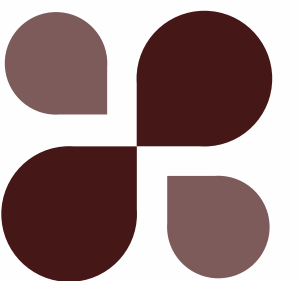




# CrewAI A-Z Course

**A beginner friendly step by step course**

*To build your own Agents*



# Introducing crewAI



## **AUTONOMOUS AI AGENT FRAMEWORK**

fostering collaborative intelligence, CrewAI empowers agents to work together seamlessly, tackling complex tasks.



# What are we gonna learn in this course?



## Core Components

Introduction to the different core components

## Core Comp Explanation

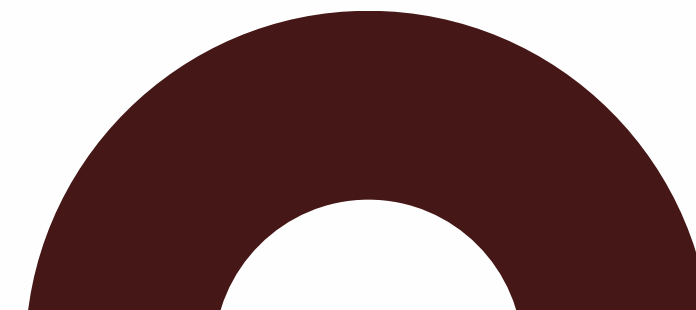
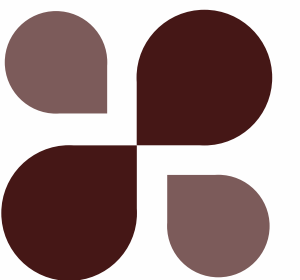
Explaining in detail about each core component including types and attributes

## The first project

A hands on first simple project on how to validate markdown reports from setup till API

## Next steps

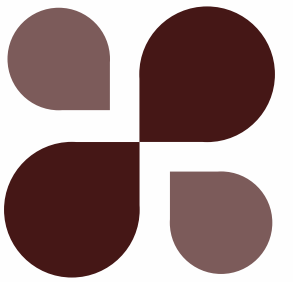
A few example projects and tips to solve real usecases





# Core Components





# Core components of CrewAI

## Agents

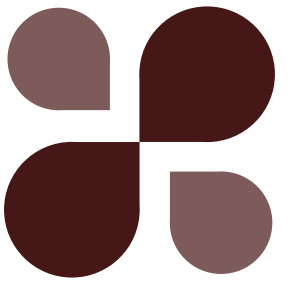
An agent is an autonomous unit programmed to Perform tasks, Make decisions, Communicate with other agents. Think of an agent as a member of a team, with specific skills and a particular job to do. Agents can have different roles like 'Researcher', 'Writer' contributing to the crew

## Tasks

In the crewAI framework, tasks are specific assignments completed by agents. They provide all necessary details for execution, such as a description, the agent responsible, required tools, and more, facilitating a wide range of action complexities.

## Crew

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.



# Core components of CrewAI

## Tools

A tool in CrewAI is a skill or function that agents can utilize to perform various actions. This includes tools from the [crewAI Toolkit](#) and [LangChain Tools](#), enabling everything from simple searches to complex interactions and effective teamwork among agents.

## Processes

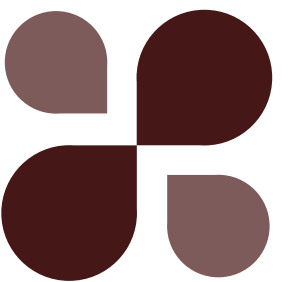
In CrewAI, processes orchestrate the execution of tasks by agents, akin to project management in human teams. These processes ensure tasks are distributed and executed efficiently, in alignment with a predefined strategy.

## Memory

The crewAI framework introduces a sophisticated memory system designed to significantly enhance the capabilities of AI agents. This system comprises short-term memory, long-term memory, entity memory, and newly identified contextual memory, each serving a unique purpose.



