

# CrewAI



# 1. CrewAI 개요







## Crew AI란 무엇인가

Crew AI는 \*\*여러 개의 인공지능 에이전트(Agent)\*\*가  
\*\*팀(Crew)\*\*을 구성하여 **협업 형태로 문제를 해결**하도록 설계된 **멀티-에이전트 프레임워크**이다.

즉, 단일 챗봇이 모든 일을 처리하는 대신  
'연구자-작가-검토자' 같은 **역할 기반 분업 구조**로 LLM을 연결한다.

→ "한 명의 AI가 아닌 **협업하는 AI 팀**" 개념이다.






## ⚙ 기본 구조

Crew AI는 세 가지 핵심 구성요소로 이루어진다.

구성요소	설명
Agent	특정 역할과 목표를 가진 LLM 인스턴스이다. (예: Researcher, Writer, Analyst 등)
Task	에이전트가 수행할 구체적인 작업이다. (예: 데이터 조사, 요약 작성 등)
Crew	여러 Agent와 Task를 묶어 실행 순서와 흐름을 제어한다.

이 구조는 LangChain의 `Runnable`, `Chain`, `AgentExecutor` 구조와 유사하지만, Crew AI는 **협업과 워크플로우 자동화에 초점을 둔 것**이 특징이다.



## LangChain과의 관계

Crew AI는 내부적으로 LangChain과 유사한 LLM 호출 인터페이스를 사용하며, **LangChain OpenAI, LangChain Chroma, LangChain Tools** 등을 그대로 연결할 수 있다.

예를 들어, LangChain의

```
python


from langchain_openai import OpenAI
```

로 정의한 LLM 객체를 Crew AI의 Agent에 바로 전달하여 사용할 수 있다.

이로써 LangChain의

**PromptTemplate, Retrieval QA, RAG, Tool Chain** 기능을

Crew AI의 **협업형 작업흐름**에 결합할 수 있다.



## 동작 흐름 요약


### 1. Agent 정의

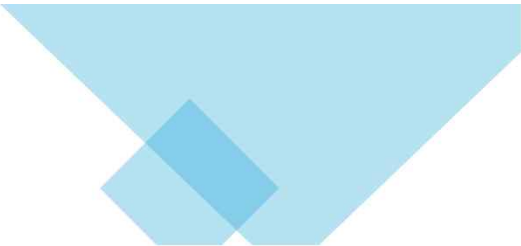
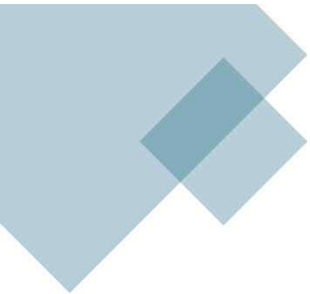
- 각 에이전트는 역할(role), 목표(goal), 배경(backstory), 사용 LLM 등을 가진다.
- 예: `researcher`, `writer`, `editor`

### 2. Task 정의

- 어떤 Agent가 무엇을 할지를 `Task` 로 정의한다.
- 예: "연구자가 핵심 정보를 찾고, 작가가 이를 요약한다."

### 3. Crew 구성 및 실행

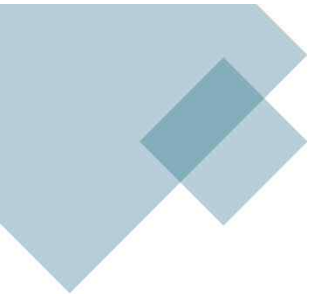
- 여러 Task를 순서대로 실행하거나 병렬로 실행한다.
  - 실행은 `crew.kickoff()` 메서드로 시작한다.
- 



## 특징 요약

항목	설명
분업형 협업 구조	여러 에이전트가 역할별로 협력한다.
자연스러운 워크플로우 구성	순차, 병렬, 조건 분기 가능
LangChain 호환성	LangChain의 LLM, Tool, VectorDB 등을 직접 활용 가능
YAML 기반 구성 지원	설정파일로도 에이전트 팀을 정의할 수 있다.
생산성 향상	복잡한 멀티스텝 작업을 자동화하여 프로젝트형 AI 개발에 유리하다.





