Analytics Vidhya | DATAHACK SUMMIT

Workshop

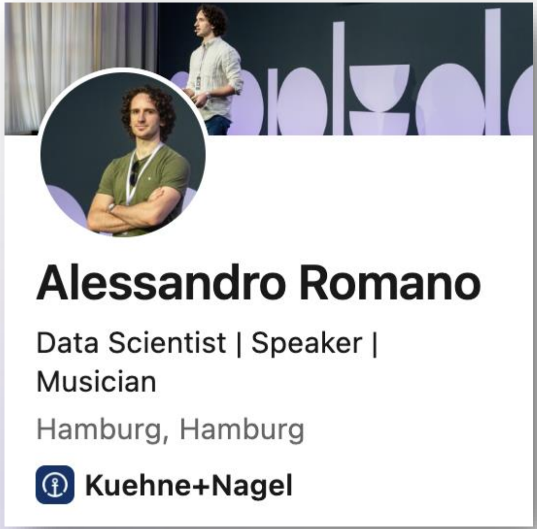
Mastering Real-World Multi-Agent Systems

Speaker

Alessandro Romano
Senior Data Scientist

Agenda

About Me

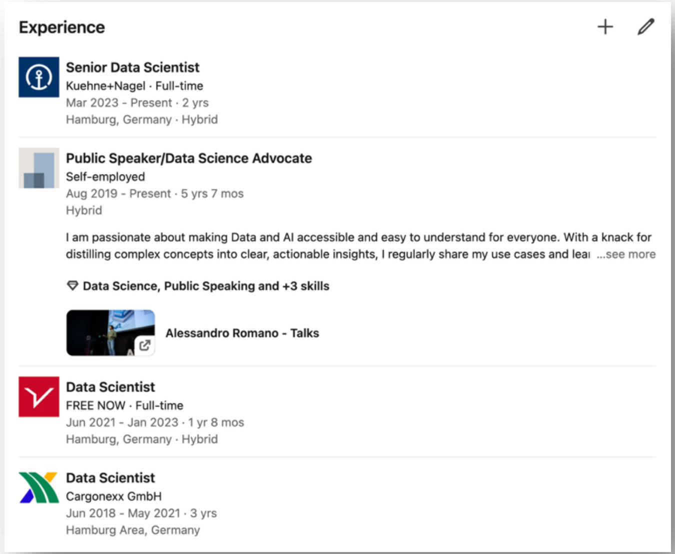


Alessandro Romano

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

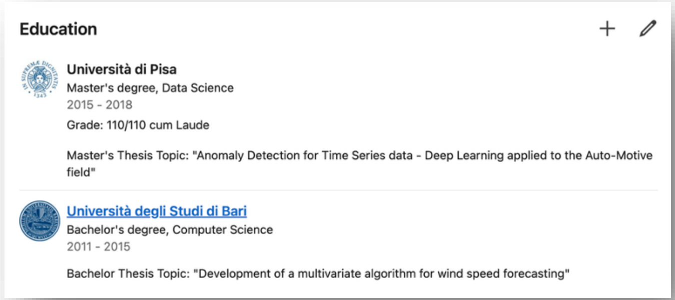
I am passionate about making Data and AI accessible and easy to understand for everyone. With a knack for distilling complex concepts into clear, actionable insights, I regularly share my use cases and lea ...see more

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

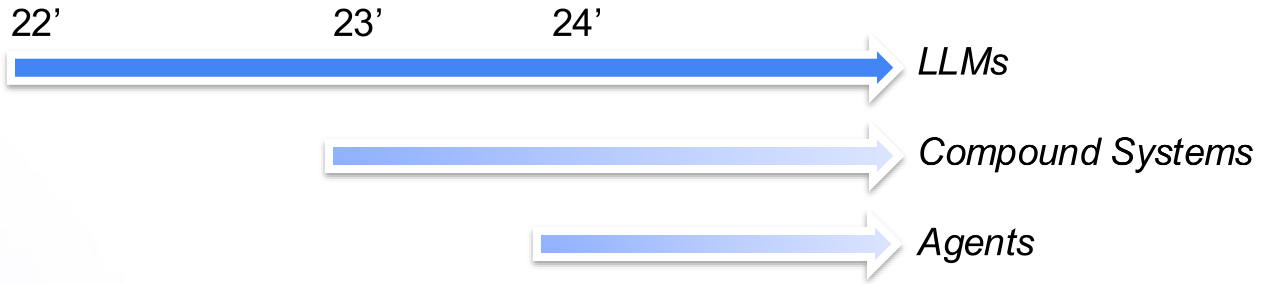
Università di Pisa
Master's degree, Data Science
2015 - 2018
Grade: 110/110 cum Laude

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

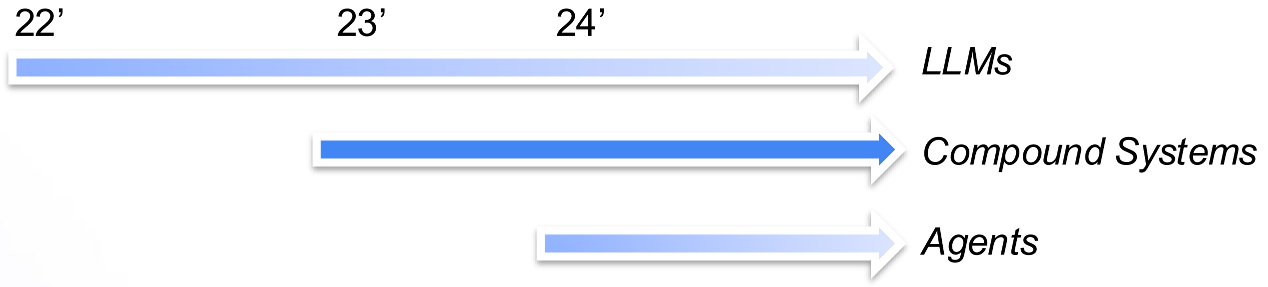
Bachelor Thesis Topic: "Development of a multivariate algorithm for wind speed forecasting"

