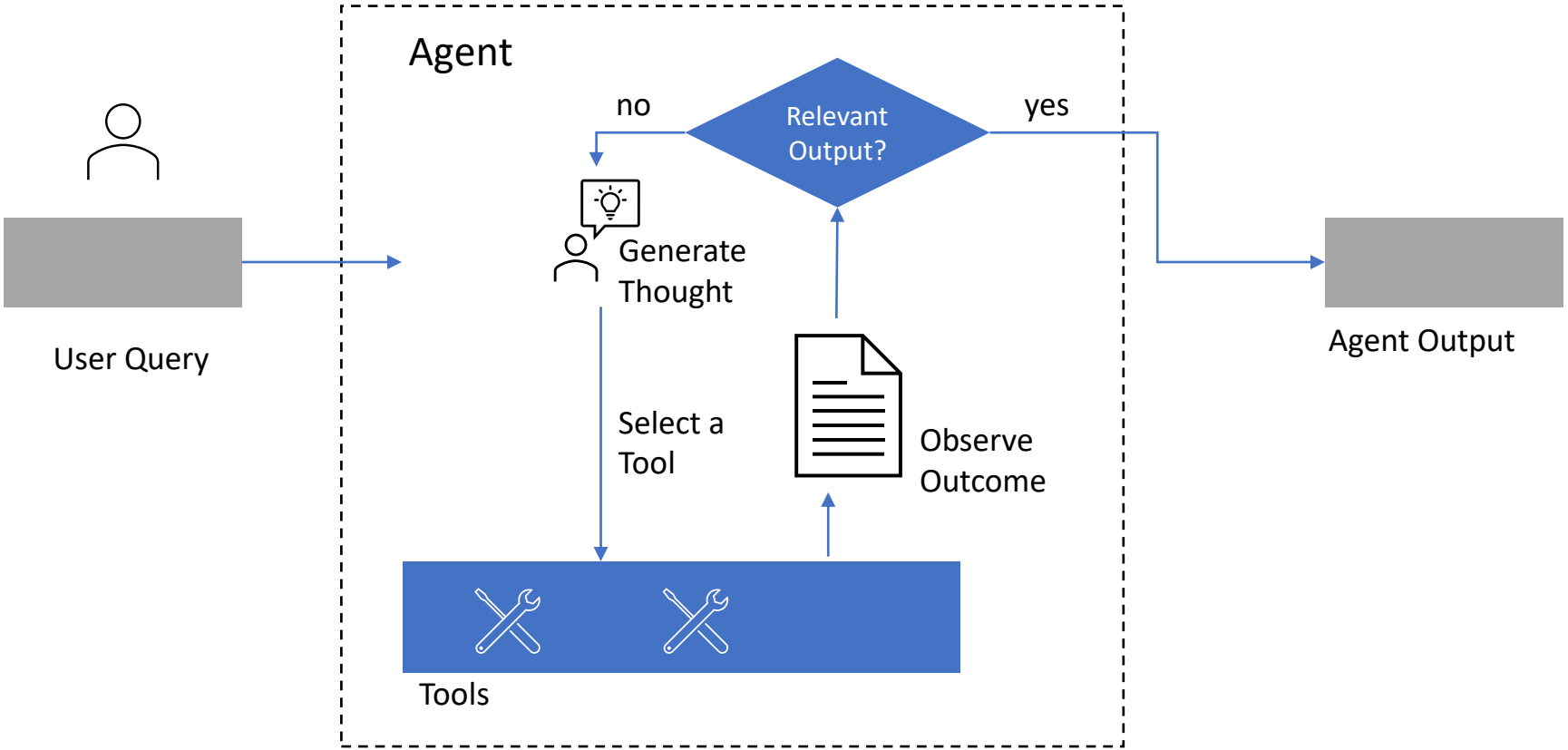
What is an Agent?

- LLM apps execute tasks
- core element: agent
- uses planning, memory, and tools
- can perform actions
- is an expert in its field



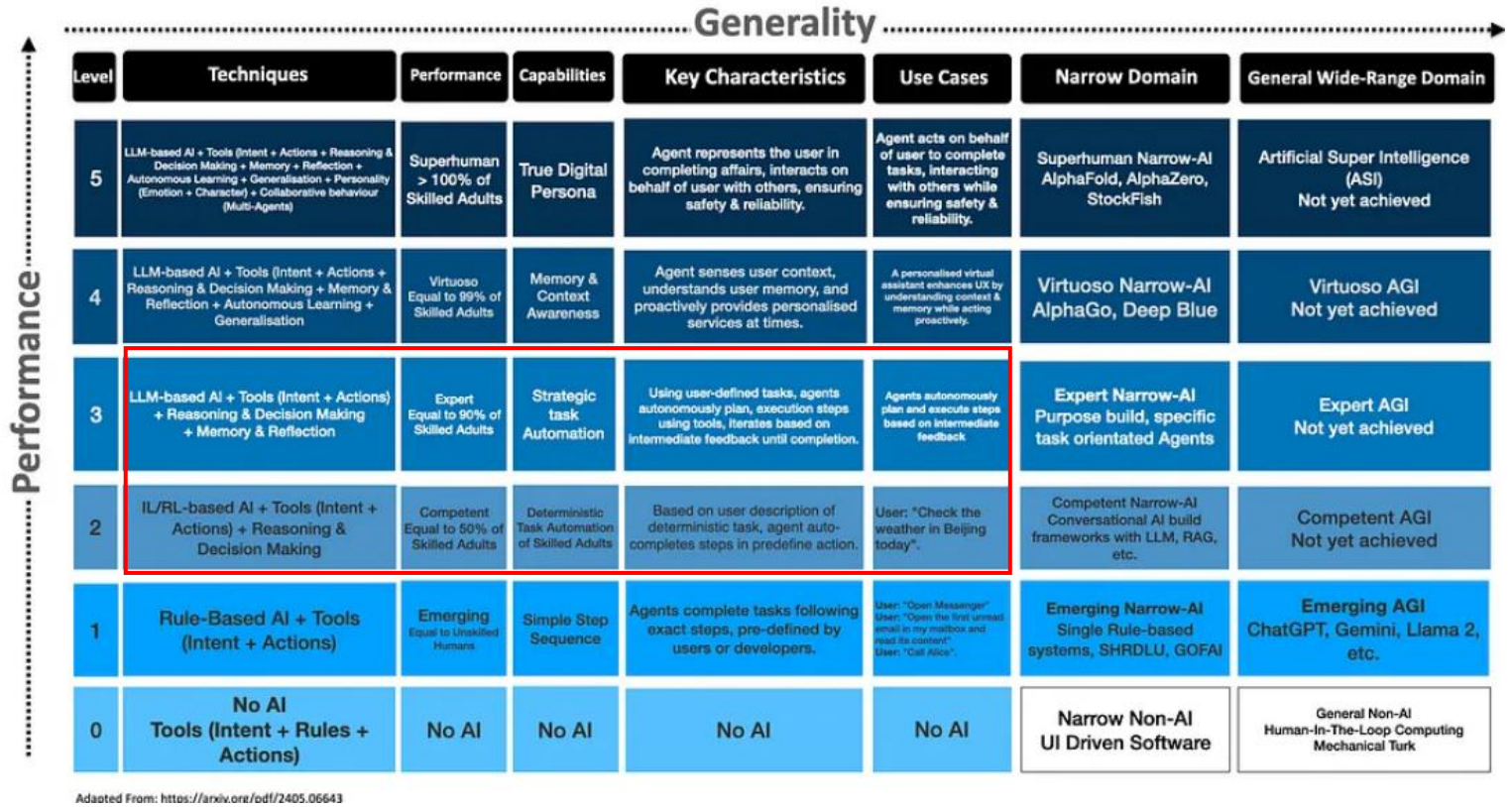
Agents

How does an Agent work?



Agents

Levels of AI Agents



Source: https://cobusgreyling.medium.com/5-levels-of-ai-agents-updated-0ddf8931a1c6

Coding Agents

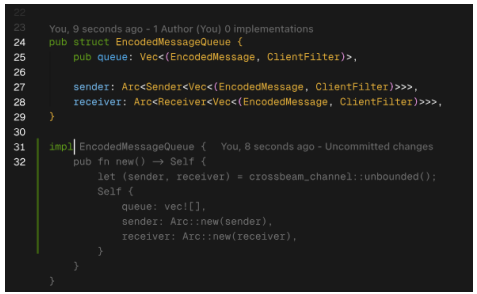
Es ist nicht die Aufgabe der Menschen gewesen, eine Programmiersprache zu lernen. Stattdessen sollten Computer nützlich sein, auch wenn man keine Programmiersprache beherrsche. Man wird allerdings lernen müssen, von KI-Software mithilfe richtig formulierter Anfragen – sogenannter Prompts – das gewünschte Ergebnis zu bekommen.