## 활용 예시

분야

예시

콘텐츠 생성

연구자 + 작가 + 교정자 팀으로 블로그 초안 작성

데이터 분석

데이터 로더 + 분석가 + 리포터 에이전트 구성

기업 업무 자동화


이메일 요약, 문서 검토, 보고서 생성 팀 구성

교육용 프로젝트

학생들이 에이전트 역할을 설계하며 협업 AI 구조 학습







## 정리

Crew AI는

“에이전트를 하나 더 만드는 게 아니라, **AI 팀을 조직한다**”

는 철학을 기반으로 한다.

LangChain이 **하나의 대화 흐름을 자동화하는 도구**라면,  
Crew AI는 **여러 대화 흐름을 조정하고 협업시키는 프레임워크**이다.



## 기본 예제 코드

```
1 !pip install crewai[tools]
```

```
1 import os
2 from langchain_openai import OpenAI
3 from crewai import Agent, Task, Crew
4
5 # LLM 세팅
6 llm = OpenAI(model="gpt-4o-mini")
7
8 # 에이전트 정의
9 researcher = Agent(
10     role="Researcher",
11     goal="태양에너지에 대한 정보를 수집한다",
12     backstory="나는 청정에너지에 관심이 많은 데이터 연구자이다",
13     llm=llm,
14     verbose=True
15 )
16
17 writer = Agent(
18     role="Writer",
19     goal="수집된 정보를 바탕으로 50단어로 요약을 작성한다",
20     backstory="나는 복잡한 정보를 간결하게 정리하는 작가이다",
21     llm=llm,
22     verbose=True
23 )
```

```
25 # 태스크 정의
26 research_task = Task(
27     description="태양에너지에 대한 핵심 사실 3가지를 찾아라.",
28     agent=researcher,
29     expected_output="3개의 핵심 사실 리스트"
30 )
31
32 write_task = Task(
33     description="위의 사실들을 바탕으로 50단어 요약을 작성하라.",
34     agent=writer,
35     expected_output="50단어 요약문"
36 )
37
38 # 크루(작업팀) 구성 및 실행
39 crew = Crew(
40     agents=[researcher, writer],
41     tasks=[research_task, write_task],
42     verbose=True
43 )
44
45 result = crew.kickoff()
46 print("최종 결과:", result)
47
48 # crew.kickoff()를 호출하면 CrewAI가 내부적으로 각 Task와
49 # Agent 실행 단계를 터미널 스타일로 시각화해서 표시한다.
```

## 결과 출력

### Crew Execution Started

#### Crew Execution Started

Name: crew

ID: 35401dfb-8e70-4e73-ac0d-076e38ae4ac9

Tool Args:

### 👤 Agent Started

Agent: **Researcher**

Task: 태양에너지에 대한 핵심 사실 3가지를 찾아라.

Loading widget...

### ✅ Agent Final Answer

Agent: **Researcher**

Final Answer:

1. **\*\*재생 가능 에너지\*\***: 태양에너지는 태양광 발전을 통해 전기를 생성할 수 있는 무한한 에너지원으로, 지구의 어느 곳에서나 활용할 수 있으며, 다른 에너지원과 비교하여 환경에 미치는 영향이 적습니다. 이는 화석 연료와 달리 이산화탄소와 같은 온실가스를 배출하지 않기 때문에 기후 변화에 대한 해결책으로 주목받고 있습니다.
2. **\*\*기술 발전\*\***: 태양광 패널의 기술은 지속적으로 발전하고 있으며, 현재의 태양광 발전 시스템은 더 높은 효율성과 내구성을 갖추고 있습니다. 최신 태양광 패널은 20% 이상의 변환 효율을 자랑하며, 향후 30% 이상의 효율을 목표로 하는 연구가 진행되고 있습니다. 또한, 태양광 발전은 배터리 저장 시스템과 결합되어 에너지를 필요할 때 사용할 수 있도록 하고 있습니다.
3. **\*\*경제적 이점\*\***: 태양에너지는 초기 설치 비용이 있지만, 장기적으로 보면 전기 요금을 절감할 수 있는

Loading widget...

### Task Completion

#### Task Completed

Name: 2d662c7a-961d-4a9b-85bd-75d05be1b57e  
Agent: Researcher  
Tool Args:

### 👤 Agent Started

Agent: **Writer**

Task: 위의 사실들을 바탕으로 50단어 요약을 작성하라.

Loading widget...

### ✅ Agent Final Answer

Agent: **Writer**

Final Answer:

태양에너지는 환경에 미치는 영향이 적고, 지속적으로 발전하는 기술 덕분에 높은 효율성을 보입니다. 초기 설치 비용이 있지만, 전기 요금 절감으로 장기적인 경제적 이점을 제공합니다. 이는 기후 변화 해결에 기여하는 중요한 에너지원입니다.