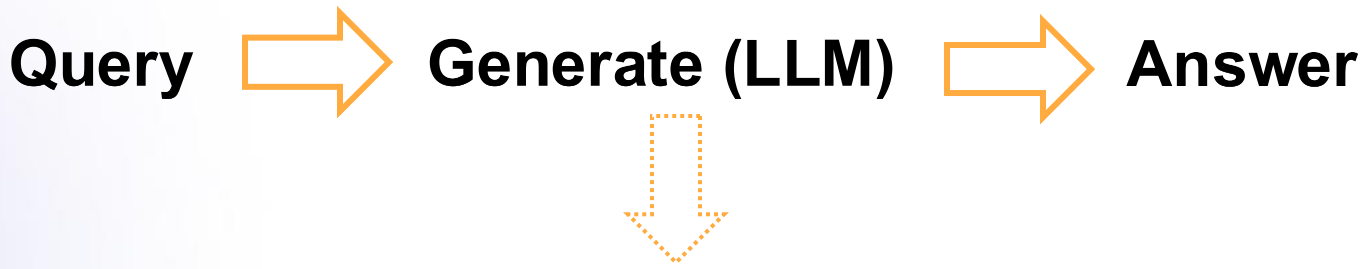
LLMs



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

Compound Systems

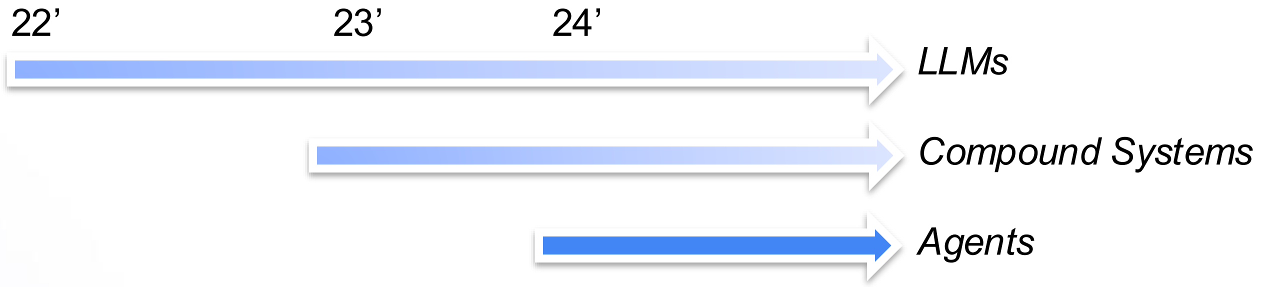


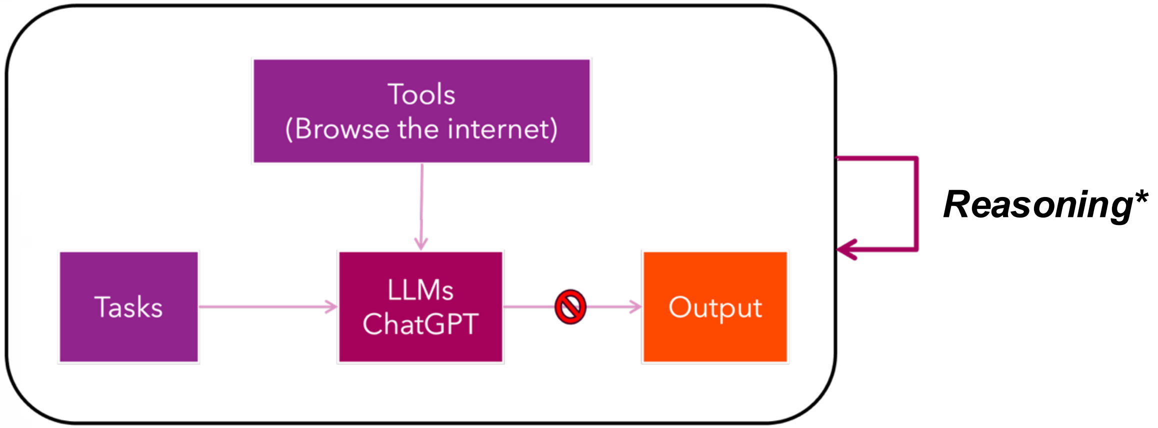


Retrieval-Augmented Generation

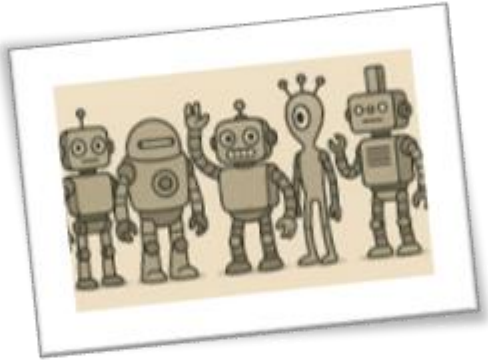
RAG

AI Agents


22'
23'
24'
LLMs
Compound Systems
Agents


Tools
(Browse the internet)
Tasks
LLMs
ChatGPT
Output
Reasoning*

The ”Agents” Family



Rule-based agents (e.g., **expert systems** in the 1980s)

Reinforcement learning agents in AI (e.g., Deep Q-learning agents, **AlphaGo**)

Robotic agents that act in the physical world

Multi-agent systems in **simulations** and games

Browser automation bots, **crawlers**, and personal assistants (like Clippy or Siri v1)

AI Agents vs. Agentic AI

***Agentic AI** refers to systems designed to act autonomously toward goals, while **AI agents** are the individual entities that perceive, reason, and take actions within those systems.*

What does »Agentic AI» Solve?