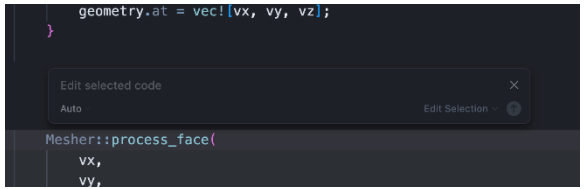
Jen-Hsun Huang, CEO von Nvidia

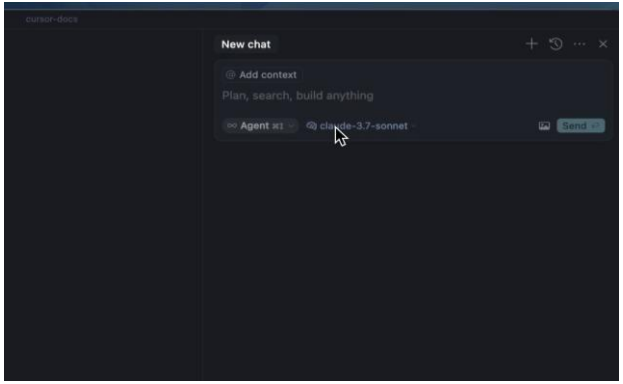
Agents

Coding Agents: Evolution

Auto Completion → **Inline Complete** → **Complete code base**







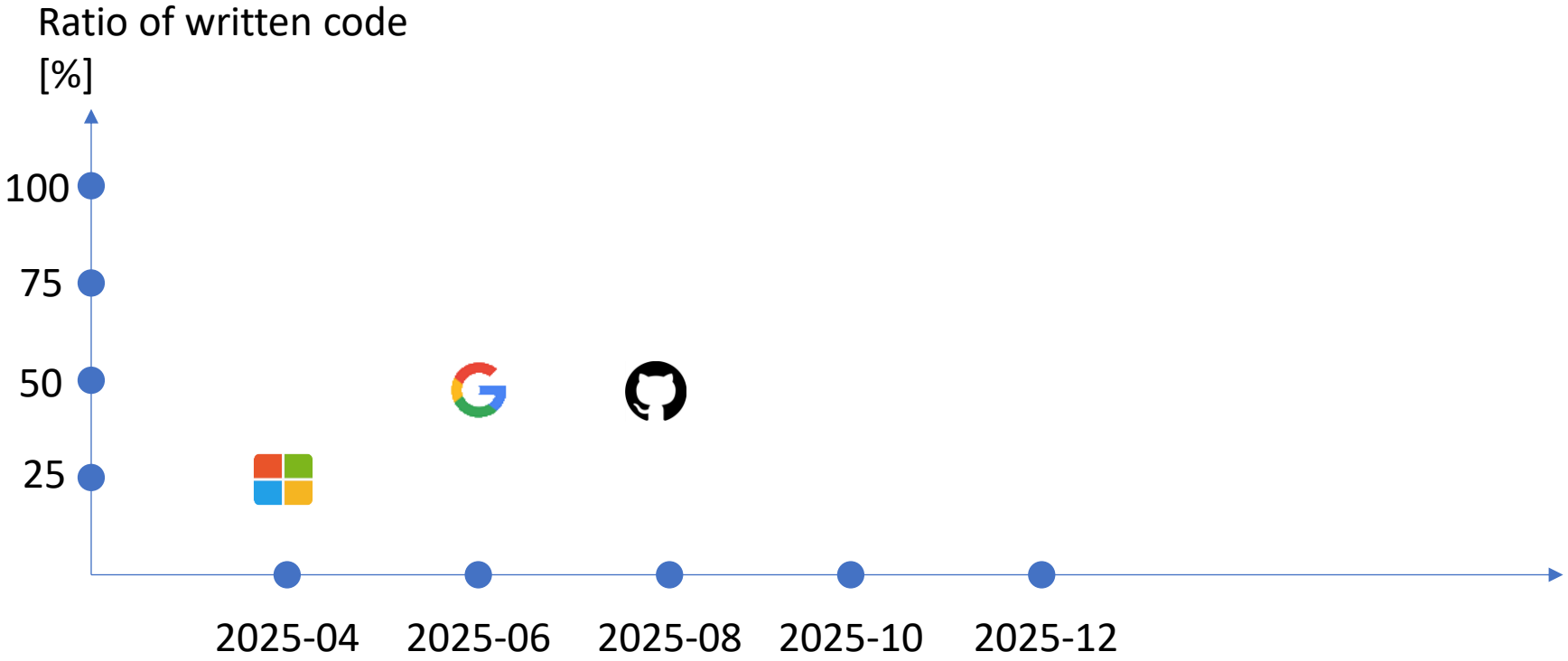
- Proposes automatically next code tokens

- Edits multiple lines
- Available in VS Code, Cursor, Windsurf

- Accesses complete code base
- Performs edits

Agents

Coding Agents: How much code is written by AI?



Agents

Coding Agents: Browser-Based

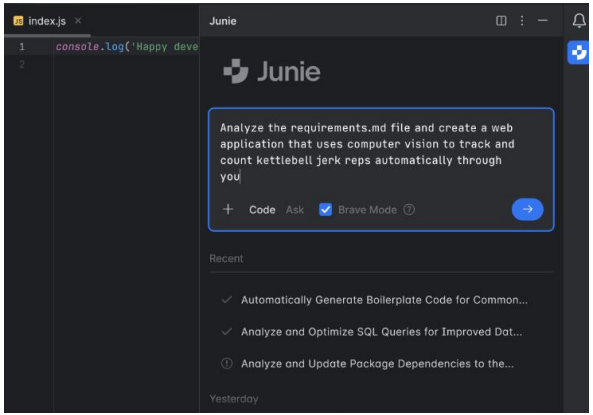
- Browser-based
- freemium
- https://replit.com/

- Browser-based
- freemium
- https://lovable.dev/

Agents

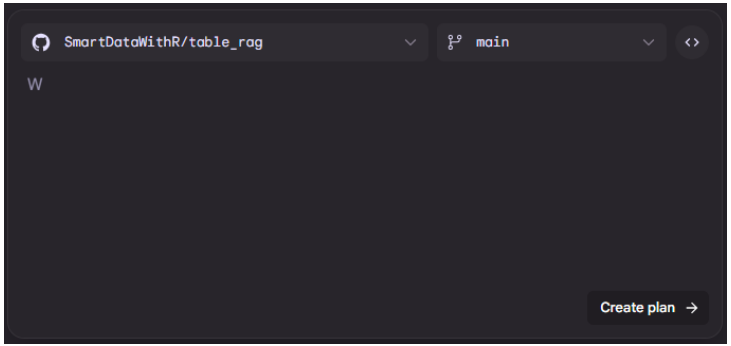
Coding Agents: Browser-Based

Junie



- Developed by Jetbrain
- Available in Jetbrain-IDEs (e.g. PyCharm)

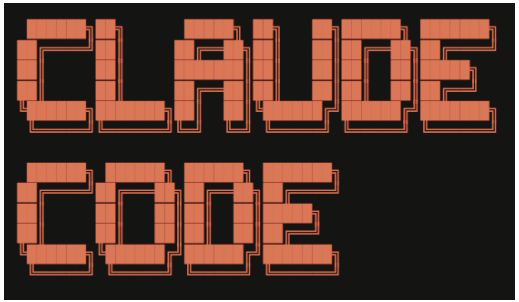
Jules



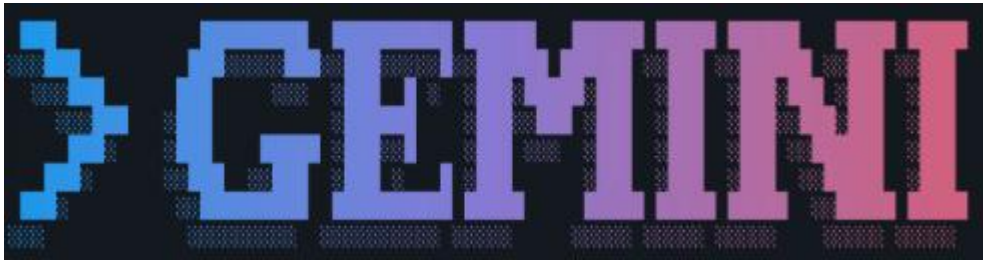
- Browser-based
- Developed by Google
- Integrates with GitHub

Agents

Coding Agents: Terminal



- Works from command line
- Can work with your complete code base
- Install: npm install -g @anthropic-ai/claude-code
- Available for claude subscribers or API-use



- Source
- Can work with your complete code base

- Install: npm install -g @google/gemini-cli
- Available via API key or Google SignIn
- 100 Gemini 2.5 Pro requests free per day!

Agents

Coding Agents: Terminal

- Works from command line
- Can work with your complete code base
- Install: npm install -g @anthropic-ai/claude-code
- Available for claude subscribers or API-use

- Source
- Can work with your complete code base

- Install: npm install -g @google/gemini-cli
- Available via API key or Google SignIn
- 100 Gemini 2.5 Pro requests free per day!

Agents

Coding Agents: Terminal

Codex

- Developed by OpenAI
- Included in ChatGPT Plus, Pro, Business, Edu, Enterprise, or via API-key
- Works from command line

Agents

Coding Agents: IDE

Cursor

- 20$/month
- Suitable for
  - Experienced devs, large codebase
- Makes Claude 4 available

Windsurf

- 15$/month
- Suitable for
  - Beginner, vibe-coding, personal projects
- Develop web apps quick and without configuration effort

Agents

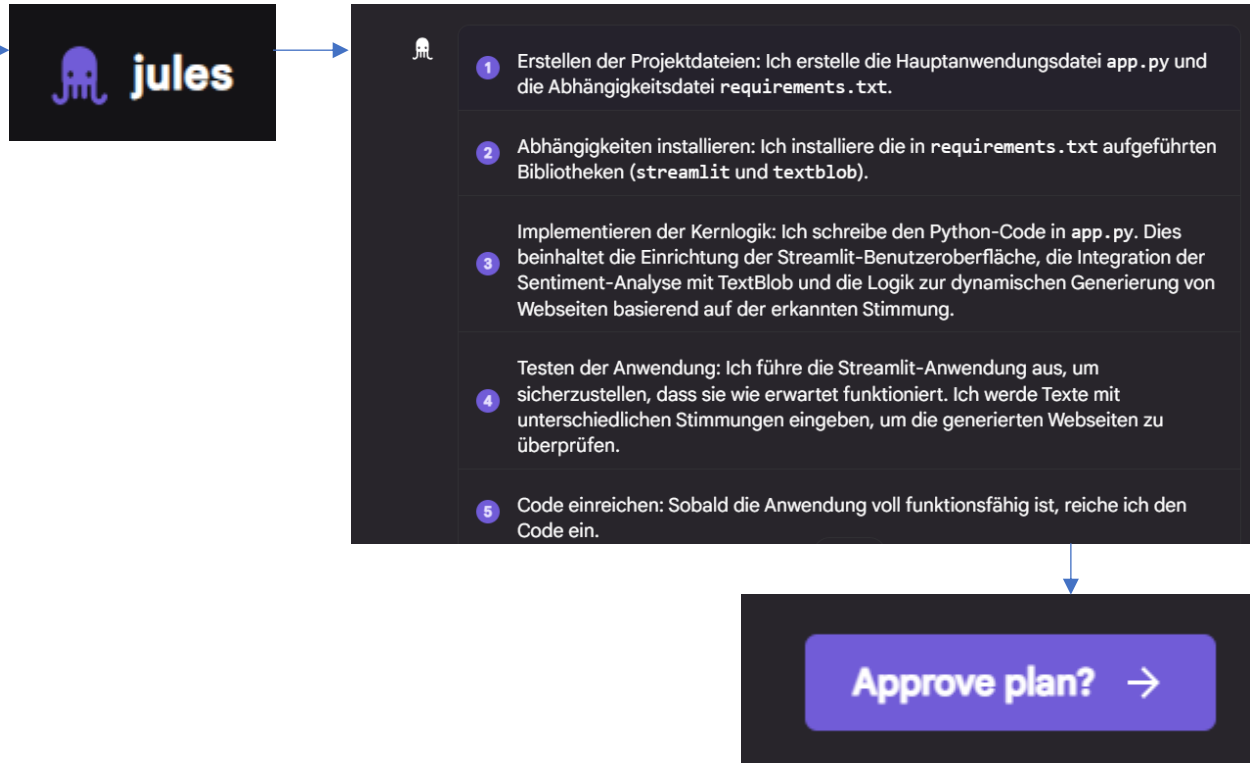
Coding Agents: IDE

Agents

Coding Agents: Exercise

User Prompt

Diese Anwendung liest einen kurzen Text ein und bewertet dessen emotionale Stimmung (z. B. positiv, negativ, neutral) mithilfe von Sentiment-Analyse. Basierend auf dem ermittelten Vibe generiert die App eine Webseite. Beispielsweise könnte ein positiver Text eine helle, freundliche Webseite mit vielen leuchtenden Farben und optimistischen Sprüchen erzeugen, während ein negativer Text eine düstere, minimalistische Seite mit gedämpften Farben und melancholischen Zitaten generiert. Der Code wird in Python geschrieben und streamlit für die App verwendet.