최종 결과: 태양에너지는 환경에 미치는 영향이 적고, 지속적으로 발전하는 기술 덕분에 높은 효율성을 보입니다. 초기 설치 비용이 있지만, 전기 요금 절감으로 장기적인 경제적 이점을 제공합니다. 이는 기후 변화 해결에 기여하는 중요한 에너지원입니다.

### Task Completion

#### Task Completed

Name: a5db3f22-a9b2-4503-989d-cfdb4c47194b  
Agent: Writer  
Tool Args:

## 1. 출력 주체

이 출력은 **CrewAI 프레임워크 내부의 콘솔 로그 시스템**에서 발생한다.

즉, 네가 `crew.kickoff()` 를 호출하면 CrewAI가 내부적으로 각 Task와 Agent 실행 단계를 **터미널 스타일로 시각화**해서 표시한다.

---

## 2. 출력 구조

이 로그는 CrewAI의 실행 단계를 시각적으로 구분하기 위해 ANSI 컬러 코드와 이모지 등을 사용한다.

- `Crew Execution Started`: 전체 워크플로우(Crew)가 시작될 때 출력된다.
- `Agent Started`: 특정 에이전트(예: Researcher, Writer)가 실행될 때 출력된다.
- `Task: ... / Thinking...`: Task가 실행 중임을 나타낸다.
- 이후에 `Completed`, `Output`, `Crew Finished` 같은 단계가 이어진다.



### 3. 왜 이런 포맷이 사용되는가

CrewAI는 콘솔에서 **멀티에이전트의 진행상황을 한눈에 볼 수 있도록** 설계되어 있다. 그래서 일반 로그보다 시각적으로 화려하고, 각 에이전트마다 색상이 다르게 지정되어 있다. 이는 CrewAI의 내부 모듈 중 `logger.py` 또는 `console.py` 에서 ANSI 코드로 처리한다.

### 4. 끄는 방법

만약 이런 형식의 콘솔 로그가 불필요하다면

`Crew` 나 `Agent` 정의 시 `verbose=False` 로 설정하면 대부분의 출력이 줄어든다.

python

📄 코드 복사

```
crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, write_task],
    verbose=False # 콘솔 출력 최소화
)
```

하지만 완전히 없애려면 `CREWAI_LOG_LEVEL=ERROR` 환경변수를 지정하거나, 파이썬 표준 로깅 설정으로 `logging.getLogger("crewai").setLevel(logging.ERROR)` 을 추가하면 된다.

## 2. CrewAI 다중 협업 AI 에이전트 구현

---



# 다중 협업 AI Agent 구현 분석

CrewAI + LangChain + DuckDuckGo(ddgs) 를 활용한 다중 협업 AI 에이전트 시스템으로, "기술 블로그 글 작성"이라는 하나의 목표를 위해 리서처 → 작가 → 편집자 → SEO 전문가의 협업 흐름을 자동화합니다.

## 1. 초기 설정 단계

### (1) 라이브러리 및 환경설정

- `crewai`, `langchain_openai`, `ddgs`, `dotenv` 등 필요한 라이브러리 로드
- `.env` 파일에서 `OPENAI_API_KEY` 불러옴
- DuckDuckGo 검색용 `DDGS()` 인스턴스 준비  
→ 없으면 `"pip install ddgs"` 안내 출력

## 2. 도구(Tools) 정의 단계

각 에이전트가 사용할 유틸리티 도구들 정의:

도구 이름	기능	설명
<code>duckduckgo_search</code>	일반 검색	DuckDuckGo로 일반 텍스트 검색 수행
<code>duckduckgo_news_search</code>	뉴스 검색	DuckDuckGo 뉴스 탭에서 최신 기사 수집
<code>content_analyzer_tool</code>	콘텐츠 분석	단어 수, 문장 길이, 가독성, 구조 평가
<code>seo_analyzer_tool</code>	SEO 분석	키워드 밀도, 태그, 구조 기반 SEO 점수 산출
<code>fact_checker_tool</code>	사실 확인	DDGS 검색으로 진술 검증 및 출처 제공

### 3. 언어모델 (LLM) 설정 단계

`get_openai_llm()` 함수가 호출되어 `ChatOpenAI` 인스턴스를 반환함  
→ 모델명 `"gpt-4o-mini"`, 온도(`temperature`)는 역할별로 다르게 설정

