AGENTIC AI

ORCHESTRATING AI AGENTS WITH CREWAI

ALESSANDRO ROMANO

THE NEURAL MAZE

HOW TO USE THIS WORKSHOP

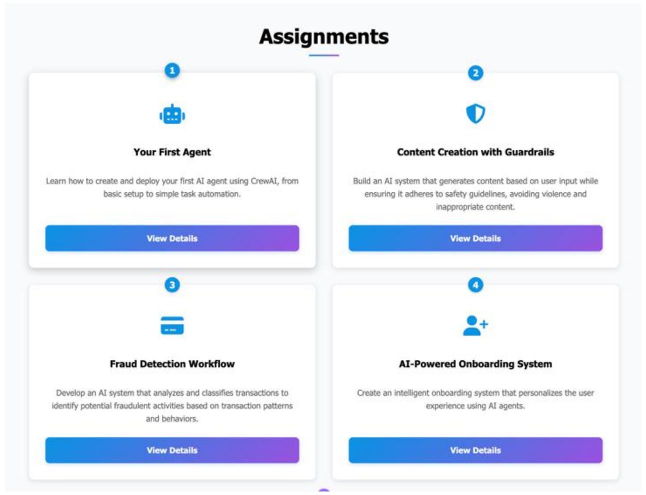
1

AGENTIC AI

ORCHESTRATING AI AGENTS WITH CREWAI

ALESSANDRO ROMANO

THE NEURAL MAZE



Assignments

1 **Your First Agent**

Learn how to create and deploy your first AI agent using CrewAI, from basic setup to simple task automation.

View Details

2 **Content Creation with Guardrails**

Build an AI system that generates content based on user input while ensuring it adheres to safety guidelines, avoiding violence and inappropriate content.

View Details

3 **Fraud Detection Workflow**

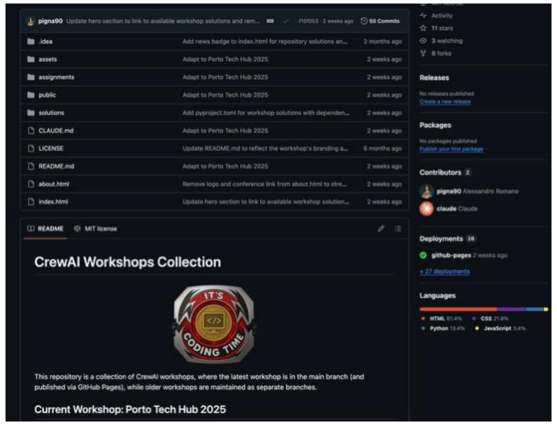
Develop an AI system that analyzes and classifies transactions to identify potential fraudulent activities based on transaction patterns and behaviors.

View Details

4 **AI-Powered Onboarding System**

Create an intelligent onboarding system that personalizes the user experience using AI agents.

View Details



pigna90 Update hero section to link to available workshop solutions and rem... f15f053 · 2 weeks ago 50 Commits

.idea	Add news badge to index.html for repository solutions an...	2 months ago
assets	Adapt to Porto Tech Hub 2025	2 weeks ago
assignments	Adapt to Porto Tech Hub 2025	2 weeks ago
public	Adapt to Porto Tech Hub 2025	2 weeks ago
solutions	Add pyproject.toml for workshop solutions with dependen...	2 weeks ago
CLAUDE.md	Adapt to Porto Tech Hub 2025	2 weeks ago
LICENSE	Update README.md to reflect the workshop's branding a...	6 months ago
README.md	Adapt to Porto Tech Hub 2025	2 weeks ago
about.html	Remove logo and conference link from about.html to stre...	2 weeks ago
index.html	Update hero section to link to available workshop solution...	2 weeks ago

README MIT license

CrewAI Workshops Collection

This repository is a collection of CrewAI workshops, where the latest workshop is in the main branch (and published via GitHub Pages), while older workshops are maintained as separate branches.

Current Workshop: Porto Tech Hub 2025

Activity
11 stars
3 watching
6 forks

Releases
No releases published
Create a new release

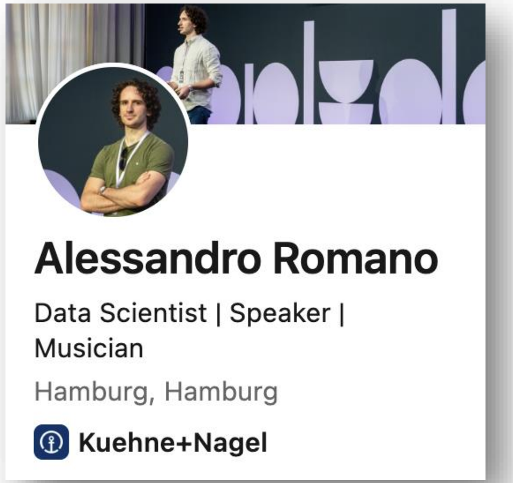
Packages
No packages published
Publish your first package

Contributors 2
pigna90 Alessandro Romano
claude Claude

Deployments 28
github-pages 2 weeks ago
+ 27 deployments

Languages
HTML 61.4% CSS 21.8%
Python 13.4% JavaScript 3.4%

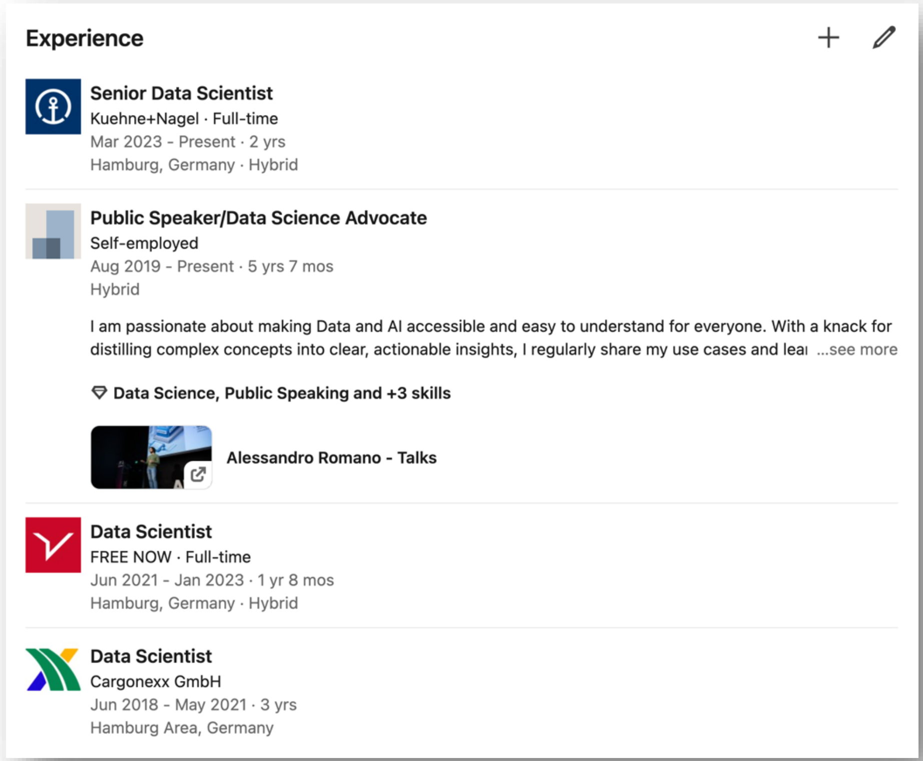
ABOUT ME


Alessandro Romano

Data Scientist | Speaker | Musician

Hamburg, Hamburg

Kuehne+Nagel


Experience

Senior Data Scientist
Kuehne+Nagel · Full-time
Mar 2023 - Present · 2 yrs
Hamburg, Germany · Hybrid

Public Speaker/Data Science Advocate
Self-employed
Aug 2019 - Present · 5 yrs 7 mos
Hybrid

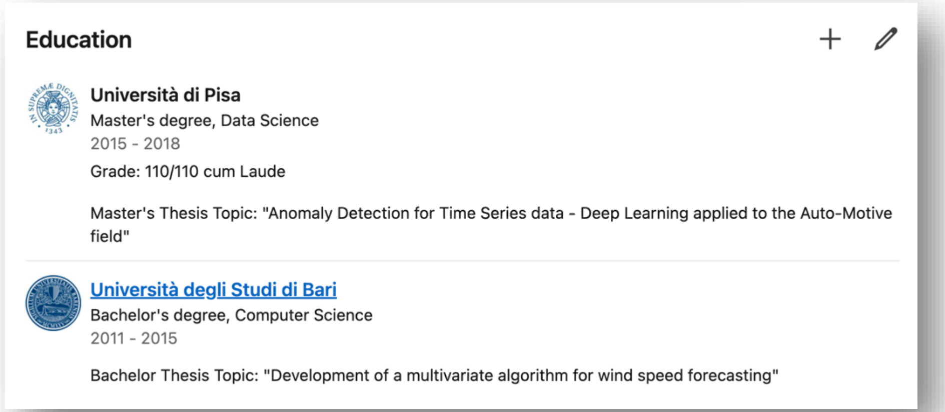
I am passionate about making Data and AI accessible and easy to understand for everyone. With a knack for distilling complex concepts into clear, actionable insights, I regularly share my use cases and lea ...see more