Agents

Coding Agents: Exercise

jules

app.py requirements.txt

/app.py

```python
import streamlit as st
from textblob import TextBlob

def generate_website(sentiment):
    """Generiert eine Webseite basierend auf der Stimmung."""
    if sentiment == "positiv":
        st.markdown("""
        <style>
        .container {
            background-color: #ffecb3;
            color: #333;
            padding: 20px;
            border-radius: 10px;
            text-align: center;
            font-family: 'Arial', sans-serif;
        }
        h1 {
            color: #ff8f00;
        }
        </style>
        <div class="container">
            <h1>Ein sonniger Tag! ☀️</h1>
            <p>Die Stimmung ist wunderbar positiv. Hier ist ein Lächeln für dich! 😊</p>
            <p><i>"Behalte dein Gesicht immer im Sonnenschein - und Schatten werden hinter dich
        </div>
        """, unsafe_allow_html=True)
    elif sentiment == "negativ":
        st.markdown("""
        <style>
        .container {
            background-color: #424242;
            color: #f5f5f5;
            padding: 20px;
            border-radius: 10px;
            text-align: center;
            font-family: 'Georgia', serif;
        }
        h1 {
            color: #9e9e9e;
```

app.py requirements.txt

/requirements.txt

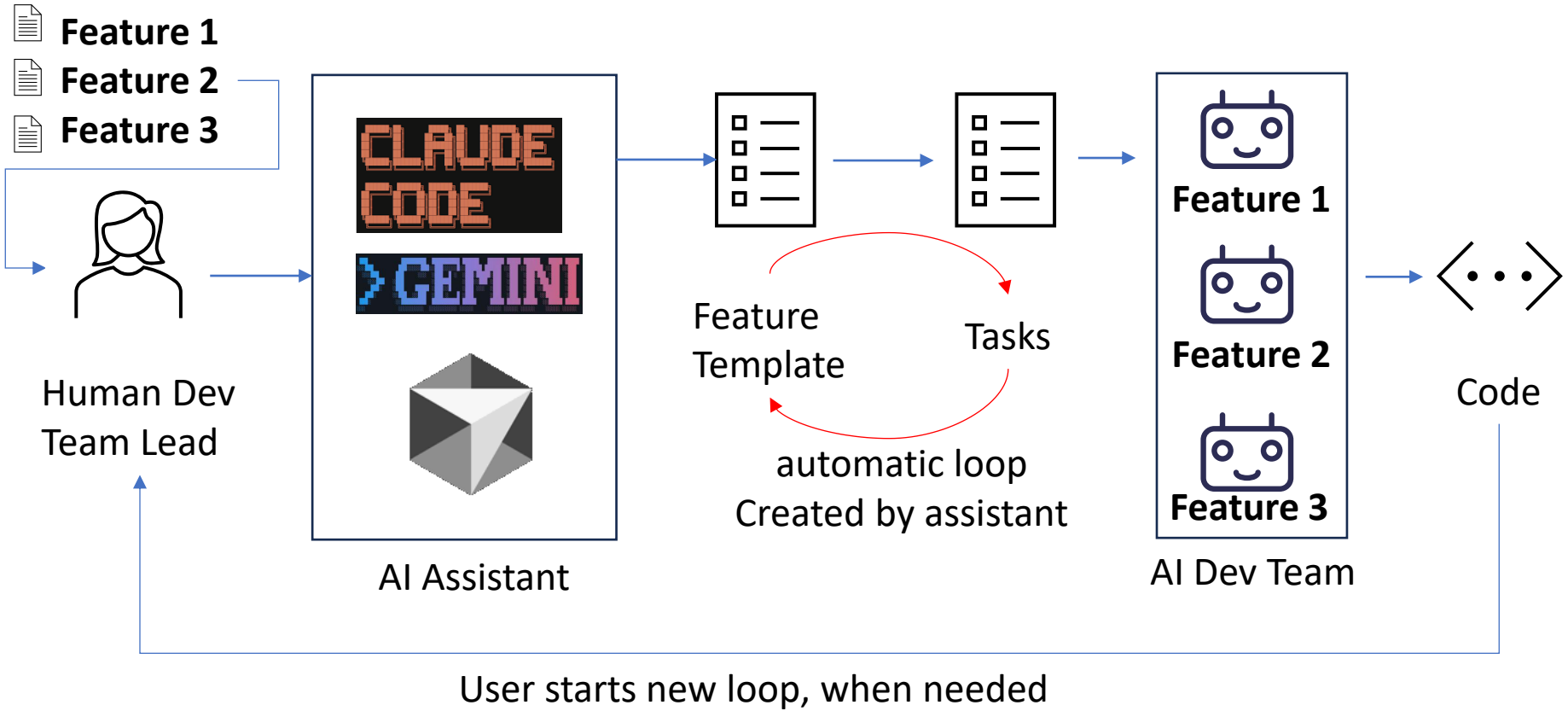
```
streamlit
textblob
```

Agents

Coding Agents: World before Agents

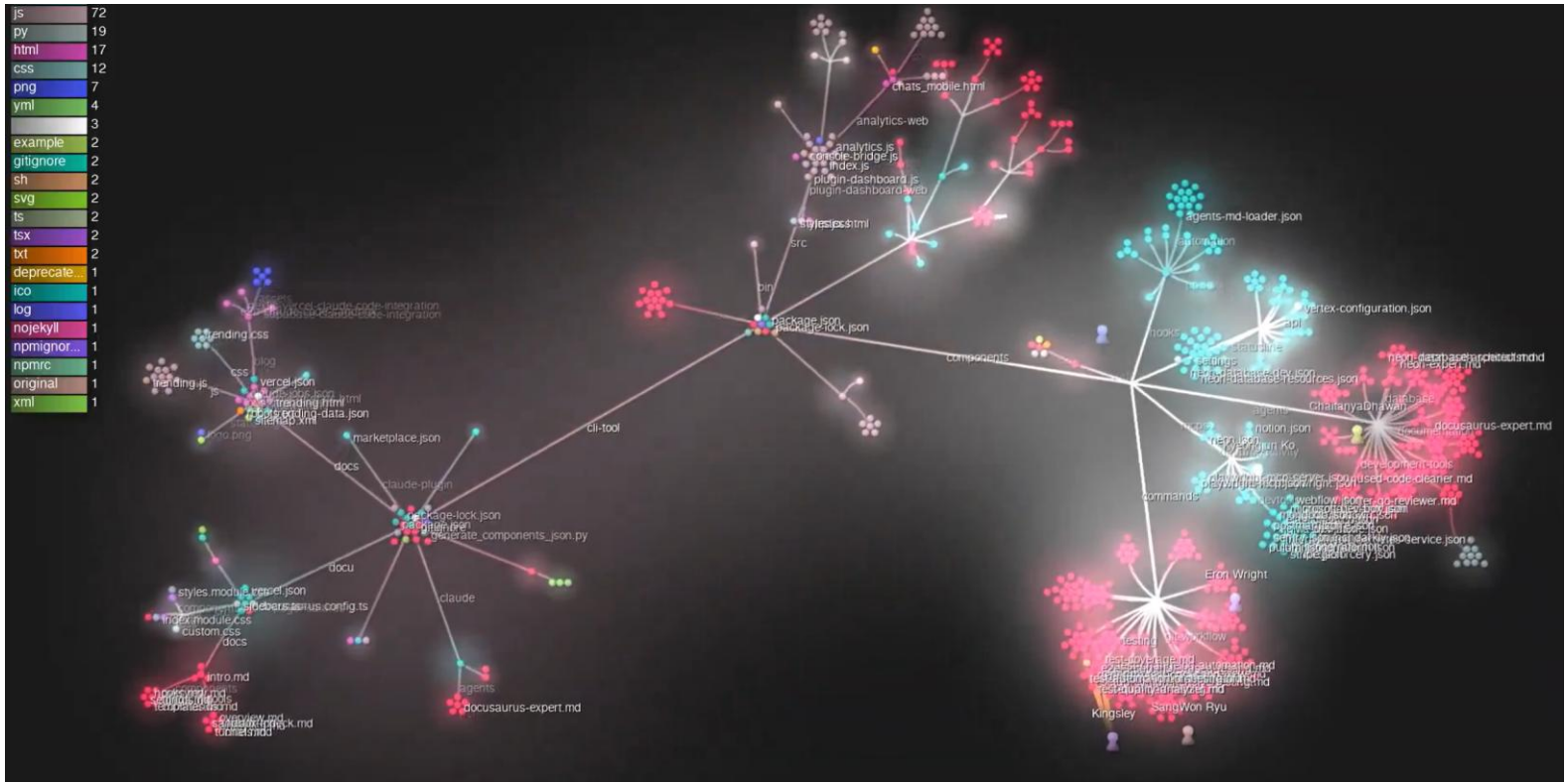
Agents

Coding Agents: Workflow



Agents

Coding Agents: Claude Code Visualisation



Source: https://x.com/dani_avila7/status/1978510902740877344

Available Frameworks

Agents

Which Frameworks are available?

LangGraph

- built on top of LangChain (same team)
- flexible, customizable
- works with any LLM

- More complex

CrewAI

- very intuitive
- suitable for many agents
- supports many LLM providers

- not ideal for very complex tasks

Agents

- very easy to use
- suitable for beginners
- Works with any LLM

Agents

Which Frameworks are available?

Microsoft

AG2 (formerly: AutoGen)

- mostly for two agents
- good for code generation
- very powerful

Google ADK

- Agent development kit
- Framework for development and deployment of AI agents
- Easy to learn
- Multi-agent architecture

Pydantic AI

Pydantic AI

- Claims to provide production grade applications and workflows
- Model-agnostic

Agents

Which Frameworks are available?

Microsoft

Magentic-One

- suitable for beginners
- pre-defined 5 agents: manager, web-surfer, file-surfer, coder, terminal
- built on top of AutoGen
- limited support and documentation