Multi-step reasoning
Example: Market research → Competitor comparison → Strategy suggestion

Dynamic workflows
Example: Customer onboarding with conditional steps and retries

Planning and goal decomposition
Example: Travel planner that books, rebooks, and adapts

Tool/API interaction
Example: SEO optimizer that audits, edits, and deploys via CMS

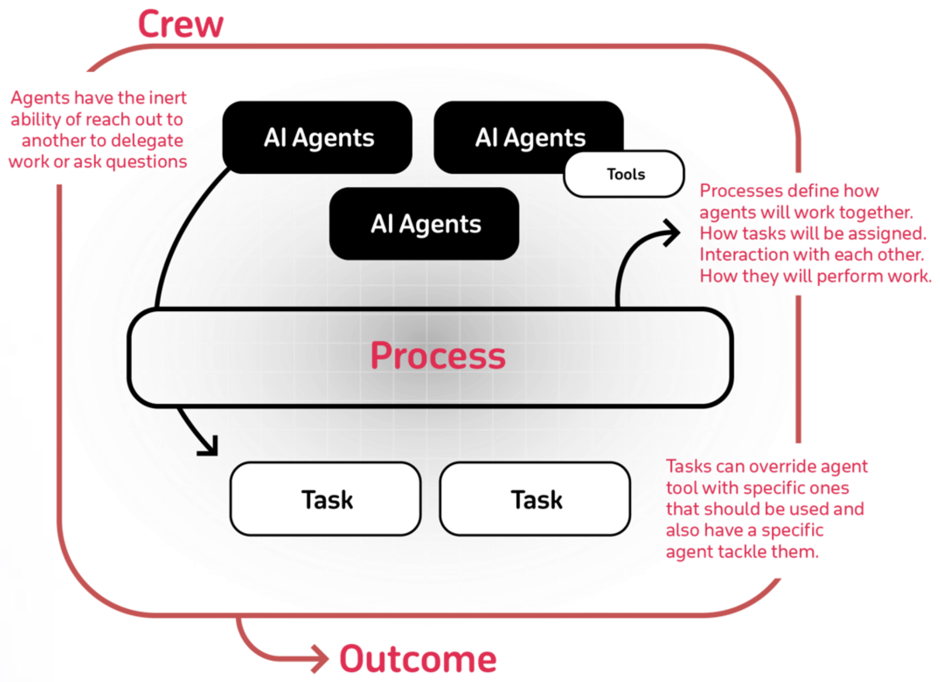
Multi-role collaboration
Example: Product strategy crew (researcher → analyst → copywriter)

CrewAI

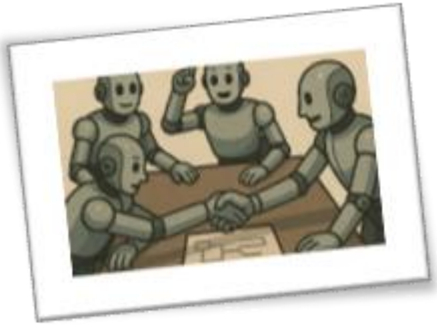
***CrewAI** is a framework for building AI agents that reason, collaborate, and act autonomously—so you can focus on outcomes, not infrastructure.*

