explaining each part in terms of **CrewAI functionalities** and how it leverages the framework to build a multi-agent system for researching and writing about "AI in Healthcare." I’ll use the corrected version from my previous response (with OpenAI as the default LLM) and focus on how each line ties into CrewAI’s features—agents, tasks, tools, and crew orchestration.

**Code with Line-by-Line Explanation - Furquan**

python

.

*# Import necessary CrewAI classes and tools*

from crewai import Agent, Task, Crew

from crewai\_tools import SerperDevTool

from dotenv import load\_dotenv

import os

* **Functionality**: Imports core CrewAI components:
  + Agent: Defines autonomous entities with roles, goals, and behaviors.
  + Task: Specifies what each agent should do and what output to produce.
  + Crew: Orchestrates multiple agents and tasks into a collaborative workflow.
  + SerperDevTool: A CrewAI tool for web search, integrating with the Serper API.
* **Purpose**: Sets up the foundation for building a multi-agent system with external tool support.

python

.

*# Load environment variables*

load\_dotenv()

if not os.getenv("OPENAI\_API\_KEY"):

raise ValueError("OPENAI\_API\_KEY not found in .env")

if not os.getenv("SERPER\_API\_KEY"):

raise ValueError("SERPER\_API\_KEY not found in .env")

* **Functionality**: Uses dotenv to load API keys from a .env file (e.g., OPENAI\_API\_KEY for the LLM, SERPER\_API\_KEY for the search tool).
* **CrewAI Context**: CrewAI relies on LLMs (defaulting to OpenAI unless specified) and tools, which need API keys. This ensures the environment is configured before agents run.
* **Purpose**: Validates that required credentials are available, preventing runtime errors.

python

.

*# Topic*

topic = "AI in Healthcare"

* **Functionality**: Defines a variable topic to parameterize the agents’ goals and tasks.
* **CrewAI Context**: CrewAI supports dynamic inputs, which can be passed to tasks via the Crew.kickoff() method (though here it’s hardcoded for simplicity).
* **Purpose**: Sets the focus of the research and writing process.

python

.

*# Tool*

search\_tool = SerperDevTool()

* **Functionality**: Instantiates SerperDevTool, a CrewAI-provided tool that enables web searches via the Serper API.
* **CrewAI Context**: Tools extend agent capabilities beyond LLM reasoning (e.g., fetching real-time data). SerperDevTool queries the web and returns results, which the agent can process.
* **Purpose**: Equips the research agent with the ability to find up-to-date information online, relying on SERPER\_API\_KEY from .env.

python

.

*# Agent 1: Senior Research Analyst*

senior\_research\_analyst = Agent(

role="Senior Research Analyst",

goal=f"Research, analyze, and synthesize comprehensive information on {topic} from reliable web sources",

backstory="You're an expert research analyst with advanced web research skills. "

"You excel at finding, analyzing, and synthesizing information from "

"across the internet using search tools. You're skilled at "

"distinguishing reliable sources from unreliable ones, "

"fact-checking, cross-referencing information, and "

"identifying key patterns and insights. You provide "

"well-organized research briefs with proper citations "

"and source verification. Your analysis includes both "

"raw data and interpreted insights, making complex "

"information accessible and actionable.",

verbose=2,

allow\_delegation=False,

tools=[search\_tool],

)

* **Functionality**: Creates an Agent instance with specific attributes:
  + role: Defines the agent’s identity (displayed in logs/output).
  + goal: Sets the agent’s objective, guiding its behavior (here, researching topic).
  + backstory: Provides context to the LLM, shaping its tone and approach (e.g., expert analyst).
  + verbose=2: Enables detailed logging (level 2 = step-by-step execution), a CrewAI feature for debugging.
  + allow\_delegation=False: Prevents this agent from passing tasks to others, ensuring it completes its work solo (CrewAI supports delegation between agents).
  + tools=[search\_tool]: Assigns the web search capability, allowing the agent to fetch external data.
* **CrewAI Context**: Agents are the workers in CrewAI, powered by an LLM (defaults to OpenAI’s GPT if OPENAI\_API\_KEY is set). The goal and backstory influence how the LLM interprets and executes tasks.
* **Purpose**: Configures a specialized researcher to gather and analyze data on "AI in Healthcare."

python

.

*# Agent 2: Content Writer*

content\_writer = Agent(

role="Content Writer",

goal="Transform research findings into engaging blog posts while maintaining accuracy",

backstory="You're a skilled content writer specialized in creating "

"engaging, accessible content from technical research. "

"You work closely with the Senior Research Analyst and excel at maintaining the perfect "

"balance between informative and entertaining writing, "

"while ensuring all facts and citations from the research "

"are properly incorporated. You have a talent for making "

"complex topics approachable without oversimplifying them.",

verbose=2,

allow\_delegation=False,

)

* **Functionality**: Defines a second Agent with:
  + role, goal, backstory: Similar to the first agent, but focused on writing rather than research.
  + verbose=2: Detailed logs for transparency.
  + allow\_delegation=False: Keeps this agent independent.
  + No tools: Relies solely on the LLM and the researcher’s output.
* **CrewAI Context**: This agent showcases CrewAI’s ability to chain agents—its task depends on the researcher’s output, demonstrating sequential workflows.
* **Purpose**: Prepares an agent to craft a blog post from research data.

python

.