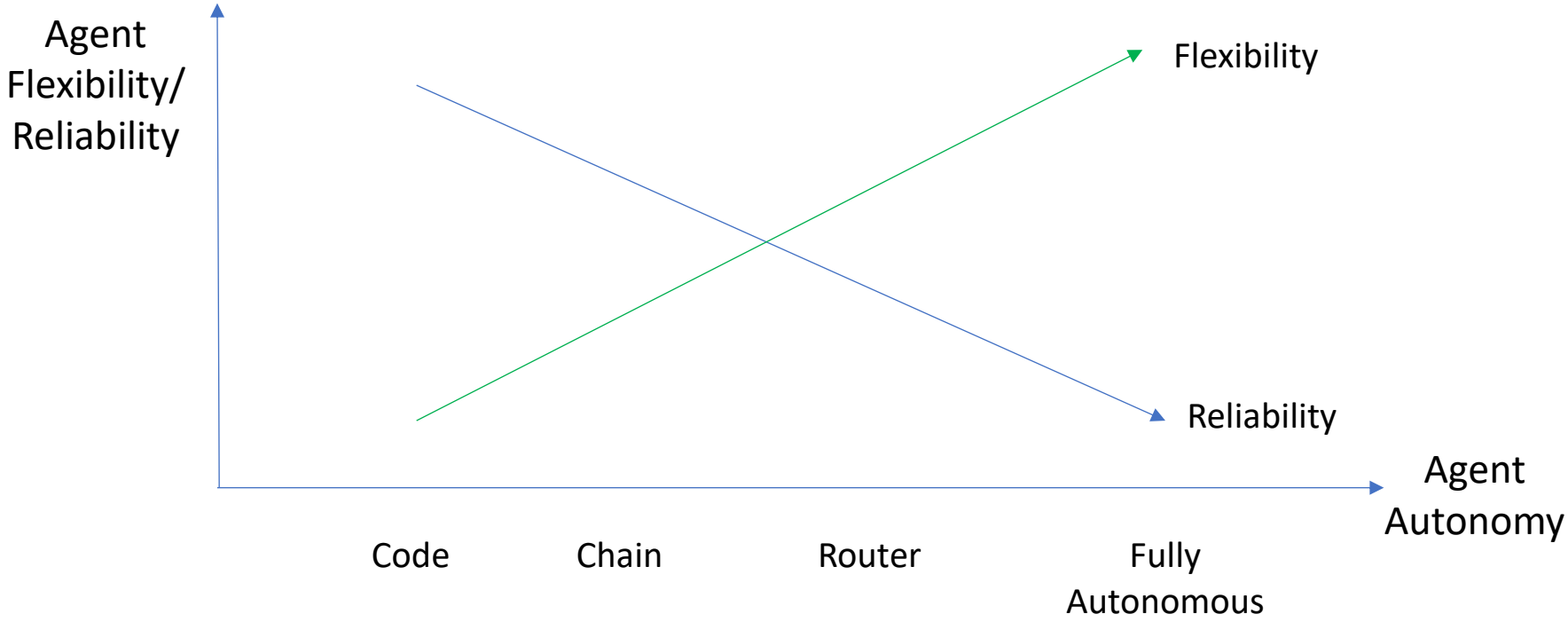
Microsoft

tinytroupe

- multiagent persona simulation for imagination enhancement and business insights
- only works with GPT-4o
- Link

Agents

Flexibility / Reliability vs. Autonomy



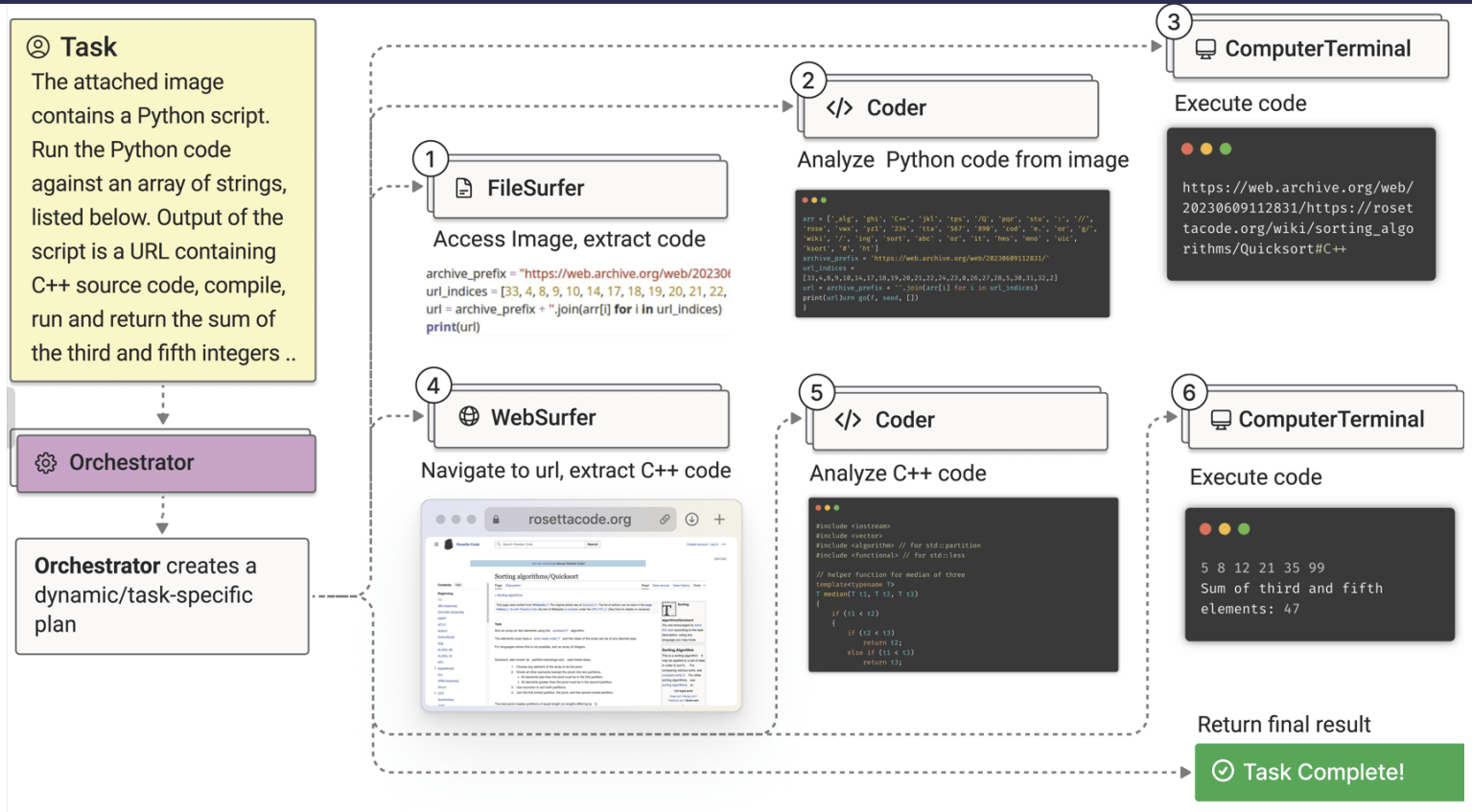
own graph; adapted from: LangChain Academy „Introduction to LangGraph“

Agents

Flexibility vs.

Agents

Magentic One



Agents

Tine Troupe


```python
factory = TinyPersonFactory("One of the largest banks in Brazil, full of bureaucracy and legacy systems.")

customer = factory.generate_person(
    """
    The vice-president of one product innovation. Has a degree in engineering and a MBA in finance.
    Is facing a lot of pressure from the board of directors to fight off the competition from the fintechs.
    """
)
```

✓ 10.1s Python

```python
customer.minibio()
```

✓ 0.0s Python

```
'Lucas Almeida is a 42 year old Vice-President of Product Innovation, Brazilian, currently living in Brazil.'
```

We can now perform the interview.

```python
customer.think("I am now talking to a business and technology consultant to help me with my professional problems.")
```

✓ 0.0s Python

```
Lucas Almeida --> Lucas Almeida: [THOUGHT]
                  > I am now talking to a business and technology consultant to help me with my
                  > professional problems.

TinyPerson(name='Lucas Almeida')
```

```python
customer.listen_and_act("What would you say are your main problems today? Please be as specific as possible.",
                        max_content_length=3000)
```

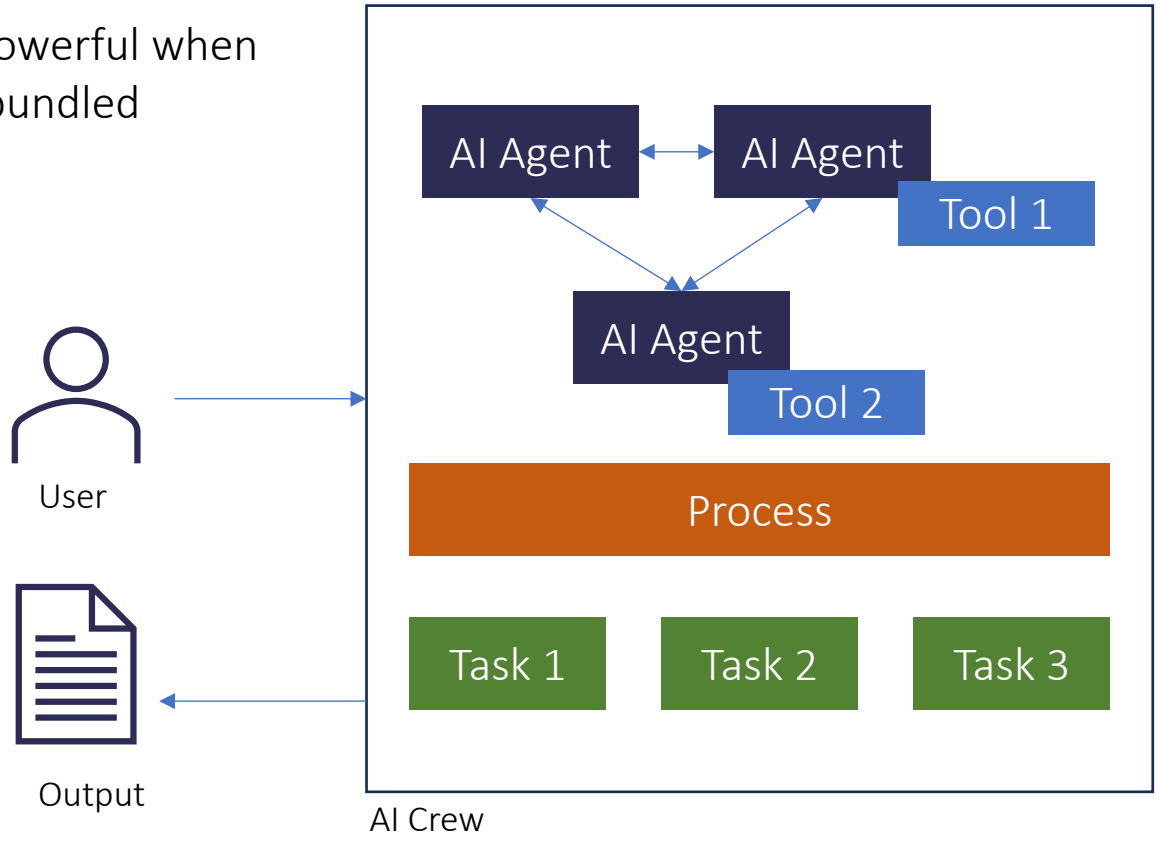
✓ 10.9s Python

crewAI

crewAI

What is an AI Crew?