- **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

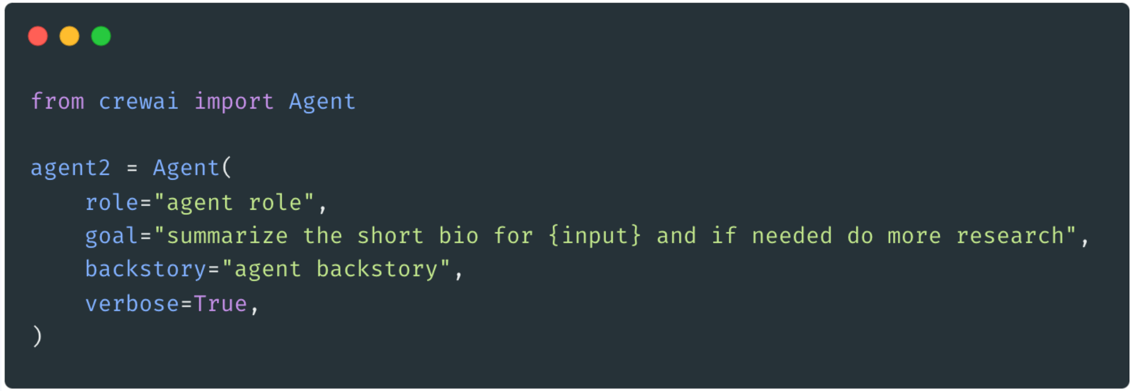
Evaluator Agent – Checks results

Agents

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

- Perform tasks
- Make decisions
- Communicate with other agents

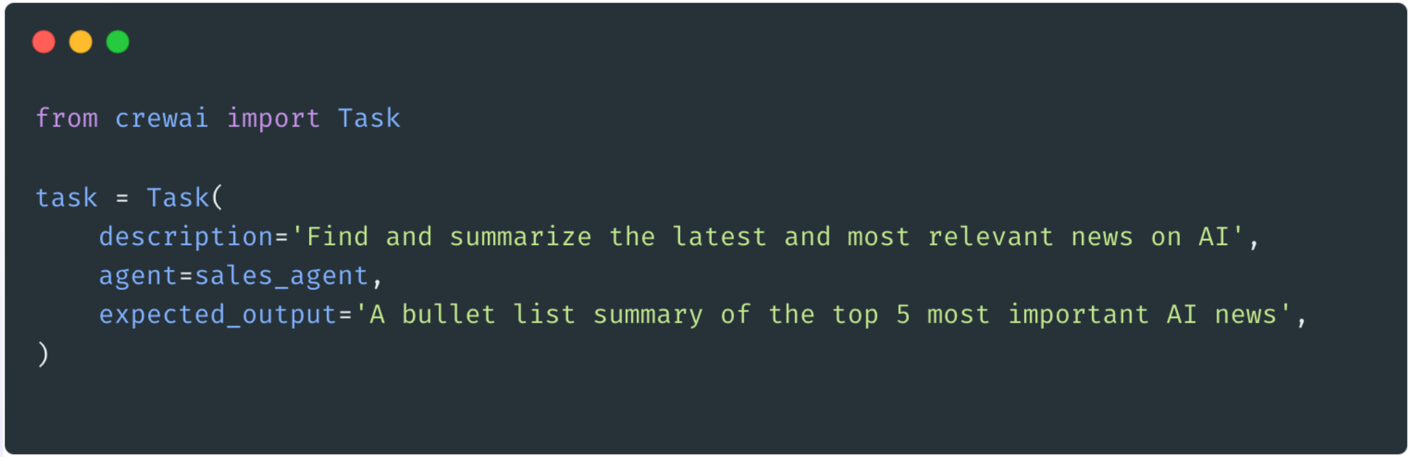
Think of an agent as a member of a team, with specific skills and a particular job to do. Agents can have different roles like *Researcher*, *Writer*, or *Customer Support*, each contributing to the overall goal of the crew.



```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,
)
```

Tasks

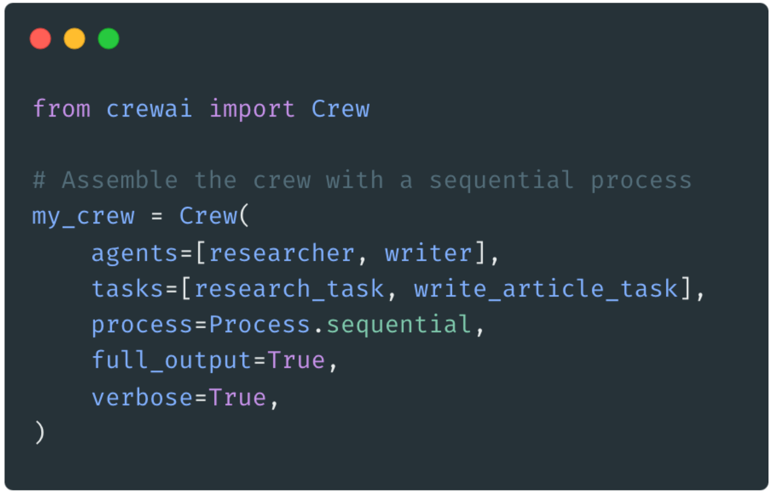
In the crewAI framework, **tasks** are assignments given to agents, including details like descriptions, responsible agents, and required tools. Tasks can be collaborative, involving multiple agents working together, with the Crew's process managing coordination for better teamwork and efficiency.



```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',
)
```

Crews

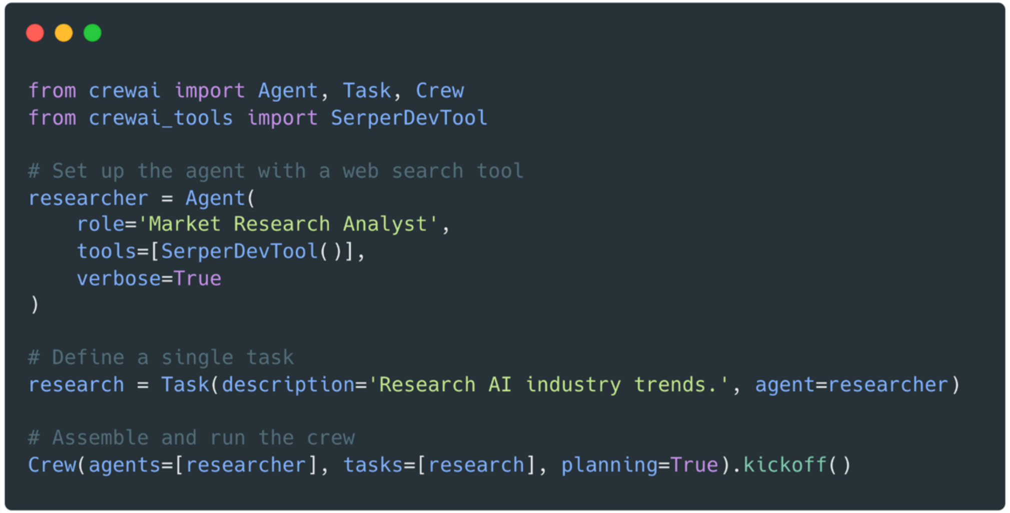
A **crew** in crewAI represents a collaborative group of agents working together to achieve a set of tasks. Each crew defines the strategy for task execution, agent collaboration, and the overall workflow.



```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,
)
```

Tools

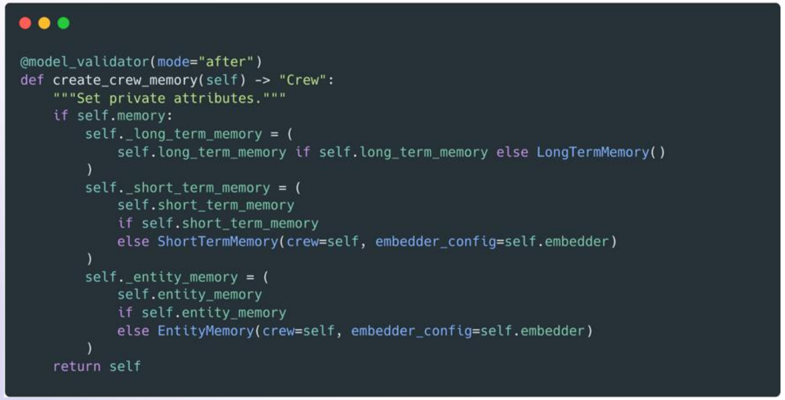
CrewAI **tools** empower agents with capabilities like web searching, data analysis, collaboration, and task delegation, allowing users to build custom tools or leverage CrewAI and LangChain tools for seamless teamwork and complex interactions.