Data Science, Public Speaking and +3 skills

Alessandro Romano - Talks

Data Scientist
FREE NOW · Full-time
Jun 2021 - Jan 2023 · 1 yr 8 mos
Hamburg, Germany · Hybrid

Data Scientist
Cargonexx GmbH
Jun 2018 - May 2021 · 3 yrs
Hamburg Area, Germany


Education

Università di Pisa
Master's degree, Data Science
2015 - 2018
Grade: 110/110 cum Laude

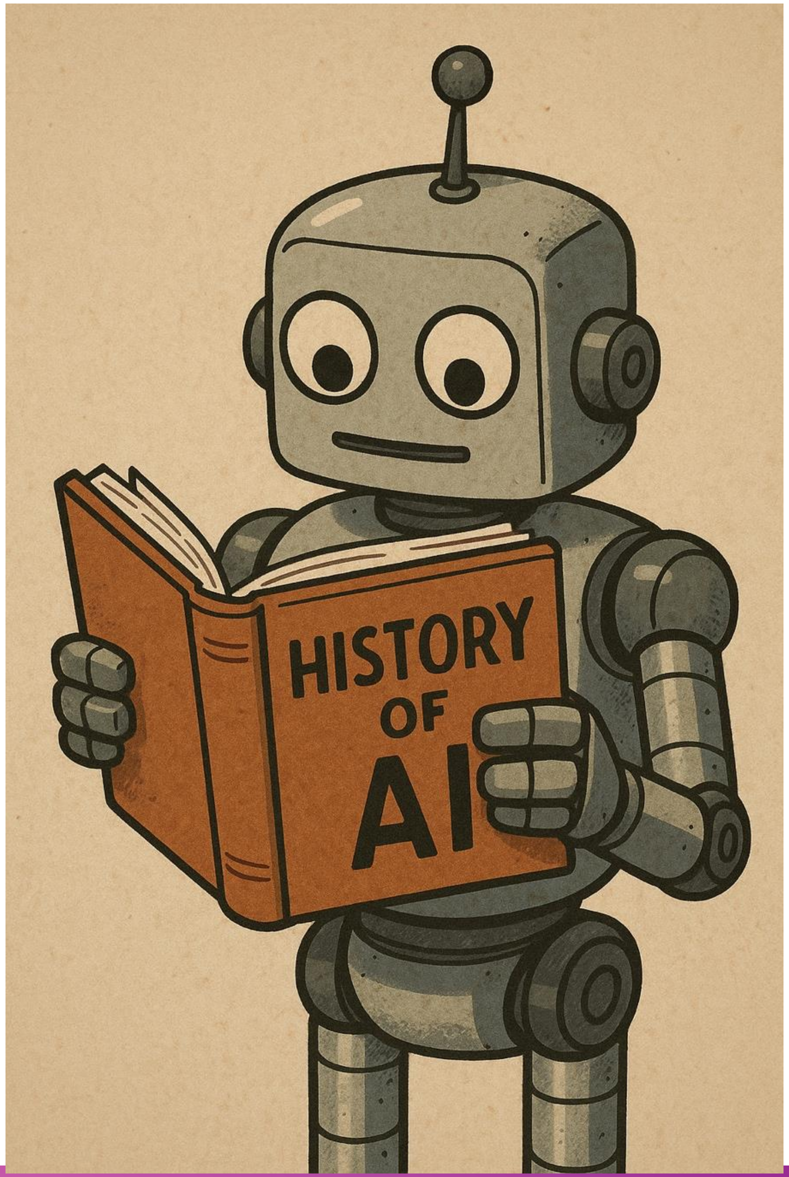
Master's Thesis Topic: "Anomaly Detection for Time Series data - Deep Learning applied to the Auto-Motive field"

Università degli Studi di Bari
Bachelor's degree, Computer Science
2011 - 2015

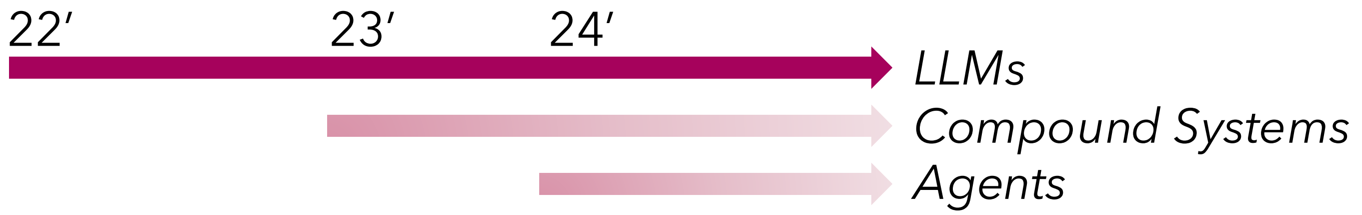
Bachelor Thesis Topic: "Development of a multivariate algorithm for wind speed forecasting"