역할	모델 온도	설명
리서처	0.3	정확성 중시
작가	0.8	창의성 강조
편집자	0.5	균형형
SEO 전문가	0.4	분석 중심

## 4. 에이전트 정의 단계

각 역할별 Agent 인스턴스 생성:

에이전트	역할	주요 도구	설명
researcher	기술 리서처	duckduckgo_search , duckduckgo_news_search , fact_checker_tool	최신 트렌드 및 자료 조사
writer	블로그 작가	(없음)	리서치 내용을 바탕으로 블로그 초안 작성
editor	편집자	content_analyzer_tool	품질 검토, 문체/가독성 개선
seo_specialist	SEO 전문가	seo_analyzer_tool	키워드/메타태그 등 SEO 최적화

## 5. 태스크(Task) 정의 단계

각 에이전트가 수행할 세부 작업(Task)을 명시:

태스크	담당 에이전트	내용
research_task	researcher	DuckDuckGo로 리서치 보고서 작성
writing_task	writer	블로그 본문 작성 (1200~1800단어)
editing_task	editor	문법, 가독성, 논리 흐름 개선
seo_task	seo_specialist	SEO 분석 및 최적화 보고서 작성

각 Task는 `expected_output` 템플릿을 갖고, 출력 품질을 가이드함.

## 6. Crew 구성 단계

`BlogCreationCrew` 클래스 내부에서 다음이 수행됨:

1. `Agent` 4명 생성
2. `Task` 4개 생성
3. `Crew` 객체 생성:

python

```
crew = Crew(  
    agents=[researcher, writer, editor, seo_specialist],  
    tasks=[research_task, writing_task, editing_task, seo_task],  
    process=Process.sequential, # 순차 실행  
    verbose=True  
)
```

## 7. 실행 단계 (`run()` 메서드)

1. 시작 시점 로그 출력
2. `crew.kickoff()` 실행 → 순차 워크플로우 시작:

리서처 → 작가 → 편집자 → SEO 전문가

3. 각 단계의 결과가 다음 태스크의 입력으로 전달됨
4. 모든 태스크 완료 후 결과를 `dict` 형태로 반환

성공 시:

```
python

{
    "success": True,
    "topic": "2025년 AI 에이전트 프레임워크 트렌드와 실무 적용",
    "result": <Crew 결과>,
    ...
}
```

## 8. 병렬 리서치 예제 (`ParallelResearchCrew`)

- 여러 주제를 동시에 조사하기 위한 별도 클래스
  - 각 주제별 `Agent` 와 `Task` 생성
  - 각기 다른 주제(예: RAG, LangChain, 보안 등)를 병렬 처리
  - `Crew(process=Process.sequential)` 로 실행 (동시에 여러 개 리서치 가능)
- 

## 9. 메인 실행 흐름 (`main()`)

1. 환경 변수 확인 (`OPENAI_API_KEY`, `ddgs` 설치 여부)
2. 예제 1: `BlogCreationCrew` 실행
  - 블로그 작성 전체 파이프라인 수행
3. 예제 2: `ParallelResearchCrew` 실행
  - 여러 주제 병렬 리서치 수행
4. 실행 결과를 콘솔 출력
5. 사용자가 `y` 입력 시 결과를 `.txt` 파일로 저장