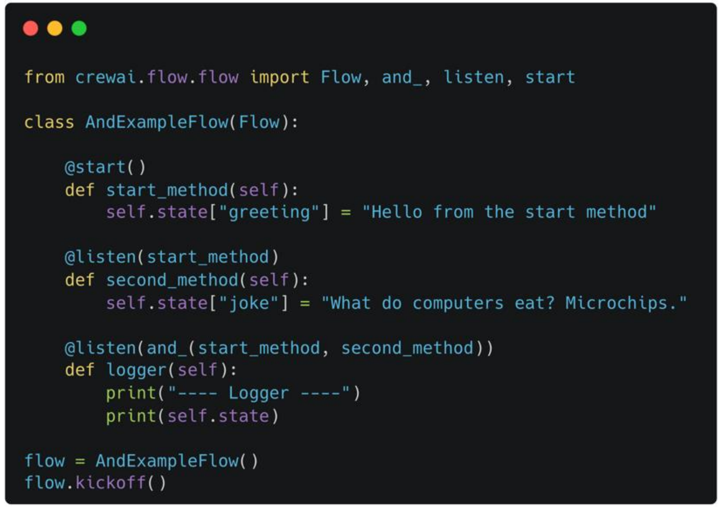
```python
from crewai import Agent, Task, Crew
from crewai_tools import SerperDevTool

# 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()
```

More Advanced Features

CrewAI **Flow** is used for creating conditional logics. Also, you have full control on what type of **memory** you want to use.



```python
@model_validator(mode="after")
def create_crew_memory(self) -> "Crew":
    """Set private attributes."""
    if self.memory:
        self._long_term_memory = (
            self.long_term_memory if self.long_term_memory else LongTermMemory()
        )
        self._short_term_memory = (
            self.short_term_memory
            if self.short_term_memory
            else ShortTermMemory(crew=self, embedder_config=self.embedder)
        )
        self._entity_memory = (
            self.entity_memory
            if self.entity_memory
            else EntityMemory(crew=self, embedder_config=self.embedder)
        )
    return self
```



```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()
```

Hand-On Guidelines

Explore the slides and the website

Feel free to ask me anything

Share your ideas with fellow attendees

Lend a hand to your neighbors

Get creative — and most importantly, have fun!

Ski Trip Use Case

The Problem

When I started skiing, I struggled with fragmented information and lacked the time and knowledge to plan effectively.

My main challenges were:

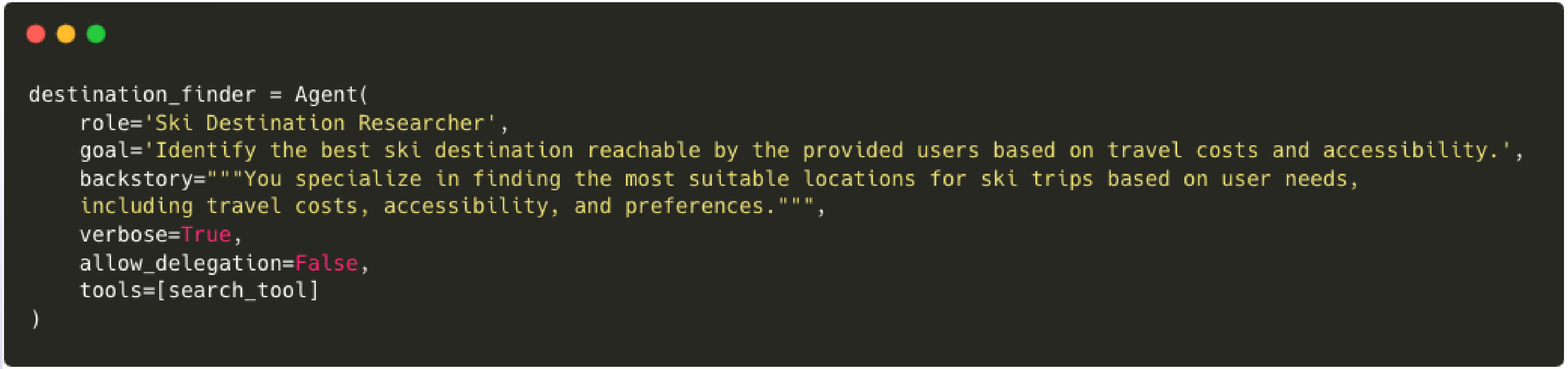
- Finding a **ski spot** accessible to at least three friends
- Renting **equipment**
- Choosing the best **travel** option
- Ensuring it was **affordable** for everyone
- Matching our different **ski levels**

<u>**Optimizing**</u> for all these factors made choosing the right destination a complex task.

Why Orchestrating AI Agents?