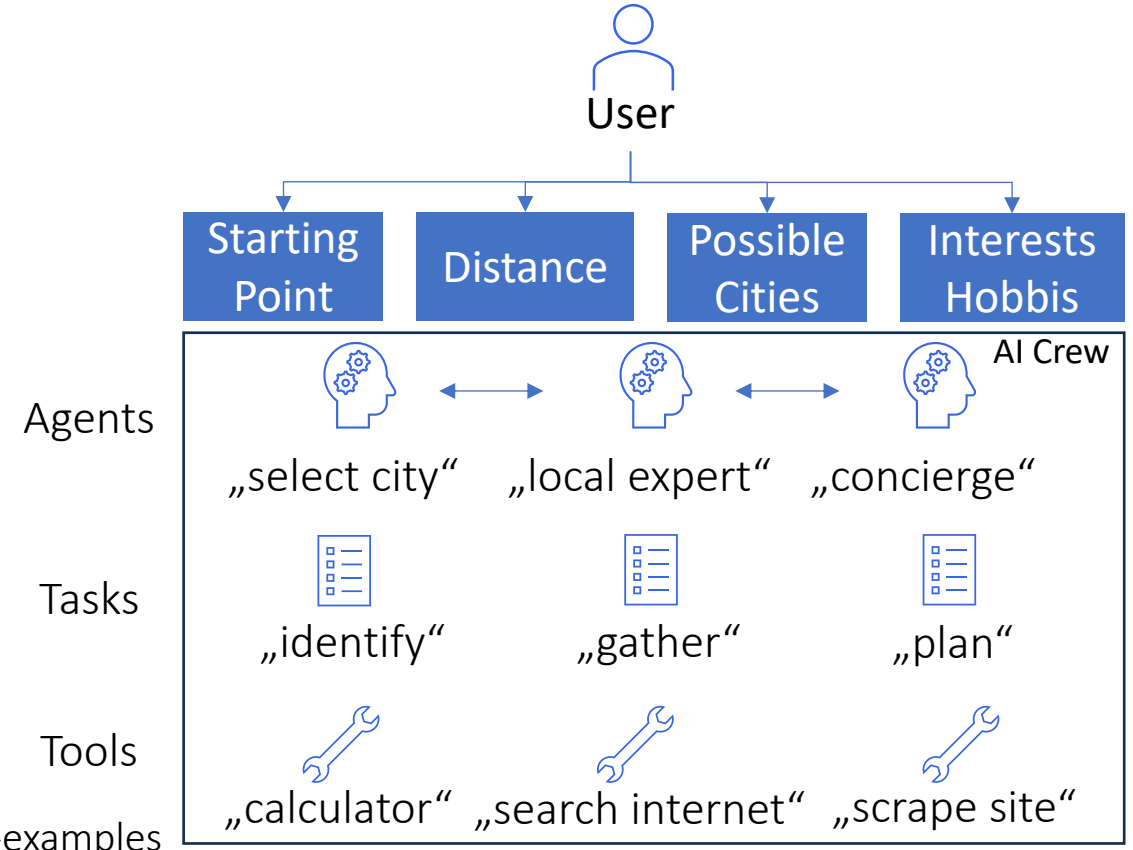
- concept extremely powerful when multiple agents are bundled
- system of experts



crewAI

Example – Plan your vacation

- 1. Define Goal
- 2. User Inputs
- 3. Set up
  - agents
  - tasks
  - if needed:
    - tools
    - process
    - …



Idea found at
https://github.com/joaomdmoura/crewAI-examples

crewAI

Example – Plan your vacation: Agents

AI Crew

Agents

„select city“ | „local expert“ | „concierge“

	„select city“	„local expert“	„concierge“
Role	City Selection Expert	Local Expert	Travel Concierge
Goal	select city based on weather, season, prices	provide insights about selected city	Create travel itineraries with budget and packing suggestions
Backstory	You are an expert in analyzing travel data to pick the best destination.	You are a knowledgeable local guide with extensive knowledge about the city.	specialist in travel planning and logistics

Tasks

„identify“ „gather“ „plan“

crewAI

Example – Plan your vacation: Tasks

AI Crew

Tasks

„identify“

„gather“

„plan“

Description

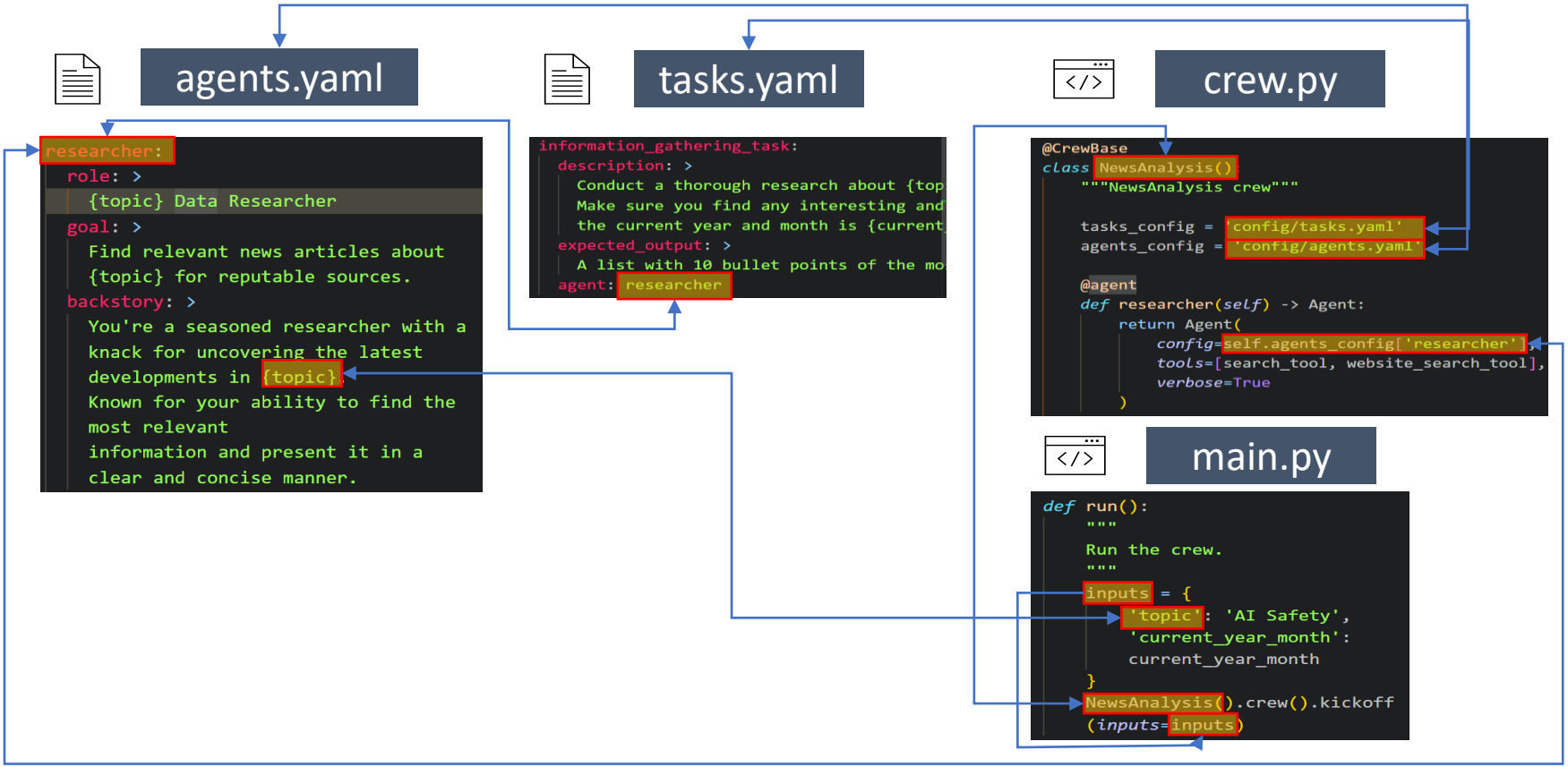
…

…

…

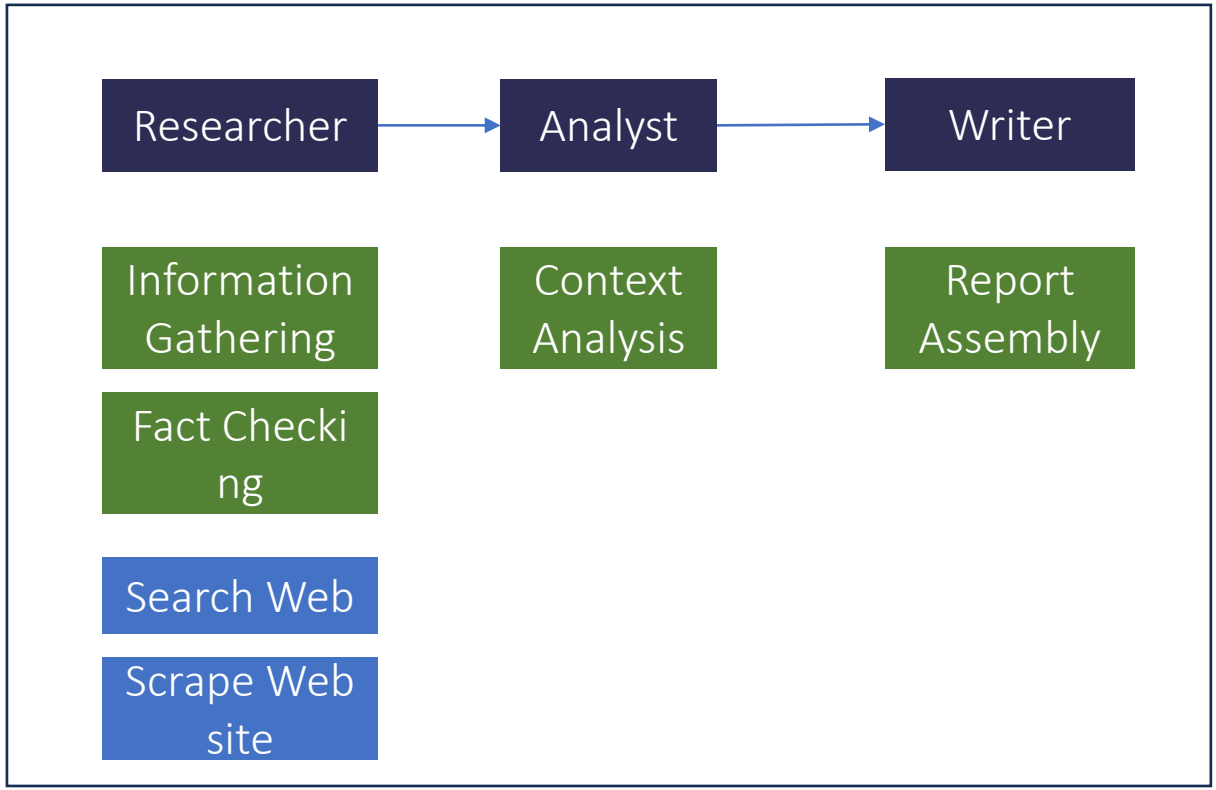
crewAI

File Dependencies



crewAI

Example – Crew Setup



Legend

Agent

Task

Tool

crewAI

Tools

- can be used by Agents for

Searching the Internet

Scraping Websites

Reading Files

Tool	Description
CodeDocsSearchTool	A RAG tool optimized for searching through code documentation and related technical documents.
CSVSearchTool	A RAG tool designed for searching within CSV files, tailored to handle structured data.
DirectorySearchTool	A RAG tool for searching within directories, useful for navigating through file systems.
DOCXSearchTool	A RAG tool aimed at searching within DOCX documents, ideal for processing Word files.
DirectoryReadTool	Facilitates reading and processing of directory structures and their contents.
FileReadTool	Enables reading and extracting data from files, supporting various file formats.
GithubSearchTool	A RAG tool for searching within GitHub repositories, useful for code and documentation search.
SerperDevTool	A specialized tool for development purposes, with specific functionalities under development.
TXTSearchTool	A RAG tool focused on searching within text (.txt) files, suitable for unstructured data.

...