RECENT HISTORY

How the last 5y changed the AI Landscape…

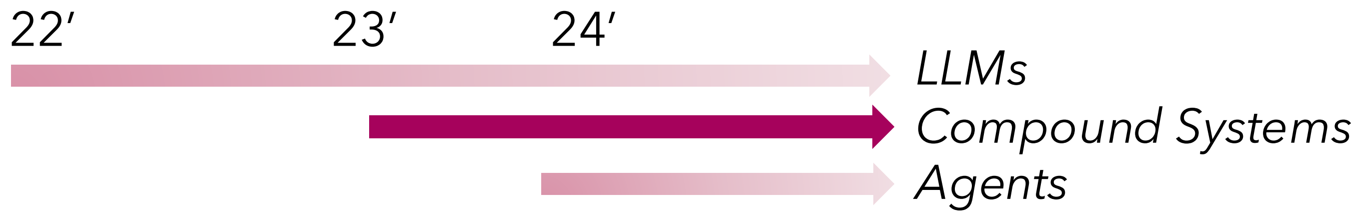


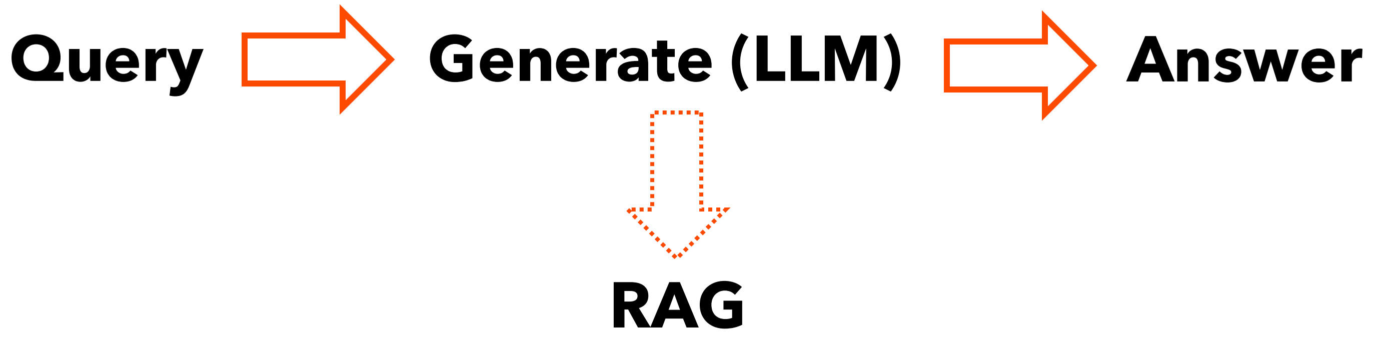
THE DAWN OF LLMS



Query ⇨ **Generate (LLM)** ⇨ **Answer**

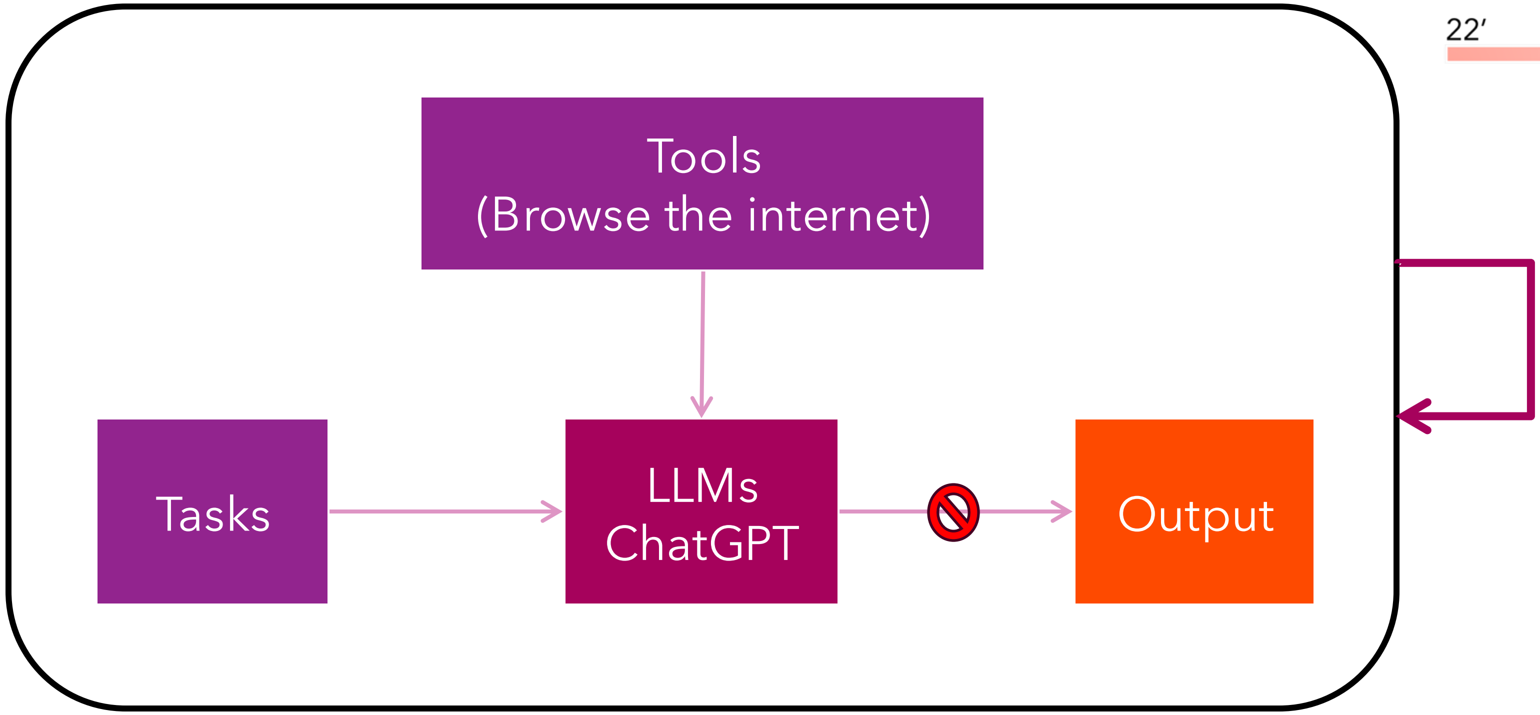
COMPOUND SYSTEMS

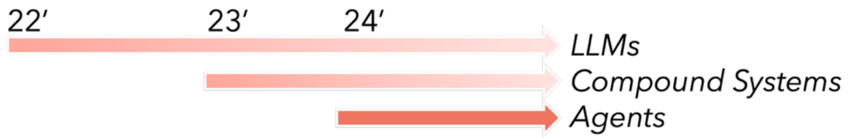




(Retrieval-Augmented Generation)

LLM AGENTS

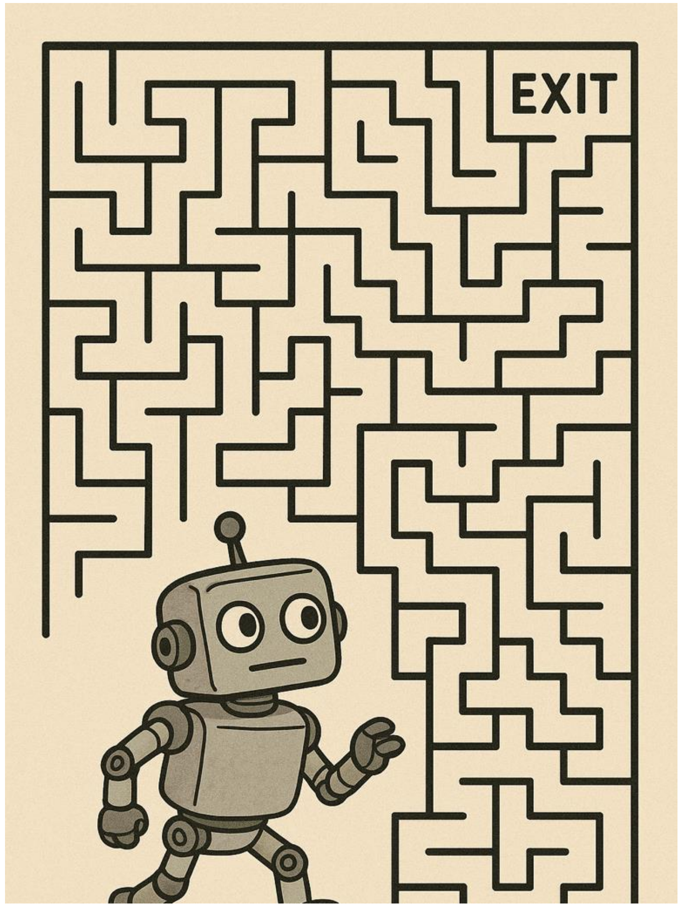




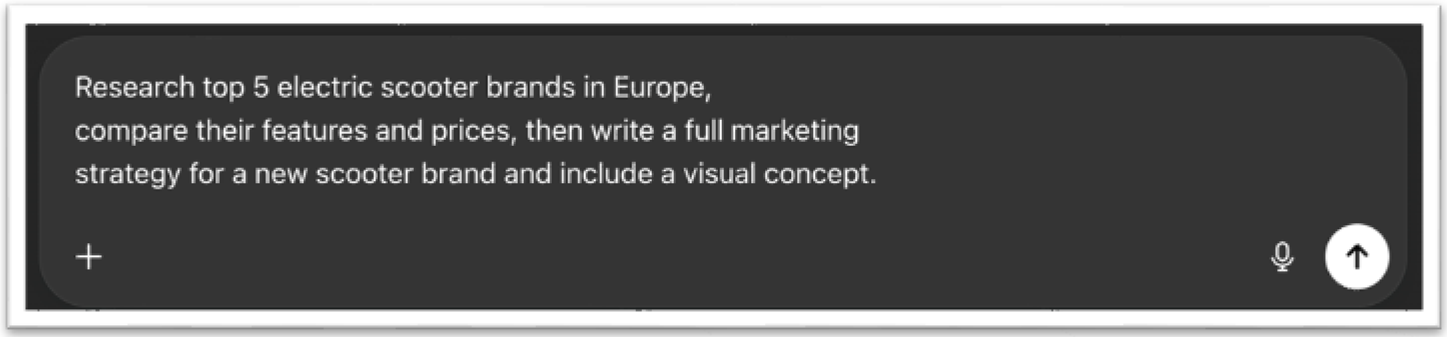
- **Access Memory**
- **Actions (Browse Web)**
- **Incorporates "Reasoning"**

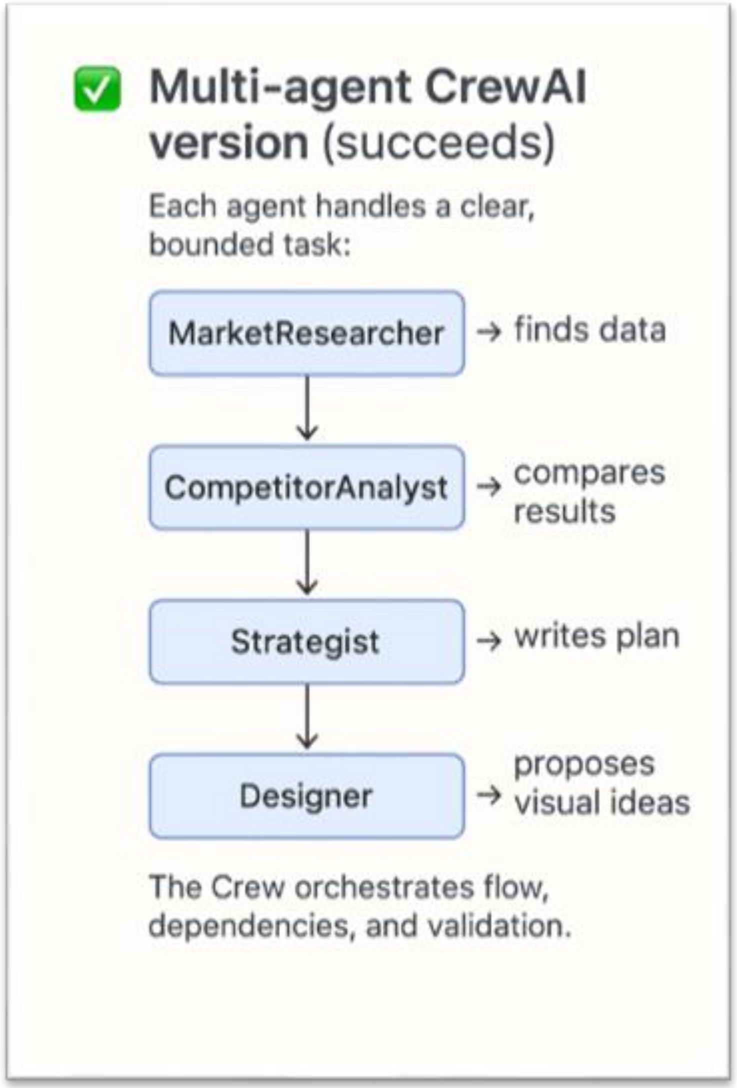
(AI) AGENTS VS DYNAMIC SYSTEMS

Agents **interact** with dynamic systems by perceiving changes, taking **actions**, and using LLMs as **reasoning** cores to adapt, plan, and make context-aware **decisions** in real time.