# Components Explanation



# Agents - Important Attributes

**Role** - Role of Agent. Shows best suited task

**Goal** - Objective of Agent to achieve. Guides decision making

**Backstory** - Context to Role and Goal. Improves decision making

**LLM** - LLM which will act as agent. **Def: GPT4**

**Tools** - Tools to help the model to accomplish the tasks. **Def: None**

**Max Iter** - Max iterations run by LLM to give the best result possible. **Def: 25**

**Max RPM** - Max reqs per minute by agent to avoid rate limit errors. **Def: None**

**Allow Delegation** - Agents can delegate tasks ask questions within. **Def: None**

*There are some others but these are important\**



# Agents Usage - Sample Code

```
# Example: Creating an agent with all attributes
from crewai import Agent

agent = Agent(
    role='Data Analyst',
    goal='Extract actionable insights',
    backstory="""You're a data analyst at a large company.
    You're responsible for analyzing data and providing insights
    to the business.
    You're currently working on a project to analyze the
    performance of our marketing campaigns."""
    tools=[my_tool1, my_tool2], # Optional, defaults to an empty list
    llm=my_llm, # Optional
    function_calling_llm=my_llm, # Optional
    max_iter=15, # Optional
    max_rpm=None, # Optional
    verbose=True, # Optional
    allow_delegation=True, # Optional
    step_callback=my_intermediate_step_callback, # Optional
    cache=True # Optional
)
```

# Tasks - Important Attributes

**Description** - A clean, concise and crisp statement on what is the task

**Agent** - Agent responsible for the task

**Expected Output** - A description on how the output should be

**Async Execution** - Runs the task asynchronously. **Def: False**

**Tools** - Tools to help the model to accomplish the tasks. **Def: None**

**Human Input** - Indication if human input is required. **Def: False**

**Output File** - Output file path to store the task output. **Def: ""**

**Callback** - A Python callable executed with output upon completion. **Def: None**

*There are some others but these are important\**

# Tasks Usage - Sample Code

```
from crewai import Task

task = Task(
    description='Find and summarize the latest AI news',
    agent=sales_agent
)
```

Creating a simple task

```
search_tool = SerperDevTool()

task = Task(
    description='Find and summarize the latest AI news',
    expected_output='A bullet list summary of the top 5 most important AI news',
    agent=research_agent,
    tools=[search_tool]
)
```

Task with tools

```
research_ai_task = Task(
    description='Find and summarize the latest AI news',
    expected_output='A bullet list summary of the top 5 most important AI news',
    async_execution=True,
    agent=research_agent,
    tools=[search_tool]
)

research_ops_task = Task(
    description='Find and summarize the latest AI Ops news',
    expected_output='A bullet list summary of the top 5 most important AI Ops news',
    async_execution=True,
    agent=research_agent,
    tools=[search_tool]
)

write_blog_task = Task(
    description="Write a full blog post about the importance of AI and its latest news",
    expected_output='Full blog post that is 4 paragraphs long',
    agent=writer_agent,
    context=[research_ai_task, research_ops_task]
)
```

Task with Other tasks

# Tools Available

**CodeDocsSearchTool**

**CSVSearchTool**

**DirectorySearchTool**

**DOCXSearchTool**

**DirectoryReadTool**

**FileReadTool**

**GithubSearchTool**

**SerperDevTool**

**TXTSearchTool**

**JSONSearchTool**

**MDXSearchTool**

**PDFSearchTool**

**PGSearchTool**

**RagTool**

**ScrapeElementFromWebsiteTool**

**ScrapeWebsiteTool**

**WebsiteSearchTool**

**XMLSearchTool**

**YoutubeChannelSearchTool**

**YoutubeVideoSearchTool**

# Tools Usage - Sample Code

```
# Instantiate tools
docs_tool = DirectoryReadTool(directory='./blog-posts')
file_tool = FileReadTool()
search_tool = SerperDevTool()
web_rag_tool = WebsiteSearchTool()

# Create agents
researcher = Agent(
    role='Market Research Analyst',
    goal='Provide up-to-date market analysis of the AI industry',
    backstory='An expert analyst with a keen eye for market trends.',
    tools=[search_tool, web_rag_tool],
    verbose=True
)
```

# Custom Tool - Sample Code

## Subclassing `BaseTool`

```
from crewai_tools import BaseTool

class MyCustomTool(BaseTool):
    name: str = "Name of my tool"
    description: str = "Clear description for what this tool is useful for, you agent will need this information"

    def _run(self, argument: str) -> str:
        # Implementation goes here
        return "Result from custom tool"
```

## Utilizing the `tool` Decorator

```
from crewai_tools import tool
@tool("Name of my tool")
def my_tool(question: str) -> str:
    """Clear description for what this tool is useful for, you agent will need this information to use it."""
    # Function logic here
    return "Result from your custom tool"
```

# Processes - Types

## Sequential Process

This method mirrors dynamic team workflows, progressing through tasks in a thoughtful and systematic manner. Task execution follows the predefined order in the task list, with the output of one task serving as context for the next.