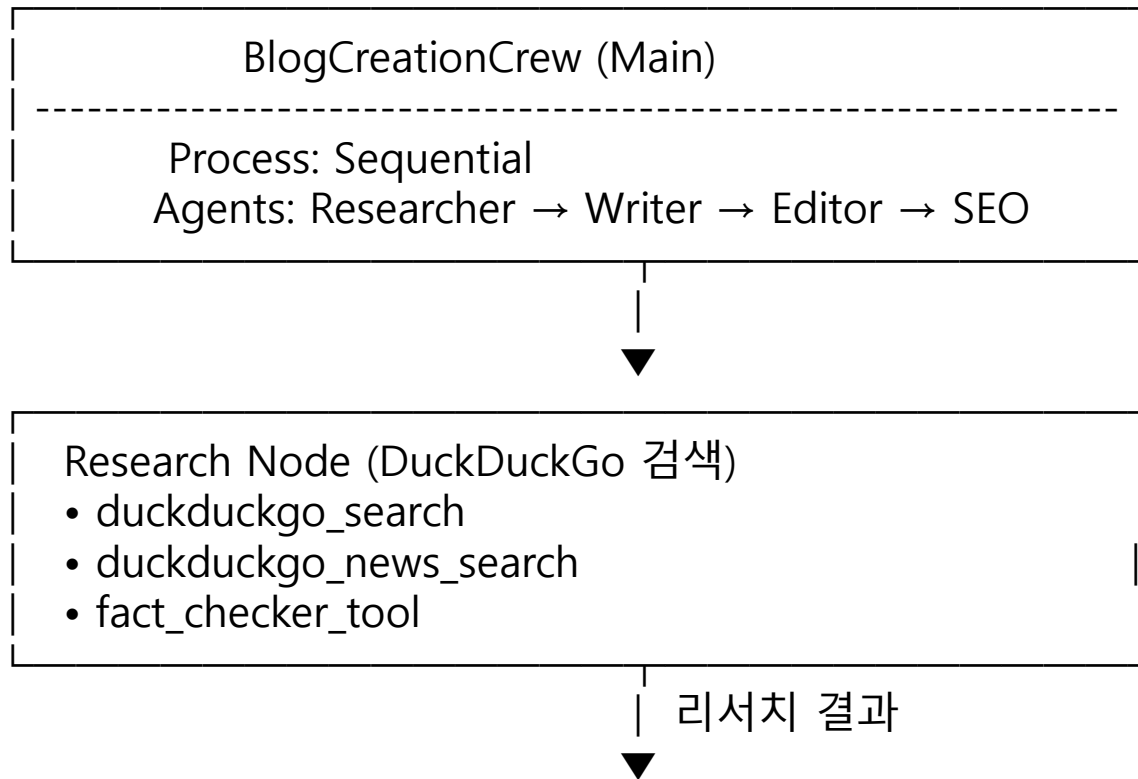
## 10. 전체 실행 순서 요약

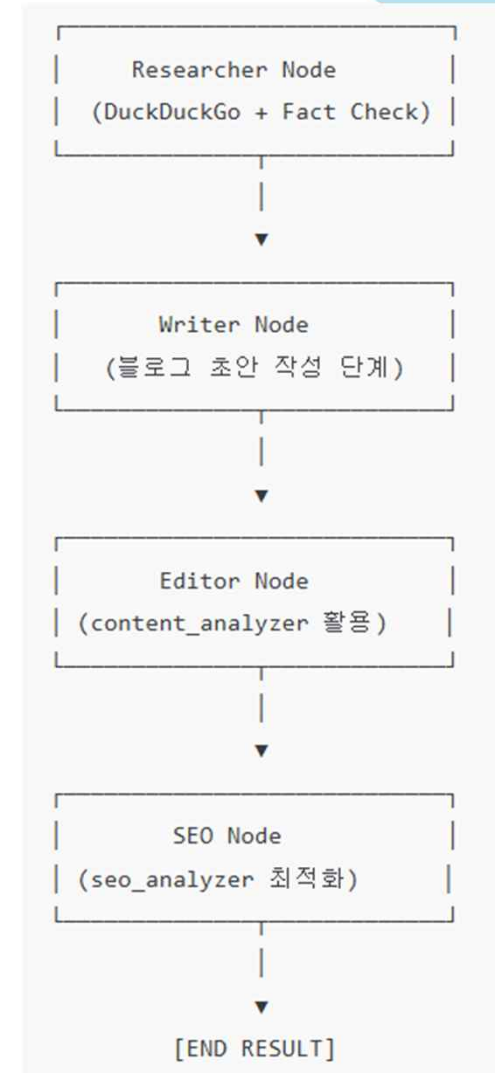
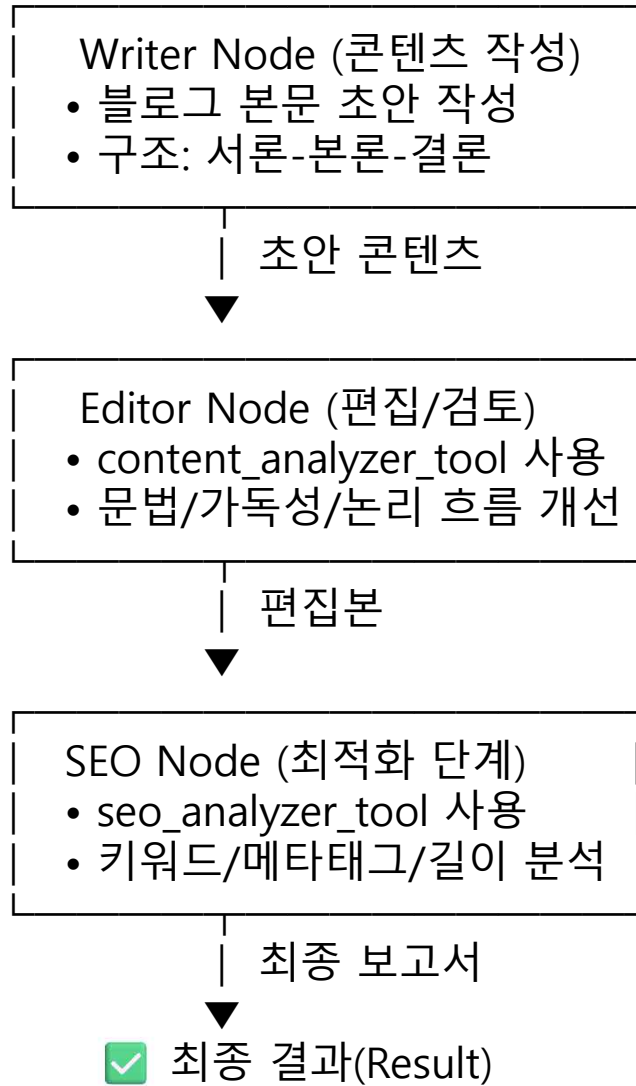
arduino