PROMPTS VS. AGENTS


Research top 5 electric scooter brands in Europe, compare their features and prices, then write a full marketing strategy for a new scooter brand and include a visual concept.


✅ Multi-agent CrewAI version (succeeds)

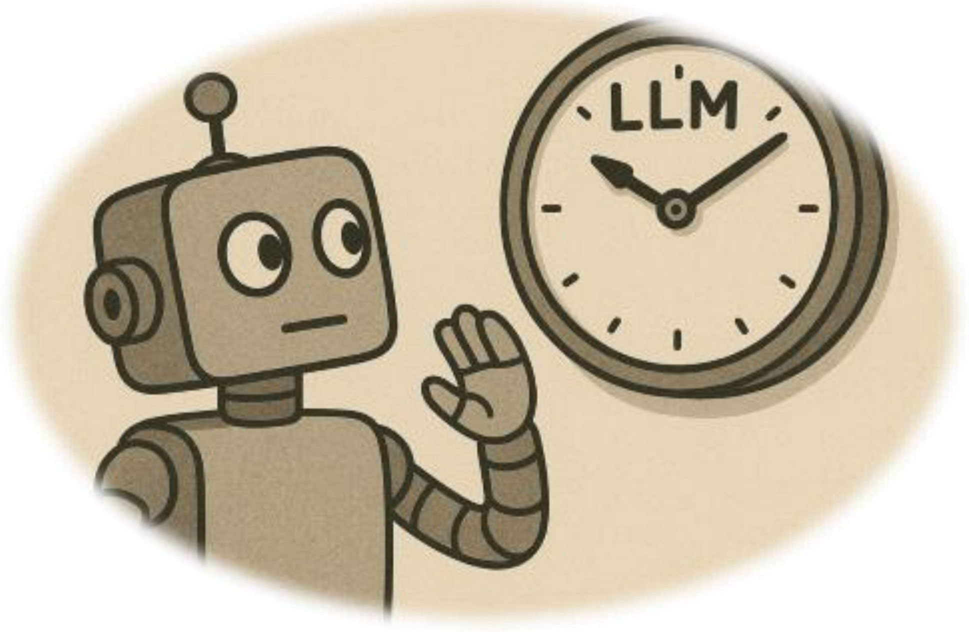
Each agent handles a clear, bounded task:

MarketResearcher → finds data
CompetitorAnalyst → compares results
Strategist → writes plan
Designer → proposes visual ideas

The Crew orchestrates flow, dependencies, and validation.

When your workflow includes **multiple steps, branching logic, dependencies, or validations**, a **CrewAI multi-agent** setup is more robust – while a **single prompt** tends to collapse under ambiguity or task overload.

WHY NOW?



Large Language Models changed the game.

Before LLMs, agents followed rigid, rule-based logic. Now they can:

- **Reason** dynamically in open-ended environments
- **Interpret** context and adapt on the fly
- **Chain decisions** across multiple steps and tools
- **Communicate** naturally with humans and systems

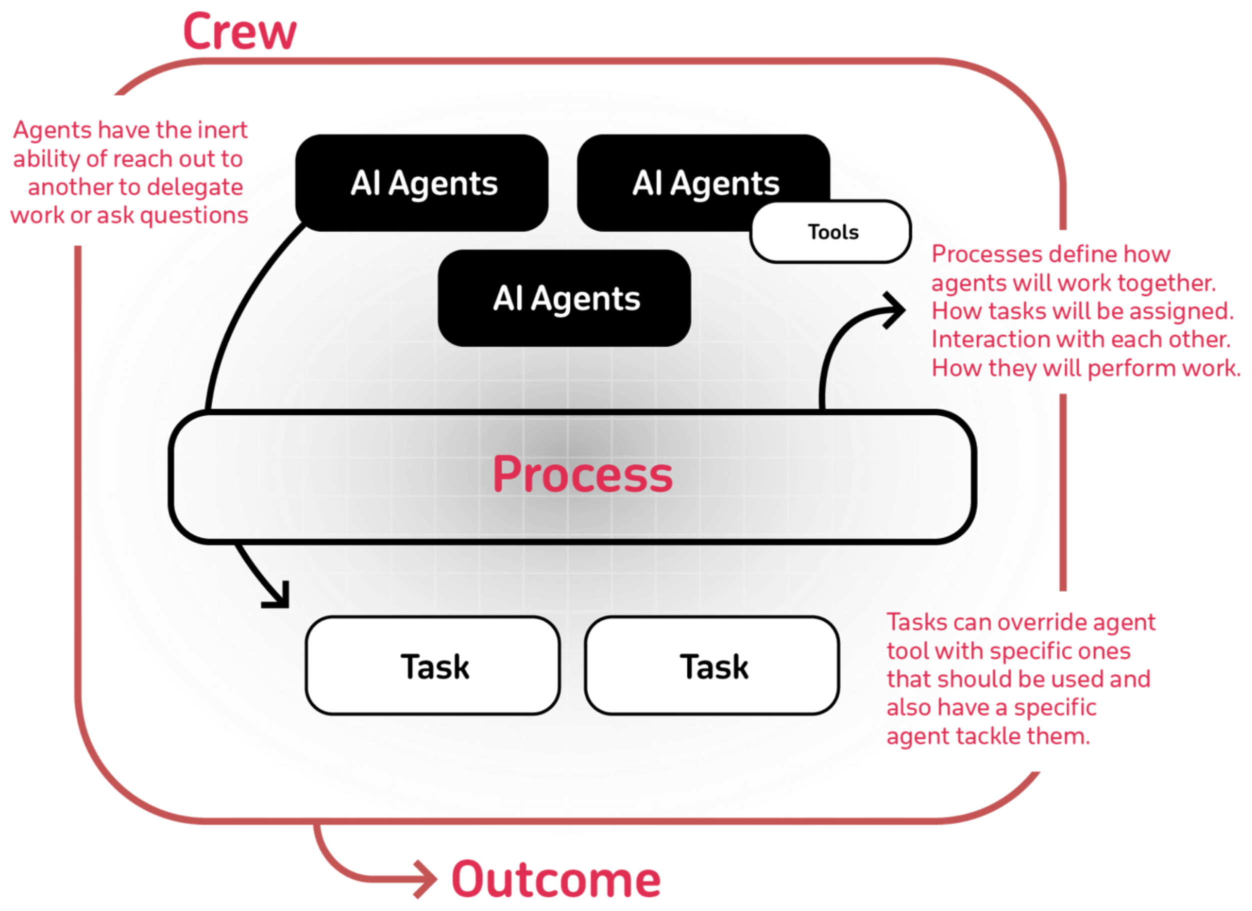
LLMs give agents the flexibility and intelligence needed to operate in real-world, unstructured tasks.

CREWAI

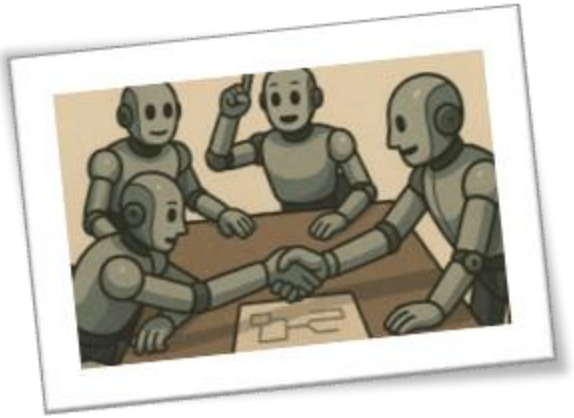
***CrewAI** is a framework for building AI agents that reason, collaborate, and act autonomously—so you can focus on outcomes, not infrastructure.*

- **Implements reasoning** and orchestration for you
- **Agentic AI** under the hood—no need to build it from scratch
- **LLM-agnostic**: use any model you prefer
- **Focus on solving your problem**, not on building agent infrastructure

CREWAI ARCHITECTURE


Crew
Agents have the inert ability of reach out to another to delegate work or ask questions
AI Agents
AI Agents
Tools
AI Agents
Processes define how agents will work together. How tasks will be assigned. Interaction with each other. How they will perform work.
Process
Task
Task
Tasks can override agent tool with specific ones that should be used and also have a specific agent tackle them.
Outcome

BUILD YOUR TEAM!