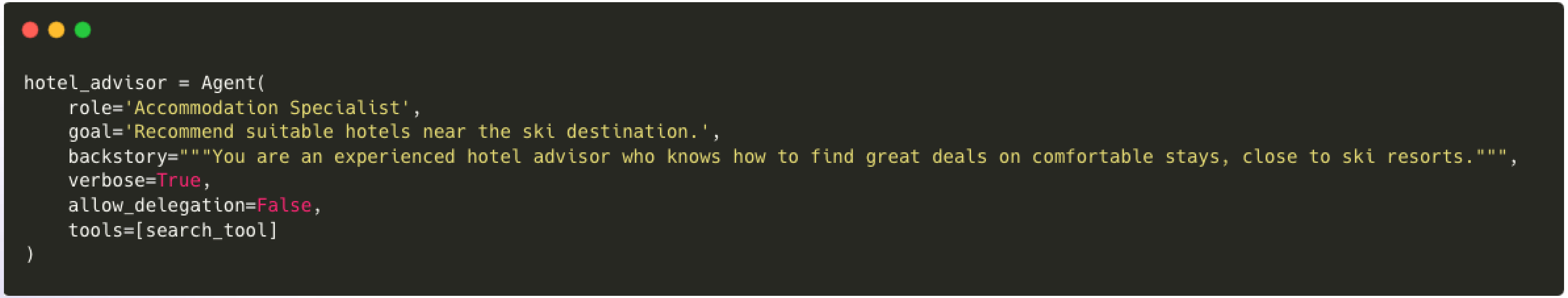
- Extensive information to process
- Optimization challenges
- Non-deterministic optimization problem
- Time-consuming internet research
- Reproducible for future trips
- Scalable solution
- I want to use LLMs for something practical

Ski Destination Researcher Agent



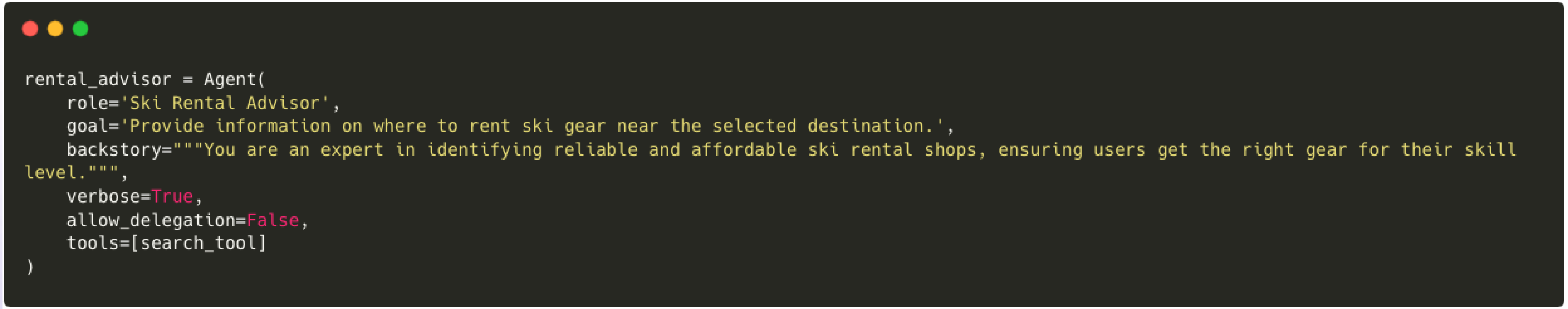
```python
destination_finder = Agent(
    role='Ski Destination Researcher',
    goal='Identify the best ski destination reachable by the provided users based on travel costs and accessibility.',
    backstory="""You specialize in finding the most suitable locations for ski trips based on user needs,
    including travel costs, accessibility, and preferences.""",
    verbose=True,
    allow_delegation=False,
    tools=[search_tool]
)
```

Accommodation Specialist Agent



```python
hotel_advisor = Agent(
    role='Accommodation Specialist',
    goal='Recommend suitable hotels near the ski destination.',
    backstory="""You are an experienced hotel advisor who knows how to find great deals on comfortable stays, close to ski resorts.""",
    verbose=True,
    allow_delegation=False,
    tools=[search_tool]
)
```

Ski Rental Advisor Agent



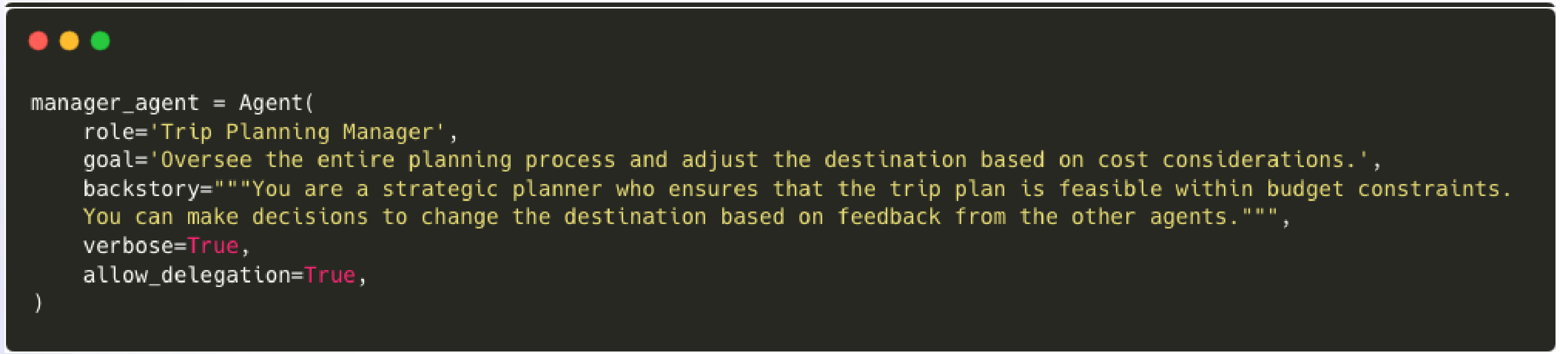
```
rental_advisor = Agent(
    role='Ski Rental Advisor',
    goal='Provide information on where to rent ski gear near the selected destination.',
    backstory="""You are an expert in identifying reliable and affordable ski rental shops, ensuring users get the right gear for their skill
level.""",
    verbose=True,
    allow_delegation=False,
    tools=[search_tool]
)
```

(Assistant) To The Regional Manager

A **manager agent ensures coordination** and adjusts decisions, like switching destinations if hotels are too costly, optimizing the trip within budget.