main()

- | 환경 확인 및 안내
- | BlogCreationCrew(topic)
  - | | LLM 생성
  - | | Agents 생성
  - | | Tasks 생성
  - | | Crew.kickoff() 실행
    - | | | Researcher 단계
    - | | | Writer 단계
    - | | | Editor 단계
    - | | | SEO 단계
- |
- | ParallelResearchCrew(topics)
  - | | 각 주제별 리서치 Agent/Task 생성
  - | | Crew.kickoff() 실행
- |
- | (선택) 결과 파일 저장

## 전체 워크플로우





이 코드의 핵심은 "CrewAI의 다중 에이전트 협업 시스템"을 통해 실제 협업(협동적 작업)이 일어나는 구조를 구현한 것입니다.

즉, 단순한 "순서 실행"이 아니라, **역할(role)** 과 **책임(task)** 이 다른 여러 AI 에이전트들이 서로의 결과물을 바탕으로 **협업(cooperation)** 하는 방식입니다.

## 1. 협업의 핵심 구조

협업이 이루어지는 핵심은 아래 세 요소의 **연결과 흐름**입니다:

구성요소	설명	협업 역할
Agent	특정 역할을 수행하는 개별 AI (리서처, 작가, 편집자, SEO 전문가)	"역할 분담"
Task	각 Agent가 수행해야 할 구체적 업무 내용	"업무 정의"
Crew (Process.sequential)	여러 Agent와 Task를 하나의 파이프라인으로 연결	"협업 실행"

## 2. 협업이 이루어지는 실제 코드 부분

### (1) Crew 생성 부분

python

```
crew = Crew(  
    agents=[  
        self.researcher,  
        self.writer,  
        self.editor,  
        self.seo_specialist  
    ],  
    tasks=[  
        self.research_task,  
        self.writing_task,  
        self.editing_task,  
        self.seo_task  
    ],  
    process=Process.sequential,  
    verbose=True,  
)
```

여기서 협업의 구조가 명시됩니다.

- `agents` : 각 역할 담당자 등록
- `tasks` : 각 역할별 구체적 업무 등록
- `Process.sequential` : 리서처 → 작가 → 편집자 → SEO 전문가 순서로 "결과물이 다음 Agent에게 전달"
- 즉, 한 Agent의 출력이 다음 Agent의 입력으로 자동 연결됨

### 3. 협업 흐름 요약

SCSS

리서처 (Researcher)

↓ 조사 결과 전달

작가 (Writer)

↓ 작성 초안 전달

편집자 (Editor)

↓ 편집 결과 전달

SEO 전문가 (SEO Specialist)

↓ 최종 보고서 완성

각 단계에서 “이전 단계의 결과물”을 참조하여 자기 역할을 수행하는 구조입니다.