## Hierarchial Process

Organizes tasks in a managerial hierarchy, where tasks are delegated and executed based on a structured chain of command. A manager language model (manager\_llm) must be specified in the crew to enable the hierarchical process

*Consensual Process Under Implementation Still*

# Processes Usage - Sample Code

```
from crewai import Crew
from crewai.process import Process
from langchain_openai import ChatOpenAI

# Example: Creating a crew with a sequential process
crew = Crew(
    agents=my_agents,
    tasks=my_tasks,
    process=Process.sequential
)

# Example: Creating a crew with a hierarchical process
# Ensure to provide a manager_llm
crew = Crew(
    agents=my_agents,
    tasks=my_tasks,
    process=Process.hierarchical,
    manager_llm=ChatOpenAI(model="gpt-4")
)
```



# Crew - Important Attributes

**Tasks** - List of tasks

**Agents** - List of agents

**Process** - Process flow.

**Manager LLM** - If hierarchical a manager LLM to monitor process.

**Verbose** - Verbosity level of logging

**Memory** - Utilized for storing execution memories (short, long-term, entity).

**Max RPM** - Max reqs per minute by agent to avoid rate limit errors.

**Full Output** - Should crew return full output or final alone.

*There are some others but these are important\**

# Crew Usage - Sample Code

```
my_crew = Crew(  
    agents=[researcher, writer],  
    tasks=[research_task, write_article_task],  
    process=Process.sequential,  
    full_output=True,  
    verbose=True,  
)
```

Creating a simple crew

```
# Start the crew's task execution  
result = my_crew.kickoff()  
print(result)
```

Starting Crew

```
researcher = Agent(  
    role='Senior Research Analyst',  
    goal='Discover innovative AI technologies',  
    tools=[DuckDuckGoSearchRun()]  
)  
  
writer = Agent(  
    role='Content Writer',  
    goal='Write engaging articles on AI discoveries',  
    verbose=True  
)  
  
# Create tasks for the agents  
research_task = Task(  
    description='Identify breakthrough AI technologies',  
    agent=researcher  
)  
write_article_task = Task(  
    description='Draft an article on the latest AI technologies',  
    agent=writer  
)  
  
# 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,  
)
```

Creating a simple crew whole code

# Memory - Types

## Short-Term Memory

Temporarily stores recent interactions and outcomes, enabling agents to recall and utilize information relevant to their current context.

## Long-Term Memory

Preserves valuable insights and learnings from past executions, allowing agents to build and refine their knowledge over time.

## Entity Memory

Captures and organizes information about entities (people, places, concepts) encountered during tasks, facilitating deeper understanding and relationship mapping.

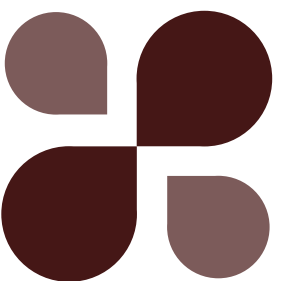
# Memory Usage - Sample Code

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

# Assemble your crew with memory capabilities
my_crew = Crew(
    agents=[...],
    tasks=[...],
    process=Process.sequential,
    memory=True,
    verbose=True
)
```

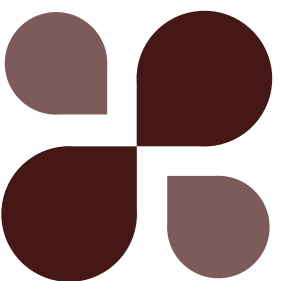


# The first project Markdown Validation



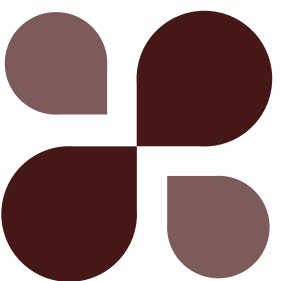


# Next Steps



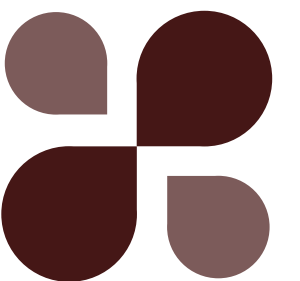
# Some advanced projects...

<https://github.com/joaomdmoura/crewAI-examples/tree/main>





**Some tips...**







**Thank you!**