Planner Agent – Turns goals into tasks

Content Creator Agent – Writes content

API Caller Agent – Connects to APIs

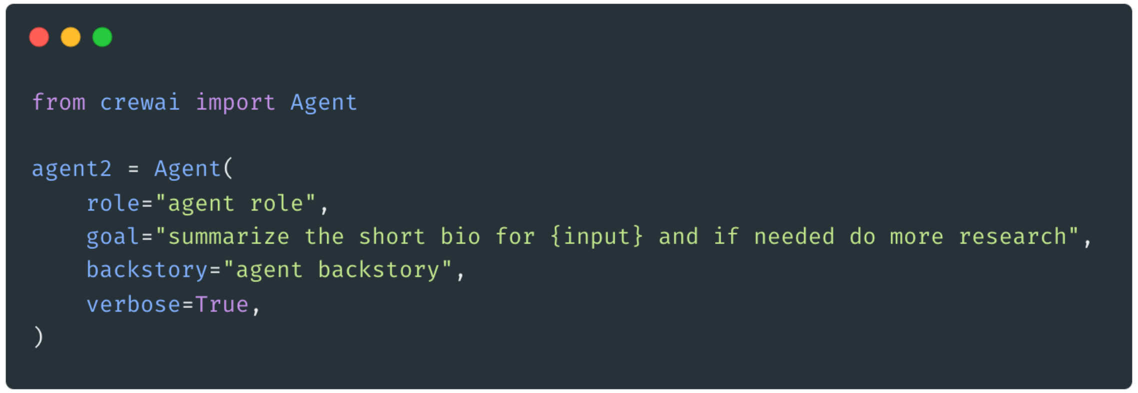
Evaluator Agent – Checks results

AGENTS

An **agent** is an **autonomous unit** programmed to:

- Perform tasks
- Make decisions
- Communicate with other agents

Think of an agent as a member of a team, with specific skills and a particular job to do. Agents can have different roles like *Researcher*, *Writer*, or *Customer Support*, each contributing to the overall goal of the crew.

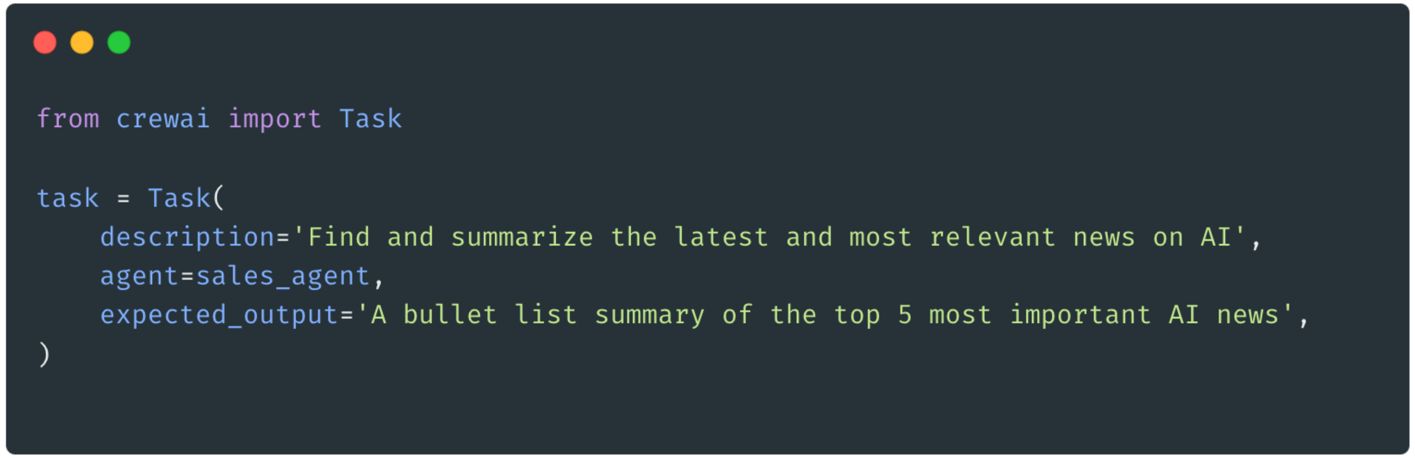


```python
from crewai import Agent

agent2 = Agent(
    role="agent role",
    goal="summarize the short bio for {input} and if needed do more research",
    backstory="agent backstory",
    verbose=True,
)
```

https://docs.crewai.com/core-concepts/Agents/

TASKS

In the crewAI framework, **tasks** are assignments given to agents, including details like descriptions, responsible agents, and required tools. Tasks can be collaborative, involving multiple agents working together, with the Crew's process managing coordination for better teamwork and efficiency.

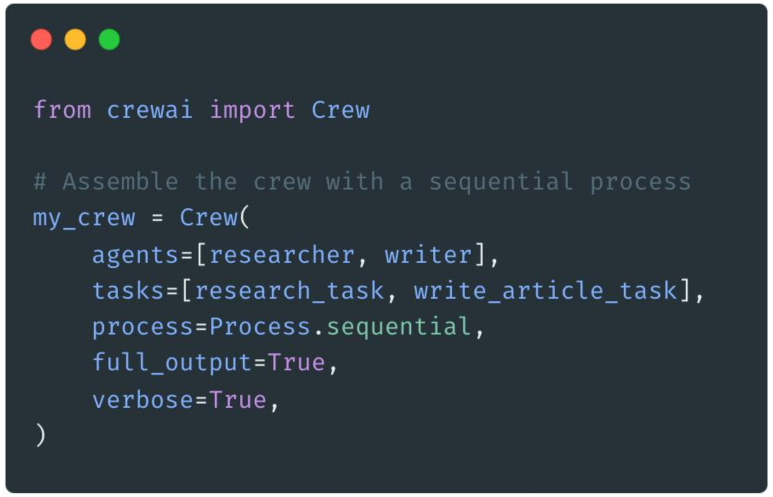


```python
from crewai import Task

task = Task(
    description='Find and summarize the latest and most relevant news on AI',
    agent=sales_agent,
    expected_output='A bullet list summary of the top 5 most important AI news',
)
```

https://docs.crewai.com/core-concepts/Tasks/

CREWS

A **crew** in crewAI represents a collaborative group of agents working together to achieve a set of tasks. Each crew defines the strategy for task execution, agent collaboration, and the overall workflow.



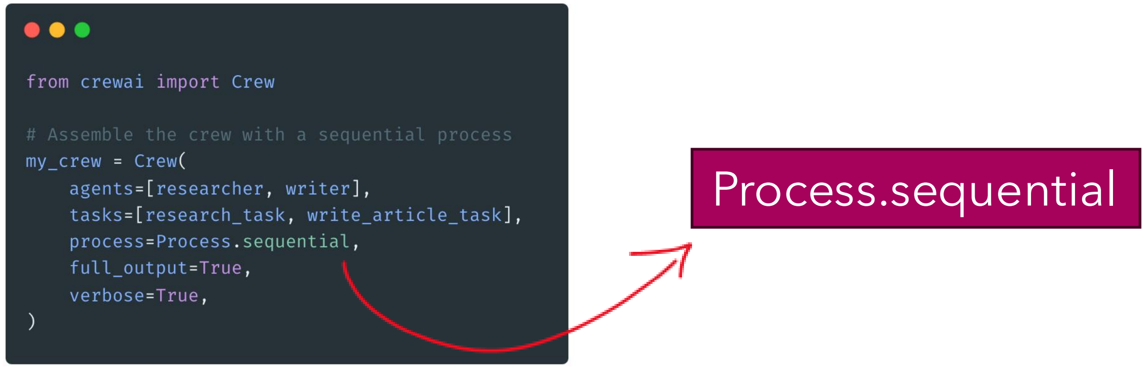
```python
from crewai import Crew

# Assemble the crew with a sequential process
my_crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, write_article_task],
    process=Process.sequential,
    full_output=True,
    verbose=True,
)
```

https://docs.crewai.com/core-concepts/Crews/

PROCESSES

In CrewAI, **processes** orchestrate the execution of tasks by agents, akin to project management in human teams. These processes ensure tasks are distributed and executed efficiently, in alignment with a predefined strategy:

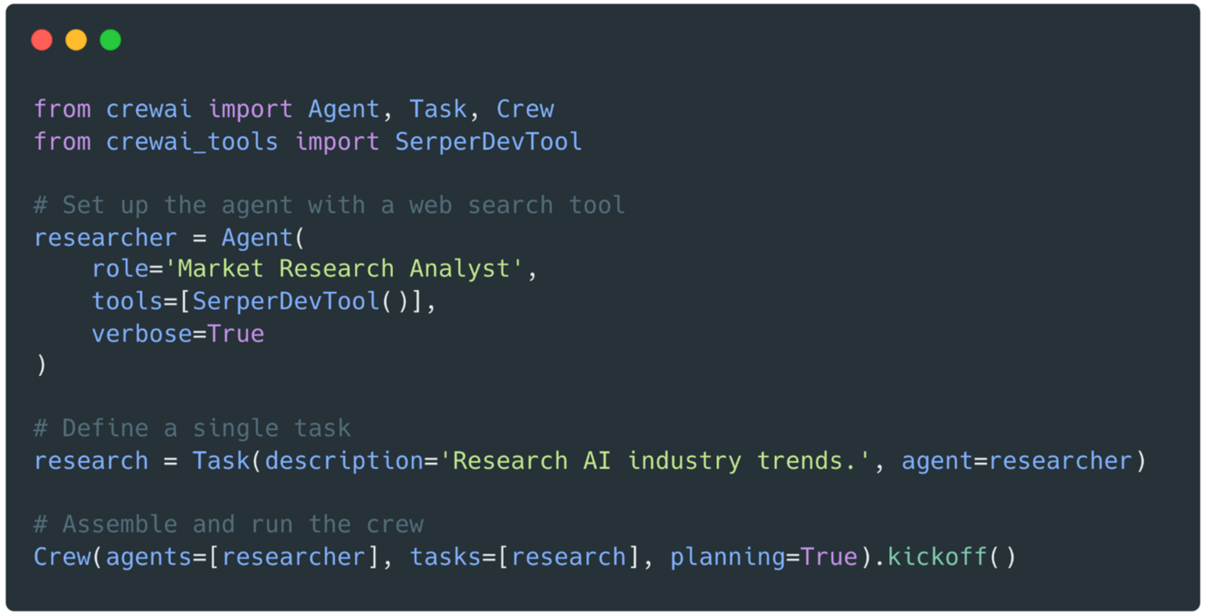
- **Sequential**: Executes tasks one after another in a specific order.
- **Hierarchical**: Organizes tasks using a chain of command, where a manager model or agent oversees and delegates tasks.
- **Consensual** (Planned): Intended for future development, this process will enable agents to collaborate and make decisions democratically on task execution.



```python
from crewai import Crew

# Assemble the crew with a sequential process
my_crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, write_article_task],
    process=Process.sequential,
    full_output=True,
    verbose=True,
)
```

TOOLS

CrewAI **tools** empower agents with capabilities like web searching, data analysis, collaboration, and task delegation, allowing users to build custom tools or leverage CrewAI and LangChain tools for seamless teamwork and complex interactions.

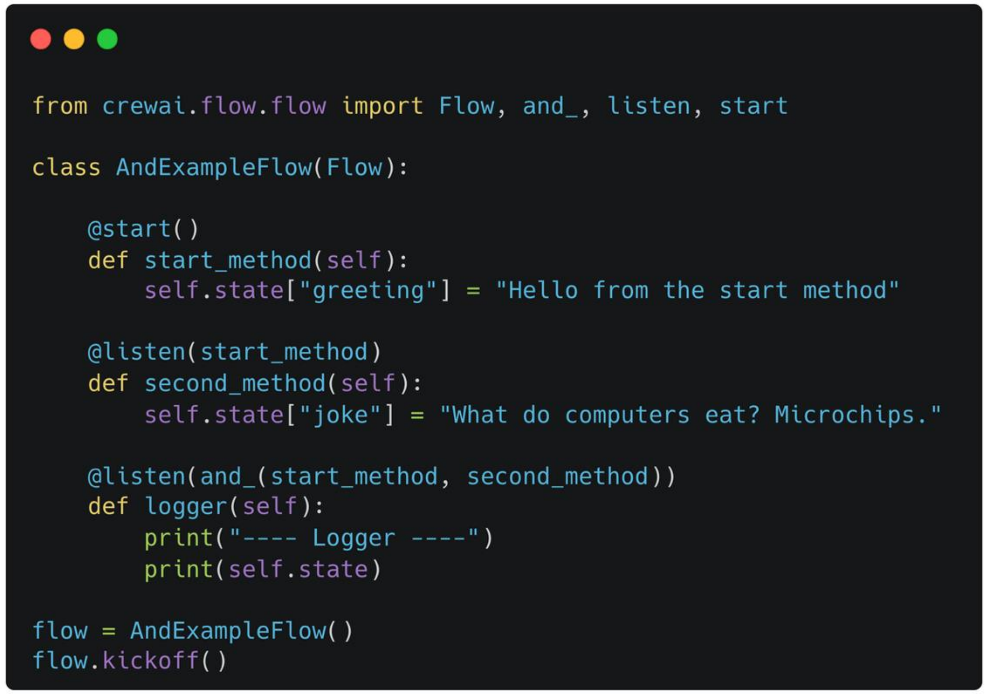


```python
from crewai import Agent, Task, Crew
from crewai_tools import SerperDevTool

# Set up the agent with a web search tool
researcher = Agent(
    role='Market Research Analyst',
    tools=[SerperDevTool()],
    verbose=True
)

# Define a single task
research = Task(description='Research AI industry trends.', agent=researcher)

# Assemble and run the crew
Crew(agents=[researcher], tasks=[research], planning=True).kickoff()
```

GUARDRAILS

With **Flow**, we can create conditional logics that help us to control the application's internal flow.

For instance, we can handle those cases where the **LLM hallucinates** and trigger the agent N number of times untill we get a "safe" output.

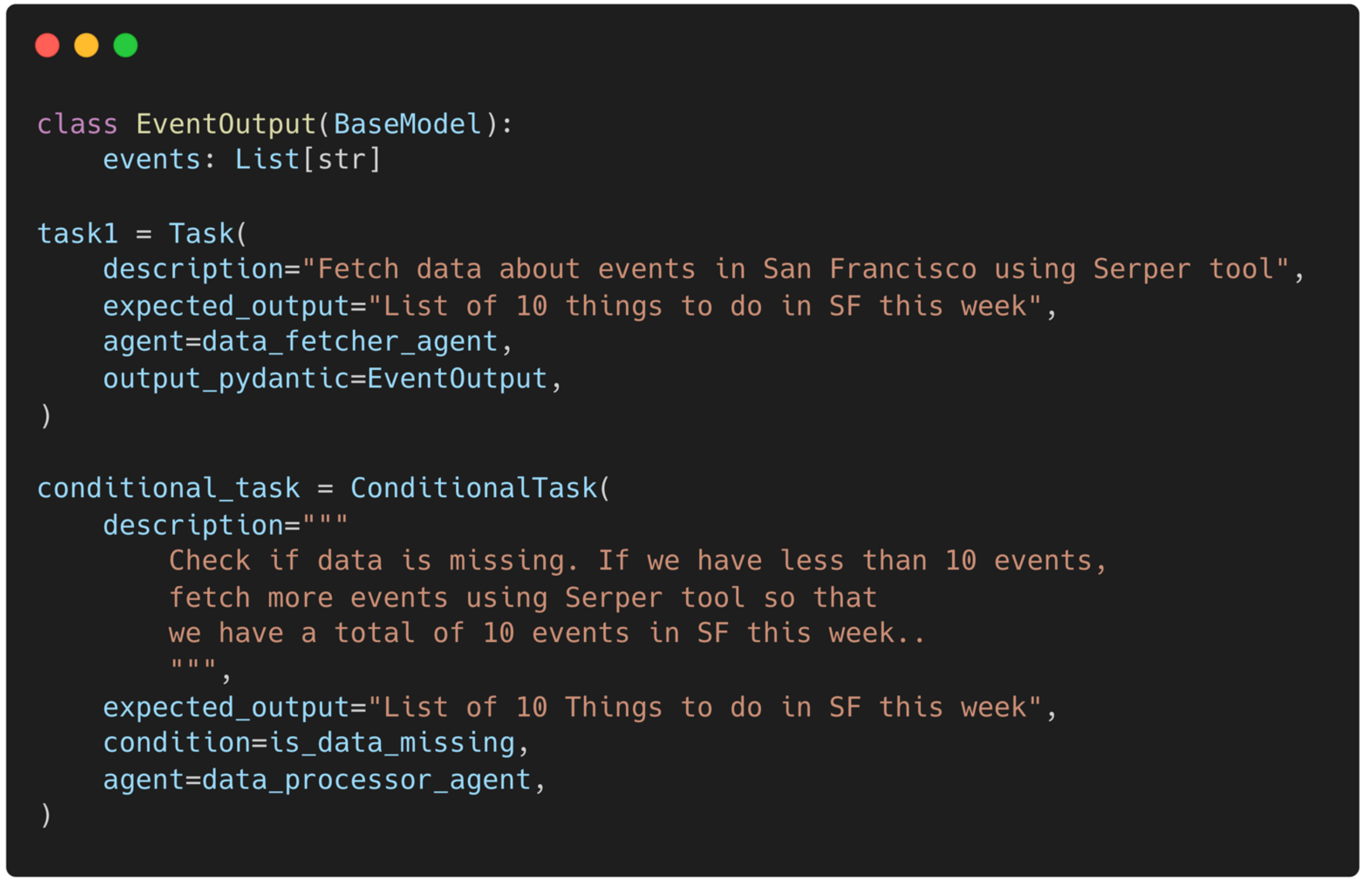


```python
from crewai.flow.flow import Flow, and_, listen, start

class AndExampleFlow(Flow):

    @start()
    def start_method(self):
        self.state["greeting"] = "Hello from the start method"

    @listen(start_method)
    def second_method(self):
        self.state["joke"] = "What do computers eat? Microchips."

    @listen(and_(start_method, second_method))
    def logger(self):
        print("---- Logger ----")
        print(self.state)

flow = AndExampleFlow()
flow.kickoff()
```