## 4. 단계별 협업 상세 분석

### (1) 리서처 (Researcher)

python

```
self.researcher = create_researcher_agent(self.researcher_llm)
self.research_task = create_research_task(self.researcher, topic)
```

- DuckDuckGo 검색과 뉴스 검색을 사용해 자료 조사
- `fact_checker_tool` 로 사실 검증까지 수행
- 출력: "정리된 리서치 보고서 (요약 + 출처 포함)"

➡ 이 결과는 다음 단계 **Writer**에게 전달됩니다.

## (2) 작가 (Writer)

python

```
self.writer = create_writer_agent(self.writer_llm)
self.writing_task = create_writing_task(self.writer, topic)
```

- 리서치의 보고서를 바탕으로 블로그 글을 작성
- 구조적으로 "서론-본론-결론" 구성
- 출력: "블로그 초안"

➡ 초안은 다음 단계 **Editor**에게 전달됩니다.

## (3) 편집자 (Editor)

python

```
self.editor = create_editor_agent(self.editor_llm)
self.editing_task = create_editing_task(self.editor)
```

- 작가의 초안을 입력으로 받아 검토 및 개선
- `content_analyzer_tool` 을 사용하여 가독성, 문장 길이, 구조를 분석
- 수정 및 개선사항을 적용한 "편집본" 출력

➡ 이 편집본이 다음 단계 **SEO 전문가**에게 전달됩니다.



#### (4) SEO 전문가 (SEO Specialist)

python

```
self.seo_specialist = create_seo_specialist_agent(self.seo_llm)
self.seo_task = create_seo_task(self.seo_specialist)
```

- 편집자의 최종본을 입력으로 받아 SEO 분석 수행
- `seo_analyzer_tool` 을 사용하여 키워드, 길이, 태그 구조 평가
- 개선 제안 및 최적화된 메타데이터 생성
- 결과적으로 "SEO 최적화된 블로그 포스트" 완성

#### SEO (Search Engine Optimization)

→ "검색 엔진 최적화"를 뜻함.

## 5. 협업이 실제로 이루어지는 방식

CrewAI 내부에서는 각 Task가 실행될 때 다음과 같은 **협업 메커니즘**이 작동합니다:

단계	내부 협업 로직
① 리서처 수행	DuckDuckGo로 최신 정보 수집 → 텍스트 요약 → 보고서 반환
② 작가 수행	이전 Agent 결과( <code>research_task.output</code> )를 프롬프트 입력으로 받아 블로그 초안 작성
③ 편집자 수행	이전 Agent 결과( <code>writing_task.output</code> )를 입력받아 <code>content_analyzer_tool</code> 활용
④ SEO 수행	편집 결과( <code>editing_task.output</code> )를 입력받아 SEO 점수 및 개선 제안 산출

즉, 각 Agent는 **자신만의 전문 모델(temperature, 도구, 목표)** 을 가지고 이전 단계의 결과를 “협업 입력”으로 사용합니다.

## 6. 협업이 중요한 이유

특징	CrewAI 협업 방식	단일 LLM 대비 장점
역할 분리	Agent별 전문 목표	더 명확한 책임 분담
입력 체인	이전 결과 자동 전달	컨텍스트 유지, 효율적 협업
도구 사용	각 Agent 전용 Tool	기능적 확장성 확보
평가 및 개선	Editor/SEO 단계	품질과 완성도 향상
실행 제어	<code>Process.sequential</code>	명시적 실행 순서 보장

## 7. 병렬 협업 (ParallelResearchCrew)에서도 협업 존재

`ParallelResearchCrew` 클래스에서는 각 주제가 별도의 **Agent + Task** 조합으로 생성됩니다

- 각 리서처가 다른 주제를 조사함 (동시 병렬 처리)
- Crew가 전체 주제를 통합하여 결과를 반환  
→ “협업의 형태는 병렬적(cooperative parallelism)” 으로 확장된 구조

## 8. 요약: 협업의 본질

단계	협업 방식	전달되는 결과
Researcher → Writer	정보 제공	리서치 요약
Writer → Editor	초안 공유	블로그 글
Editor → SEO	편집 결과 공유	완성도 높은 글
SEO → Result	분석 피드백	SEO 최적화 글

결과적으로,

이 코드의 협업은 “단계 간 결과 전달 + 역할 기반 협력 + 도구 기반 지원”  
즉, “LLM 기반 전문가 팀의 자동화 협업 시스템”으로 구현되어 있습니다.



감사합니다