Source: https://docs.crewai.com/core-concepts/Tools/#available-crewai-tools

crewAI

Collaboration

Agents can collaborate on a task to

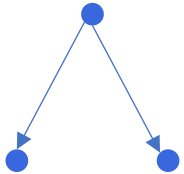
- share information
- assist on a task
- allocate and optimize resources

```
from crewai import Agent, Task, Crew,
Process
```

```
crew = Crew(
    agents=[planner, writer, editor
],
    tasks=[plan, write, edit],
    verbose=2,
    manager_llm=llm,
    process= Process.hierarchical
)
```

Process.sequential

Process.hierarchical



crewAI

Expected Task Outcome

- output formats can be defined in detail

```
class OutputFormat(BaseModel):
    chapter_title: str
    bullet_points: list[str]
```

```
Task(
    description=(“..."),
    expected_output="A well-written slideset …“,
    agent=editor,
    output_format="markdown",
    output_format_model=OutputFormat,
    output_format_description=(
        "The output format is a markdown file …"
    ),
    output_file = "slideset.md"
)
```

crewAI

Use of other LLMs

- set up an llm-object
- pass it as a parameter

```
from langchain_groq import ChatGroq

llm=ChatGroq(temperature=0,
             model_name=MODEL,
             api_key=os.environ["GR
OQ_API_KEY"])

planner = Agent(
    role="…",
    goal="…",
    backstory="…",
    allow_delegation=False,
    llm=llm,
    verbose=True
)
```

crewAI

Installation

- Problem:
  - crewAI is a huge framework with many dependencies
  - Does currently have issues with Python 3.13

```
C:\Temp\crewai>where python
C:\Python313\python.exe
C:\Users\BertGollnick\AppData\Local\Programs\Python\Python312\python.exe
C:\Users\BertGollnick\AppData\Local\Microsoft\WindowsApps\python.exe
```

- Start from totally clean environment:

```
C:\Temp\crewai>C:\Users\BertGollnick\AppData\Local\Programs\Python\Python312\python.exe -m venv .venv
```

```
C:\Temp\crewai>.venv\Scripts\activate
```

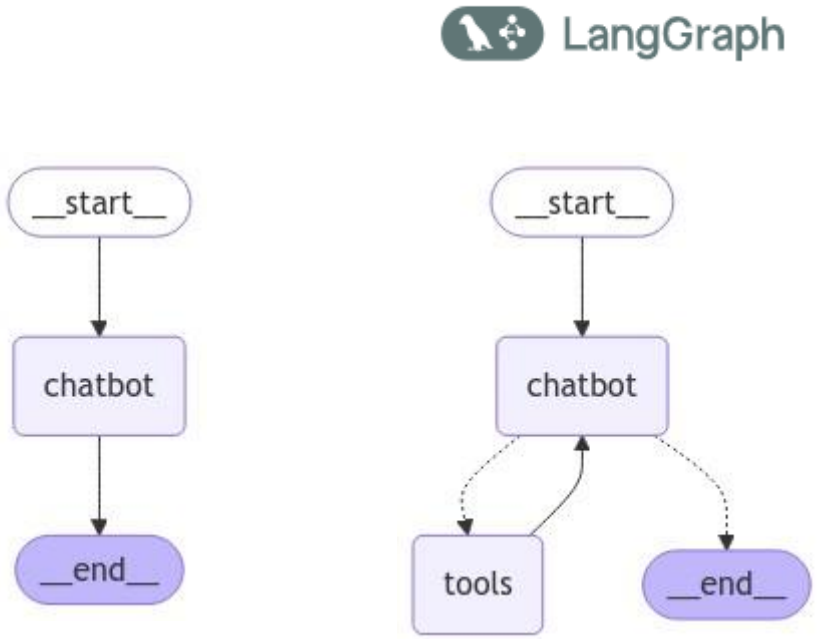
```
(.venv) C:\Temp\crewai>pip install crewai[tools]
```

LangGraph

LangGraph

Introduction

- agentic system
- graph-based representation (directed acyclical graph DAG)
- integrates well with LangChain ecosystem
- focuses on complex workflows
- based on nodes (tasks), and edges (dependencies)



Typical LangGraph graphs

LangGraph

State

- Shared data structure
- Nodes read from and write to
- TypedDict…for type hinting
- Annotated…new messages are appended to existing list

```python
class ConversationState(TypedDict):
    """
    Represents the state of our conversation graph.

    Attributes:
        messages: A list of BaseMessage objects, annotated to
        append new messages.
        summary: An optional string to store the conversation
        summary.
    """
    messages: Annotated[List[BaseMessage], operator.add]
    summary: str = ""
```

LangGraph

Chat with Summarization

messages:

System Message

cnt=2

Human Message

AI Message

messages:

...

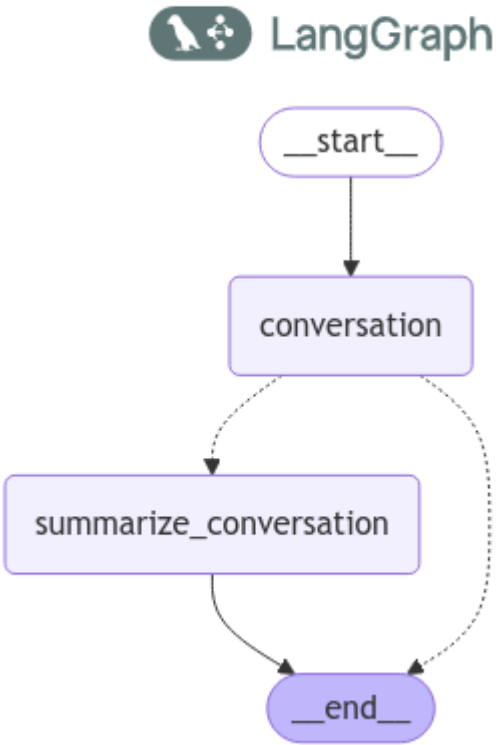
cnt>6:

Human Message

AI Message

summary:

Summary



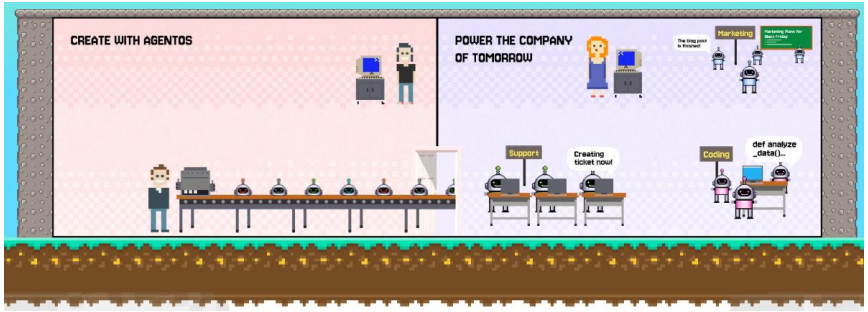
AG2 (formerly autogen)

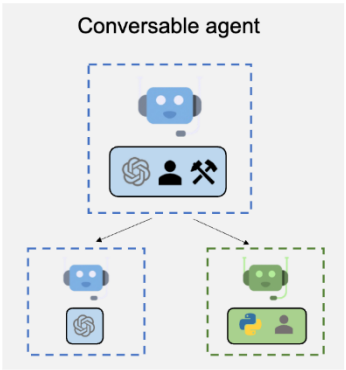
AG2

Introduction

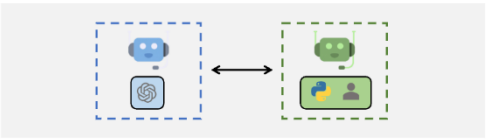
- Open-Source framework for building AI agents