https://docs.crewai.com/concepts/flows

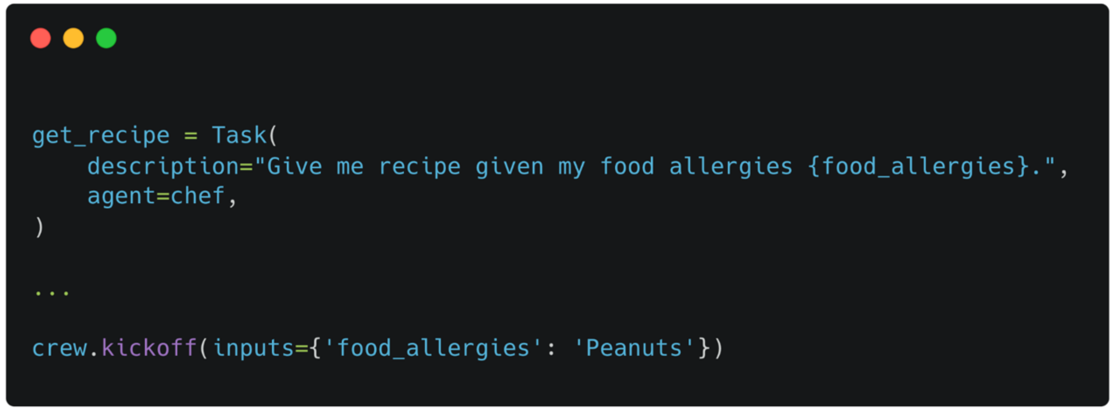
CONDITIONAL TASKS

Conditional Tasks in CrewAI let workflows adapt based on previous task results, enabling selective execution for more flexible and efficient processes.



```python
class EventOutput(BaseModel):
    events: List[str]

task1 = Task(
    description="Fetch data about events in San Francisco using Serper tool",
    expected_output="List of 10 things to do in SF this week",
    agent=data_fetcher_agent,
    output_pydantic=EventOutput,
)

conditional_task = ConditionalTask(
    description="""
        Check if data is missing. If we have less than 10 events,
        fetch more events using Serper tool so that
        we have a total of 10 events in SF this week..
        """,
    expected_output="List of 10 Things to do in SF this week",
    condition=is_data_missing,
    agent=data_processor_agent,
)
```

TASKS INPUT/OUTPUT

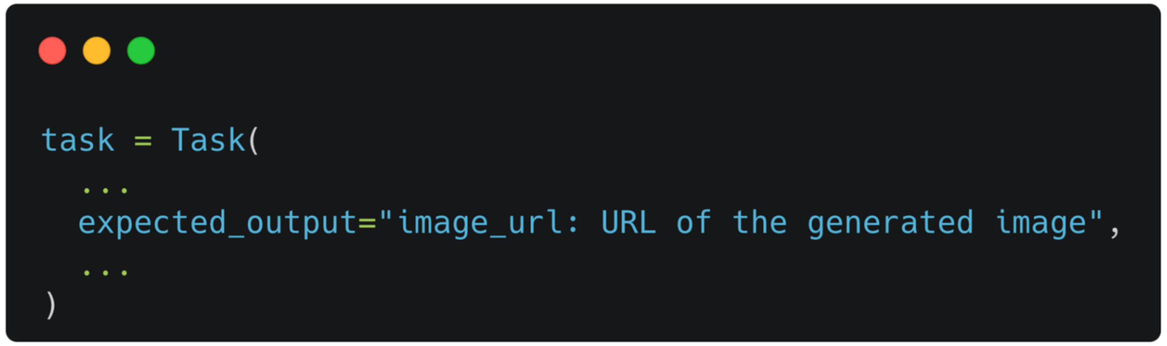


```
get_recipe = Task(
    description="Give me recipe given my food allergies {food_allergies}.",
    agent=chef,
)

...

crew.kickoff(inputs={'food_allergies': 'Peanuts'})
```

Input:

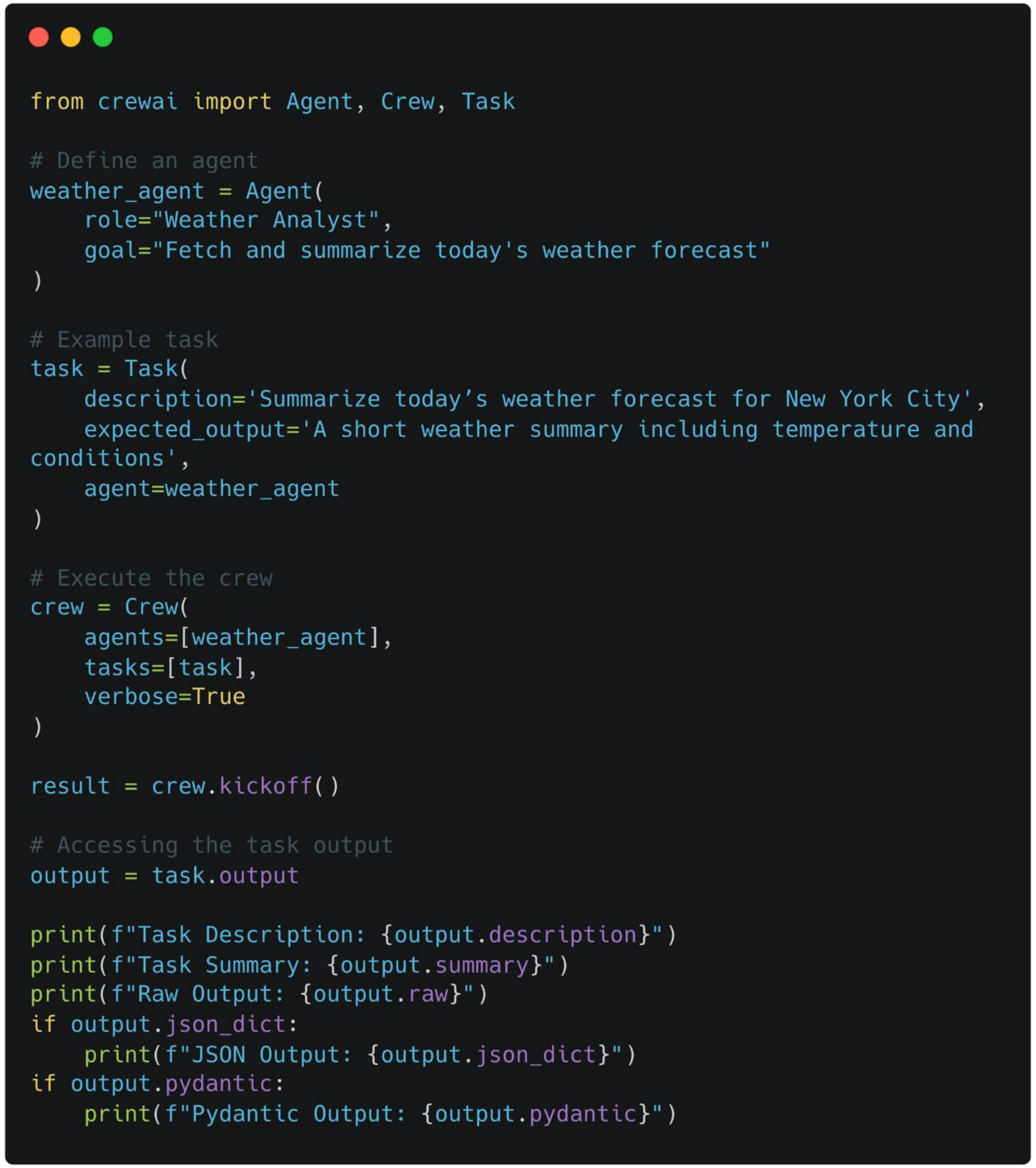
- Use *inputs* field in *kickoff()*
- Wrap your tasks into a function

The expected output is a safe yet powerfull way to make sure the next agent is able access the input data correctly.