Why **I don't want to implement a manager** from scratch:

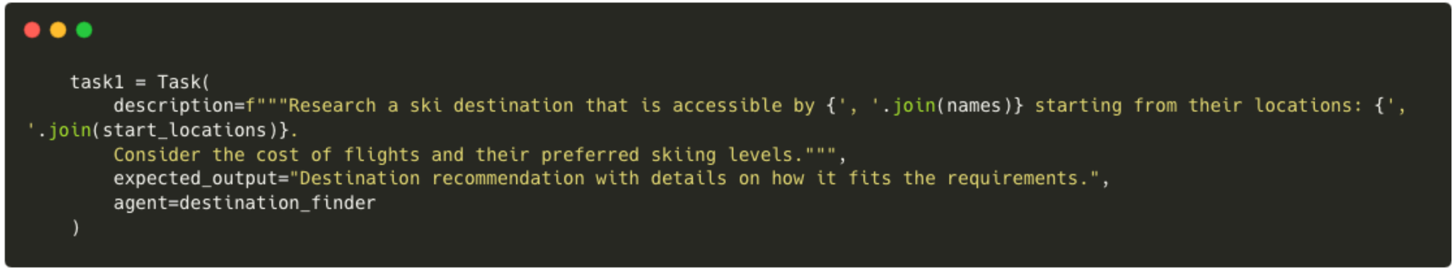
- Manual coordination and decision-making required by the manager.
- Error-prone logic using simple keyword checks for conditions.
- No built-in support for handling dependencies between agents.
- Difficult to scale and reuse for complex workflows.



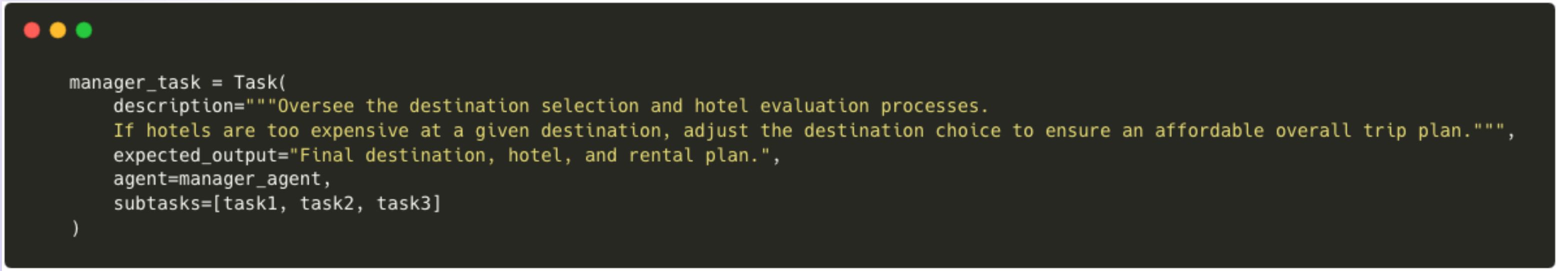
```
manager_agent = Agent(
    role='Trip Planning Manager',
    goal='Oversee the entire planning process and adjust the destination based on cost considerations.',
    backstory="""You are a strategic planner who ensures that the trip plan is feasible within budget constraints.
    You can make decisions to change the destination based on feedback from the other agents.""",
    verbose=True,
    allow_delegation=True,
)
```

Tasks

Tell the ***destination_finder*** agent what to do, focusing on what we want to optimize for.



```
    task1 = Task(
        description=f"""Research a ski destination that is accessible by {', '.join(names)} starting from their locations: {',
'.join(start_locations)}.
        Consider the cost of flights and their preferred skiing levels.""",
        expected_output="Destination recommendation with details on how it fits the requirements.",
        agent=destination_finder
    )
```

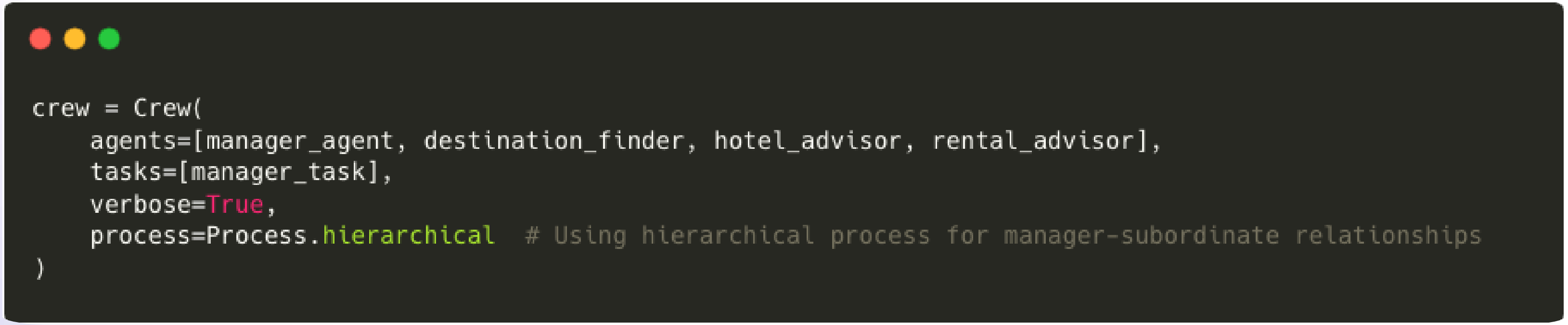


```
    manager_task = Task(
        description="""Oversee the destination selection and hotel evaluation processes.
        If hotels are too expensive at a given destination, adjust the destination choice to ensure an affordable overall trip plan.""",
        expected_output="Final destination, hotel, and rental plan.",
        agent=manager_agent,
        subtasks=[task1, task2, task3]
    )
```

Here we implement our trip **constraints.** The Manager will switch to a different location if necessary.