- Installation: pip install ag2
- Docker: optional

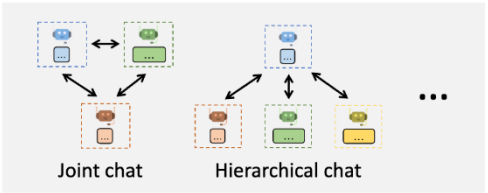




Agent Customization



Multi-Agent Conversations

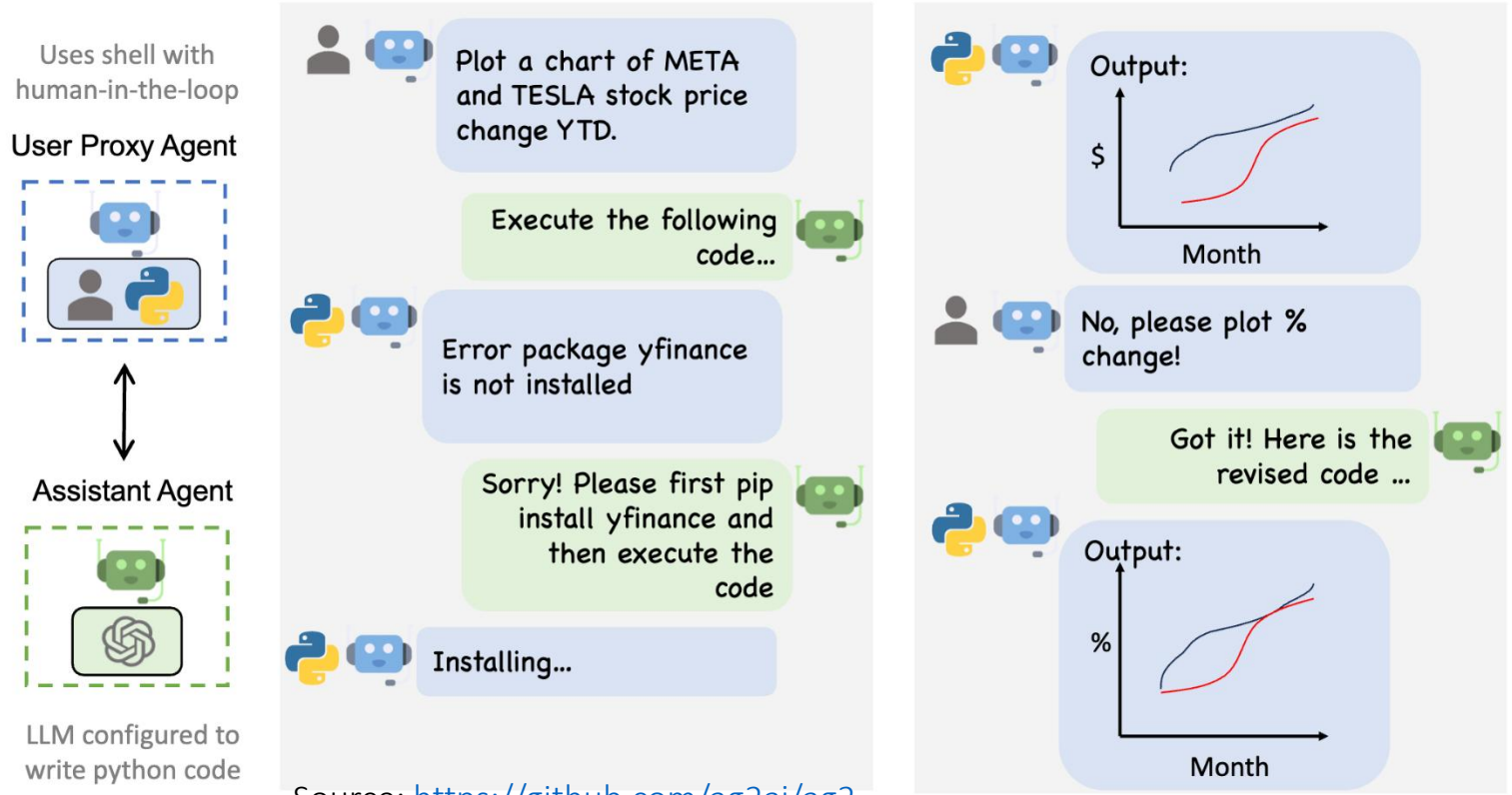


Flexible Conversation Patterns

Source: https://github.com/ag2ai/ag2

AG2

Introduction

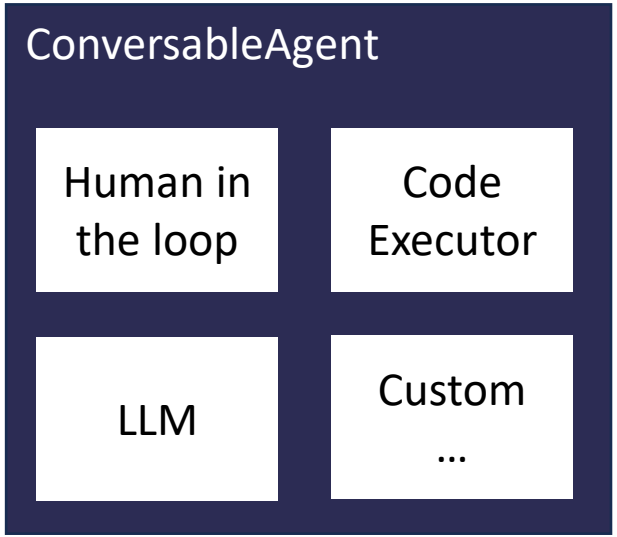


Source: https://github.com/ag2ai/ag2

AG2

Agent

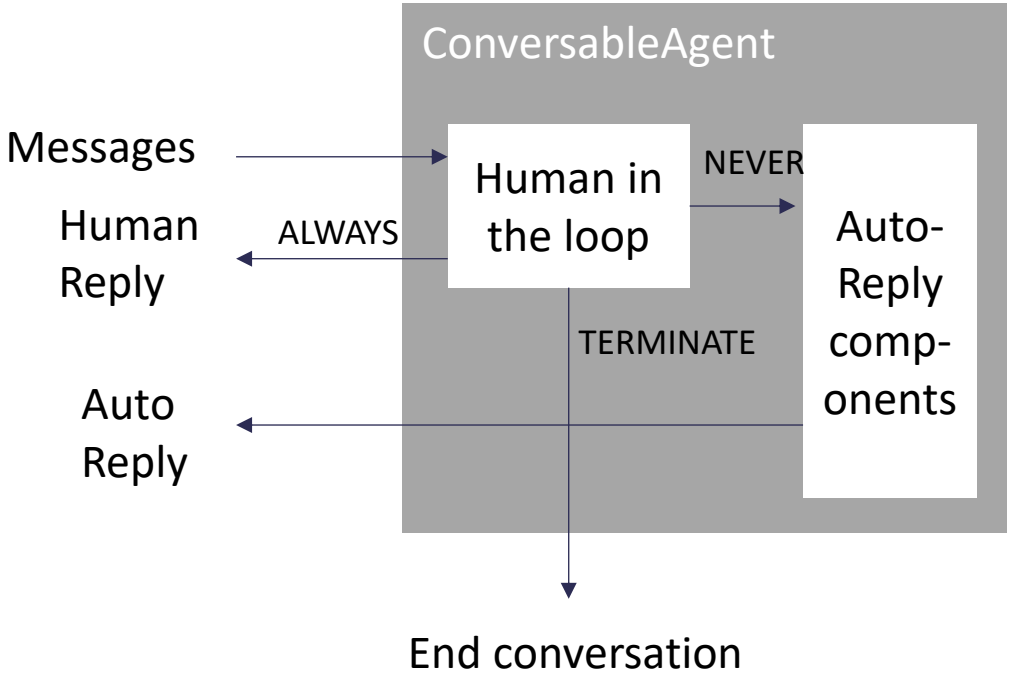
- AG2 agent
  - entity that can send and receive messages to and from other agents
  - agent can be run by
    - models,
    - code executors,
    - human, or
    - a combination of above



AG2

Human in the loop

- HITL in front of Auto reply, intercepts incoming messages, and decides to pass to auto-reply, or to human feedback
- customizable through *human_input_mode* parameter
- modes:
  - NEVER, TERMINATE (default), ALWAYS



AG2

Tools

- agents can use tools
- register_for_llm
  - exposes tool to LLM
  - allows LLM to reason about tool, decide when to call
- register_for_execution
  - handles execution of tool when LLM decides to call it
  - connects logical request generated by LLM to process
  - without it, even if LLM decides to use a tool, there would be no backend to execute tool‘s functionality

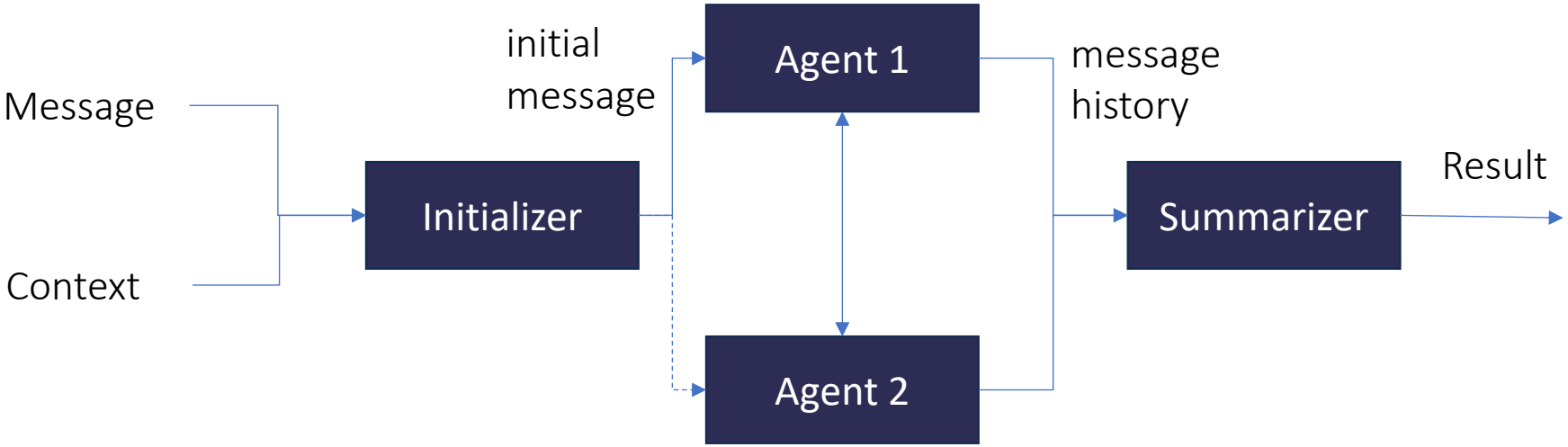
```
# %% create an agent with a tool
my_assistant = ConversableAgent(
    name="my_assistant",
    system_message="You are a helpful AI assistant.",
    llm_config=config_list
)

# register the tool signature at agent level
my_assistant.register_for_llm(
    name="get_current_date",
    description="Returns the current date in the form
at YYYY-MM-DD."
)(get_current_date)

# register the tool function at execution level
my_assistant.register_for_execution(name="get_current
_date")(get_current_date)
```

AG2

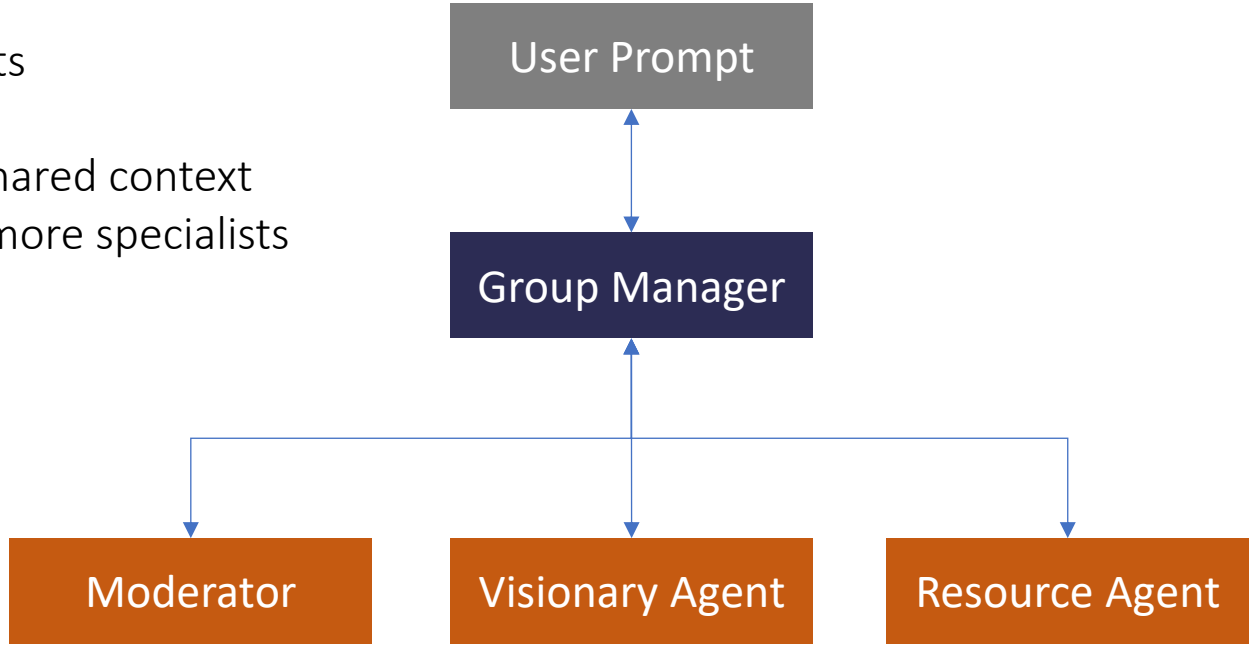
Conversation Patterns: Two Agents Chatting



AG2

Group Chat

- Multiple agents interact
- Allows a team of specialists
- Dynamic collaboration
- All agents use the same shared context
- Easily scalable by adding more specialists