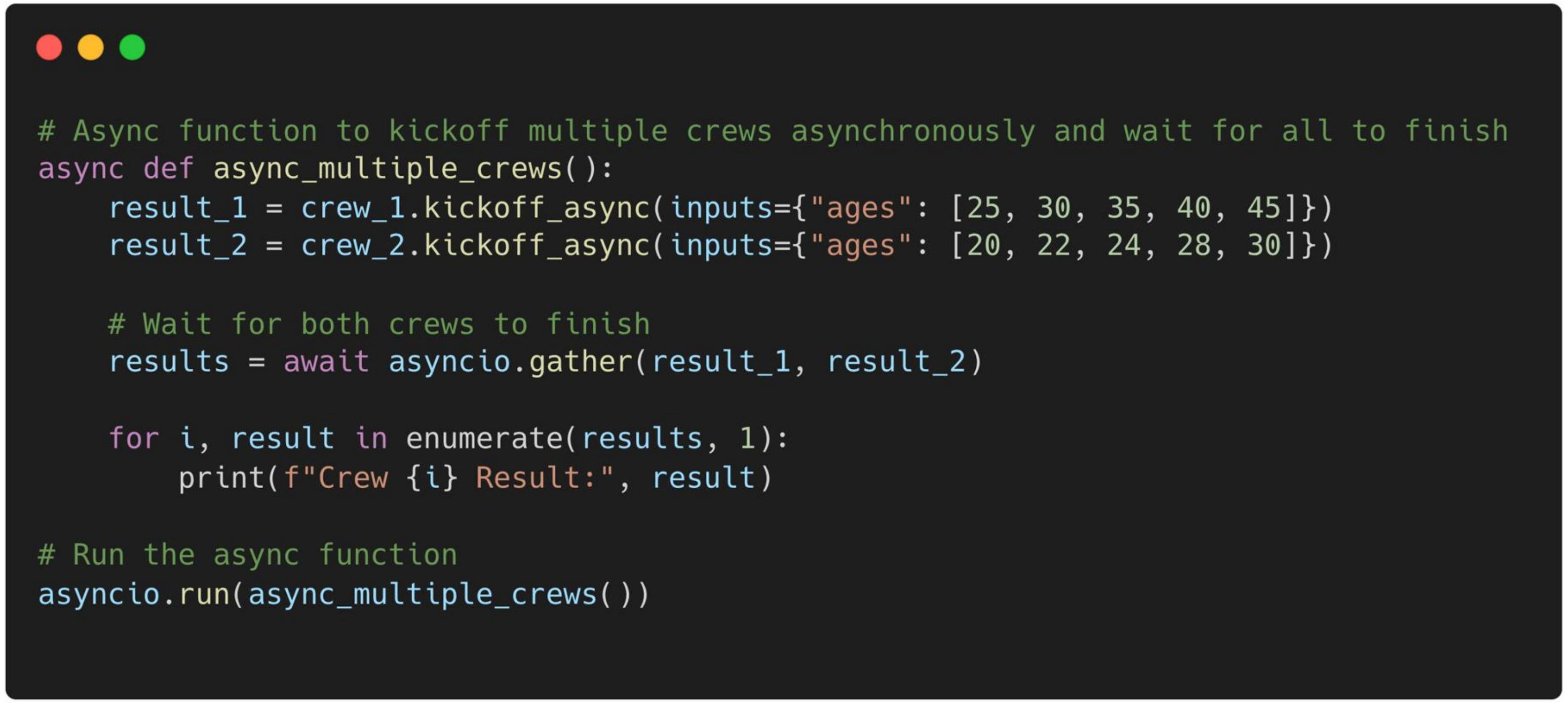
```
task = Task(
  ...
  expected_output="image_url: URL of the generated image",
  ...
)
```

ACCESS TASKS OUTPUT

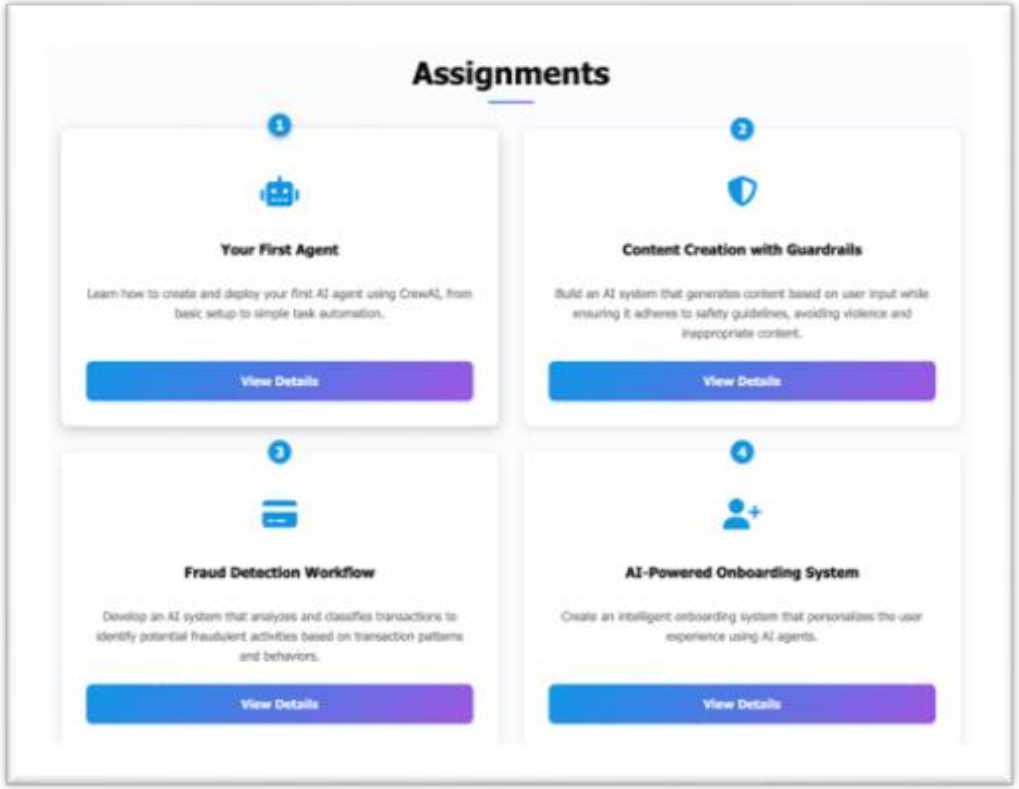


```python
from crewai import Agent, Crew, Task

# Define an agent
weather_agent = Agent(
    role="Weather Analyst",
    goal="Fetch and summarize today's weather forecast"
)

# Example task
task = Task(
    description='Summarize today’s weather forecast for New York City',
    expected_output='A short weather summary including temperature and
conditions',
    agent=weather_agent
)

# Execute the crew
crew = Crew(
    agents=[weather_agent],
    tasks=[task],
    verbose=True
)

result = crew.kickoff()

# Accessing the task output
output = task.output

print(f"Task Description: {output.description}")
print(f"Task Summary: {output.summary}")
print(f"Raw Output: {output.raw}")
if output.json_dict:
    print(f"JSON Output: {output.json_dict}")
if output.pydantic:
    print(f"Pydantic Output: {output.pydantic}")
```

ASYNCHRONOUS CREW KICKOFF



```python
# Async function to kickoff multiple crews asynchronously and wait for all to finish
async def async_multiple_crews():
    result_1 = crew_1.kickoff_async(inputs={"ages": [25, 30, 35, 40, 45]})
    result_2 = crew_2.kickoff_async(inputs={"ages": [20, 22, 24, 28, 30]})

    # Wait for both crews to finish
    results = await asyncio.gather(result_1, result_2)

    for i, result in enumerate(results, 1):
        print(f"Crew {i} Result:", result)

# Run the async function
asyncio.run(async_multiple_crews())
```



Assignments

1

Your First Agent

Learn how to create and deploy your first AI agent using CrewAI, from basic setup to simple task automation.

View Details

2

Content Creation with Guardrails

Build an AI system that generates content based on user input while ensuring it adheres to safety guidelines, avoiding violence and inappropriate content.

View Details

3

Fraud Detection Workflow

Develop an AI system that analyzes and classifies transactions to identify potential fraudulent activities based on transaction patterns and behaviors.

View Details

4

AI-Powered Onboarding System

Create an intelligent onboarding system that personalizes the user experience using AI agents.

View Details

https://pigna90.github.io/crewai-workshops/about.html

THANK YOU