*# Task 1: Research*

research\_task = Task(

description=f"""

1. Conduct comprehensive research on {topic} including:

- Recent developments and news

- Key industry trends and innovations

- Expert opinions and analyses

- Statistical data and market insights

2. Evaluate source credibility and fact-check all information

3. Organize findings into a structured research brief

4. Include all relevant citations and sources

""",

expected\_output="""A detailed research report containing:

- Executive summary of key findings

- Comprehensive analysis of current trends and developments

- List of verified facts and statistics

- All citations and links to original sources

- Clear categorization of main themes and patterns

Please format with clear sections and bullet points for easy reference.""",

agent=senior\_research\_analyst

)

* **Functionality**: Creates a Task instance:
  + description: Detailed instructions for the agent, using an f-string to insert topic.
  + expected\_output: Specifies the desired format and content, guiding the agent’s deliverable.
  + agent: Assigns this task to senior\_research\_analyst.
* **CrewAI Context**: Tasks are the building blocks of work in CrewAI. The description acts as a prompt to the LLM, while expected\_output sets a quality standard. The agent uses its tools (e.g., SerperDevTool) to fulfill this.
* **Purpose**: Defines what the researcher must produce—a structured brief on "AI in Healthcare."

python

.

*# Task 2: Content Writing*

writing\_task = Task(

description=f"""

Using the research brief provided, create an engaging blog post that:

1. Transforms technical information into accessible content

2. Maintains all factual accuracy and citations from the research

3. Includes:

- Attention-grabbing introduction

- Well-structured body sections with clear headings

- Compelling conclusion

4. Preserves all source citations in [Source: URL] format

5. Includes a References section at the end

""",

expected\_output="""A polished blog post in markdown format that:

- Engages readers while maintaining accuracy

- Contains properly structured sections

- Includes inline citations hyperlinked to the original source URL

- Presents information in an accessible yet informative way

- Follows proper markdown formatting, use H1 for the title and H3 for subsections""",

agent=content\_writer

)

* **Functionality**: Defines a second Task:
  + description: Instructs the agent to use the researcher’s output, with specific writing guidelines.
  + expected\_output: Details the blog post format, including markdown syntax.
  + agent: Links to content\_writer.
* **CrewAI Context**: This task demonstrates CrewAI’s ability to pass results between tasks implicitly—the writer uses the researcher’s brief without explicit coding, thanks to the Crew orchestration.
* **Purpose**: Outlines the creation of a reader-friendly blog post from raw research.

python

.

*# Crew Setup*

crew = Crew(

agents=[senior\_research\_analyst, content\_writer],

tasks=[research\_task, writing\_task],

verbose=2

)

* **Functionality**: Instantiates a Crew object:
  + agents: List of agents involved in the workflow.
  + tasks: List of tasks to execute in sequence (research → writing).
  + verbose=2: Provides detailed execution logs.
* **CrewAI Context**: The Crew class is the heart of CrewAI, managing agent collaboration and task execution. It ensures tasks are completed in order and results are passed along (e.g., research output feeds into writing).
* **Purpose**: Ties the agents and tasks into a cohesive unit for execution.

python

.

*# Run the Crew*

result = crew.kickoff()

print("Final Result:\n", result)

* **Functionality**:
  + kickoff(): Starts the crew’s workflow, executing tasks sequentially with the assigned agents.
  + result: Captures the final output (typically the last task’s result—in this case, the blog post).
  + print: Displays the result.
* **CrewAI Context**: kickoff() triggers the agentic process, leveraging the LLM and tools. The verbose logs show each step (e.g., search queries, reasoning, writing).
* **Purpose**: Executes the entire pipeline and shows the final blog post.

**How CrewAI Functionalities Tie Together**

1. **Agents**: Act as independent workers with specialized skills (research, writing), powered by an LLM and tools.
2. **Tasks**: Define the work, serving as prompts and output templates for the LLM within each agent.
3. **Tools**: Extend agents’ capabilities (e.g., SerperDevTool for web data), integrating external resources.
4. **Crew**: Orchestrates the process, ensuring tasks flow logically and agents collaborate implicitly by sharing outputs.

**Example Workflow Execution**

* **Researcher**: Uses SerperDevTool to search "AI in Healthcare trends," processes results with GPT-4o, and produces a brief (e.g., trends, stats, citations).
* **Writer**: Takes the brief, transforms it into a markdown blog post (e.g., # AI in Healthcare\n### Trends\n...), and outputs the final result.
* **Logs**: With verbose=2, you’ll see each agent’s steps (e.g., search queries, reasoning traces).

**Suggestions for Enhancement**

* **Dynamic Inputs**: Pass topic via crew.kickoff(inputs={"topic": topic}) and update tasks to use {inputs[topic]} in descriptions (requires inputs parameter in Task).
* **Memory**: Add memory=True to Crew with Chroma (pip install chromadb) for context retention across runs.
* **Error Handling**: Wrap kickoff() in a try-except to catch API or tool failures.