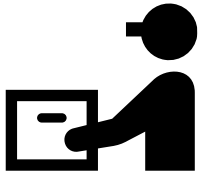
OpenAI Agents SDK

OpenAI Agents SDK

Introduction

Agent

- LLM
- Instructions
- tools



Handoffs

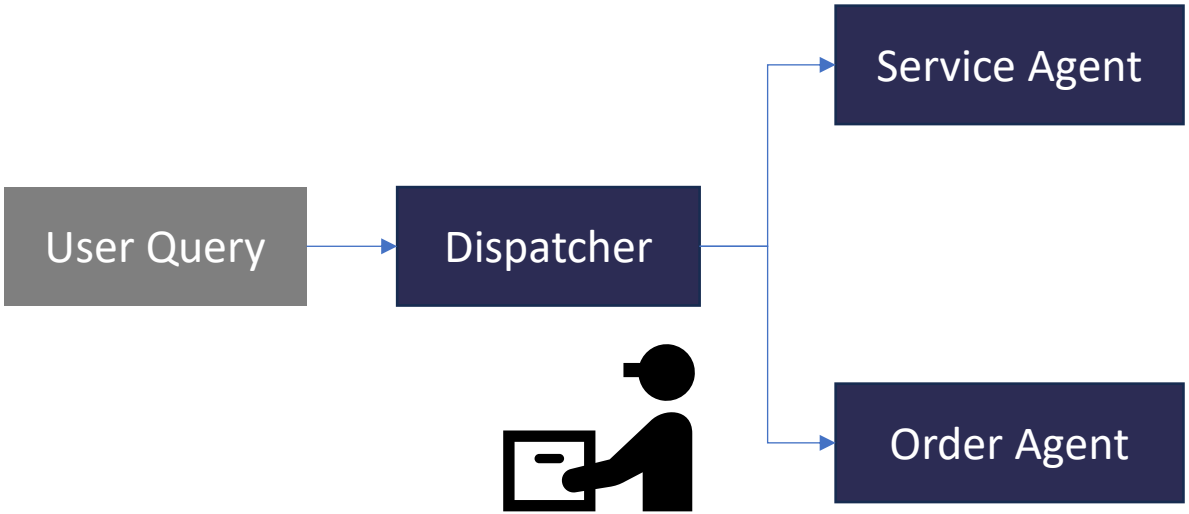
- Agents delegate tasks to other agents

Guardrails

- Enables input validation

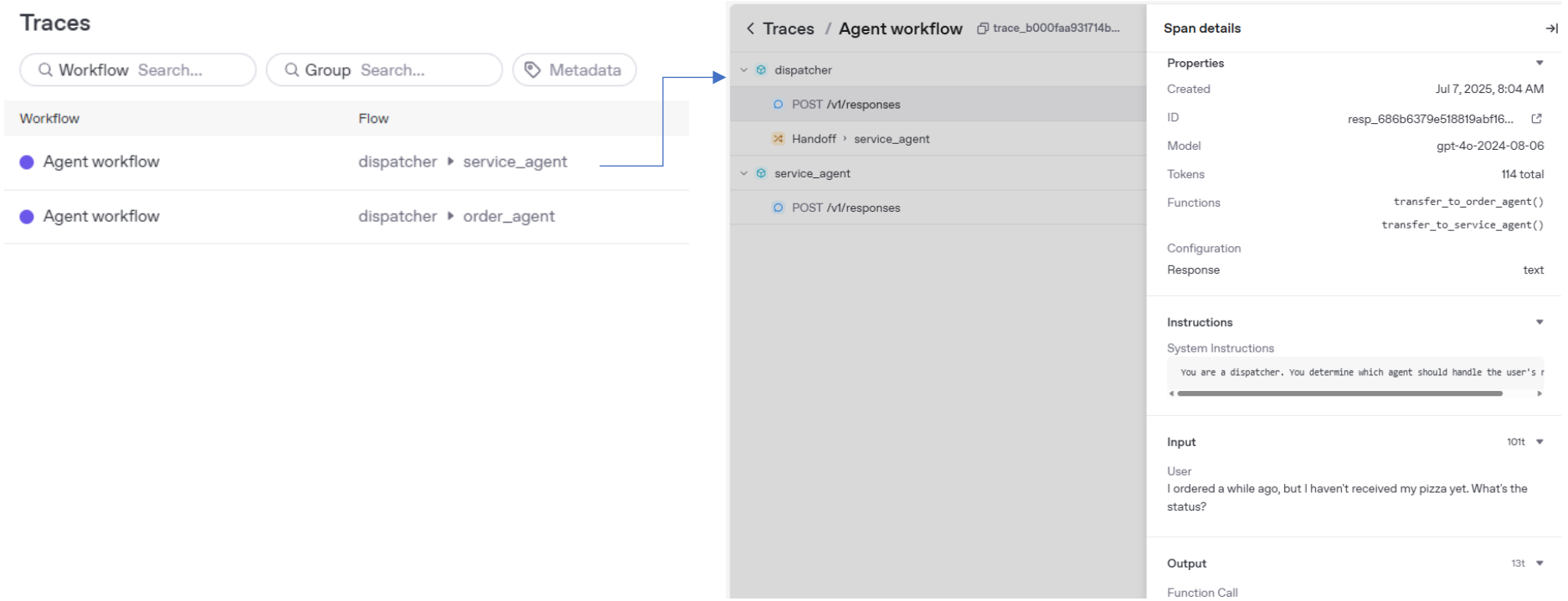
OpenAI Agents SDK

Handoff Implementation



OpenAI Agents SDK

Traces



OpenAI Agents SDK

Running Options

Runner.run()

- Asynchronous
- Returns RunResult

Runner.run_sync()

- Synchronous
- Returns RunResult

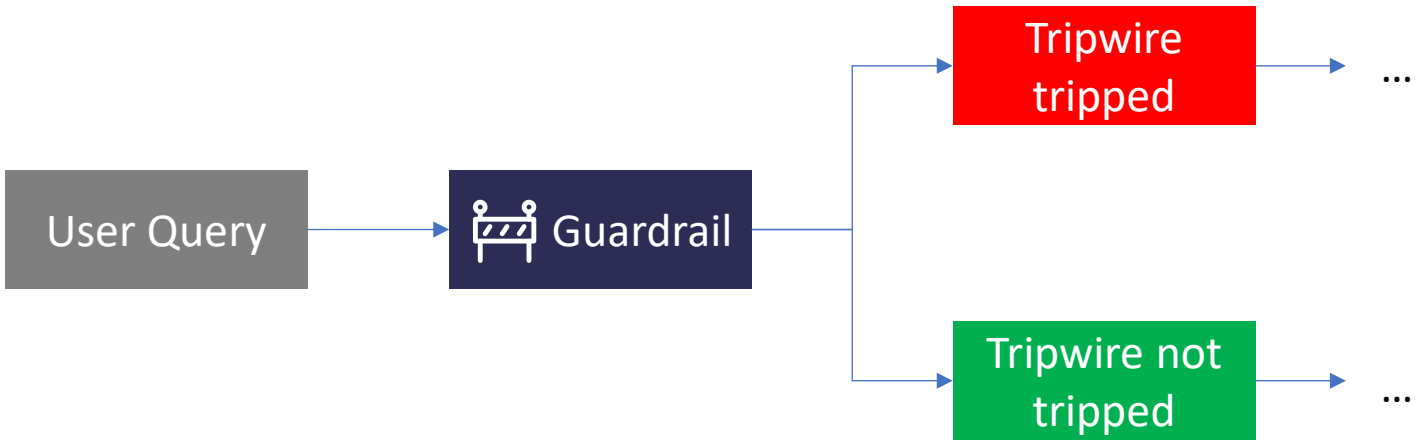
Runner.run_streamed()

- asynchronous
- Returns RunResultStreaming

OpenAI Agents SDK

Guardrails

- Run parallel to agents
- Allow for input validation



Google ADK

Google ADK

Agent Development Kit (ADK)

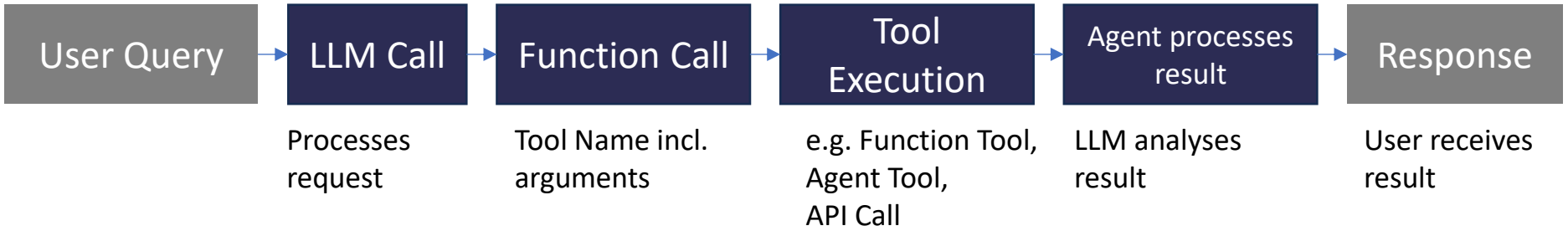
- Framework for developing AND deploying AI agents
- model-agnostic
- deployment-agnostic
- Many pre-built tools
- Modular and scalable

Agent Development Kit

Google ADK

Tool Use

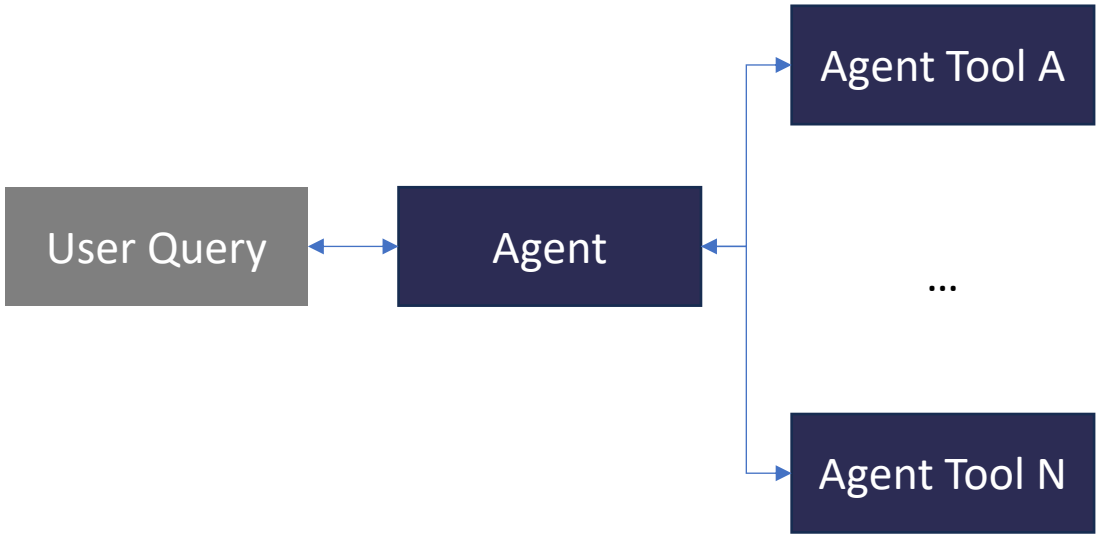
- tool is a feature provided to agent
- Enables to perform actions and interact with the world



Oriented on https://google.github.io/adk-docs/tools/#what-is-a-tool

Google ADK

Agent Tool Use



Google ADK

Run Agent

parent_folder

navigate here

adk run agent_project

agent_project

__init__.py

adk web

.env.py

agent.py