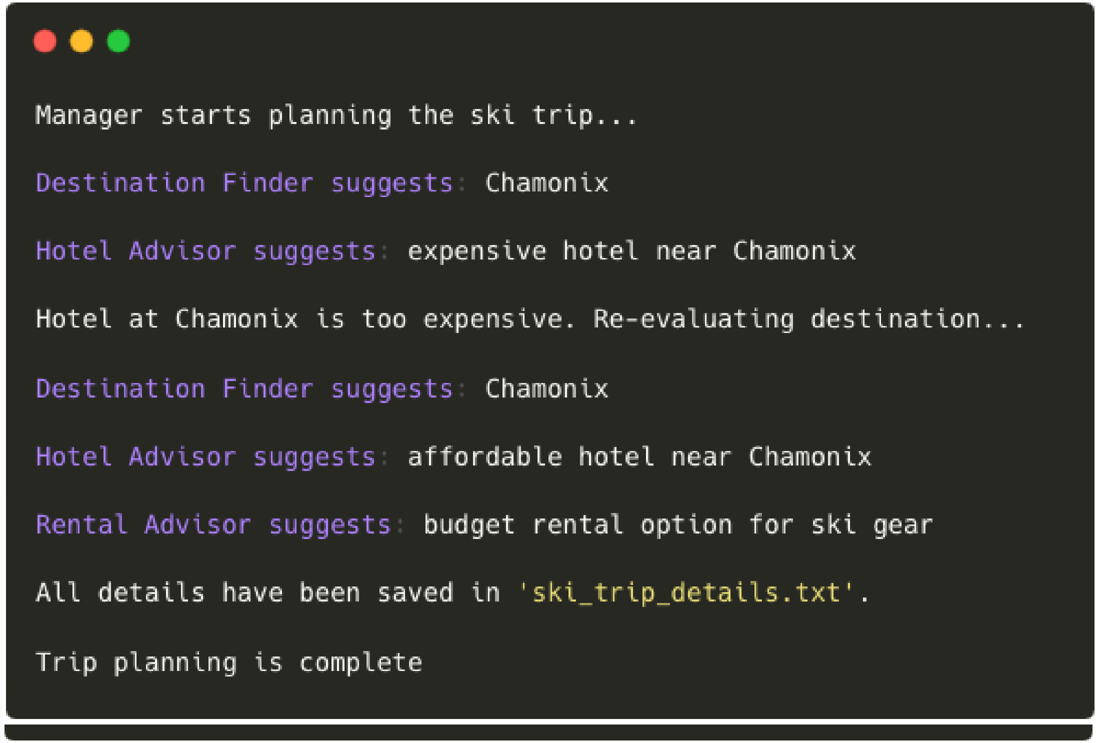
Assembling The Crew

The Crew puts together the agents and the (manager) tasks. Finally, we also decide which kind of **process** to go for.

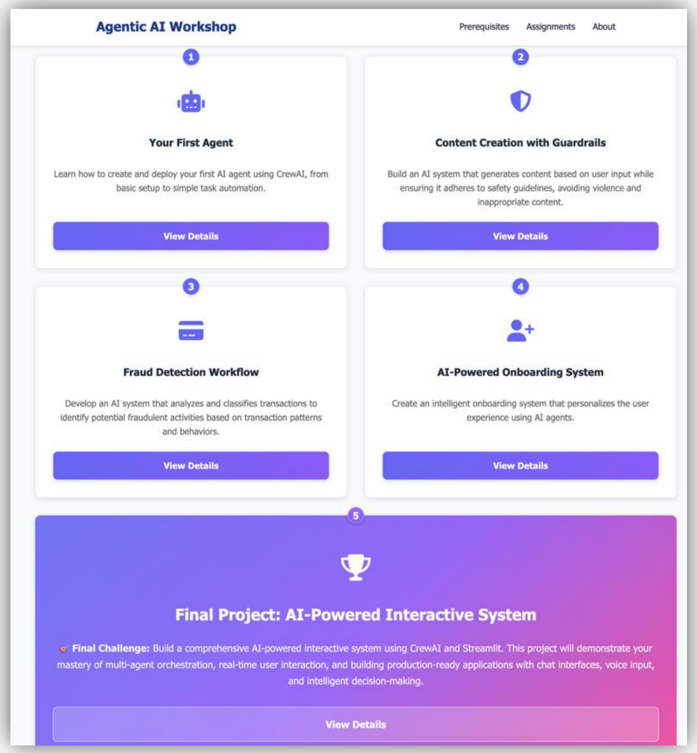


```
crew = Crew(
    agents=[manager_agent, destination_finder, hotel_advisor, rental_advisor],
    tasks=[manager_task],
    verbose=True,
    process=Process.hierarchical  # Using hierarchical process for manager-subordinate relationships
)
```

Final Output



```
Manager starts planning the ski trip...

Destination Finder suggests: Chamonix

Hotel Advisor suggests: expensive hotel near Chamonix

Hotel at Chamonix is too expensive. Re-evaluating destination...

Destination Finder suggests: Chamonix

Hotel Advisor suggests: affordable hotel near Chamonix

Rental Advisor suggests: budget rental option for ski gear

All details have been saved in 'ski_trip_details.txt'.

Trip planning is complete
```


Agentic AI Workshop

1 Your First Agent

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

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.

3 Fraud Detection Workflow

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

4 AI-Powered Onboarding System

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

5 Final Project: AI-Powered Interactive System

Final Challenge: Build a comprehensive AI-powered interactive system using CrewAI and Streamlit. This project will demonstrate your mastery of multi-agent orchestration, real-time user interaction, and building production-ready applications with chat interfaces, voice input, and intelligent decision-making.

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

Thank You