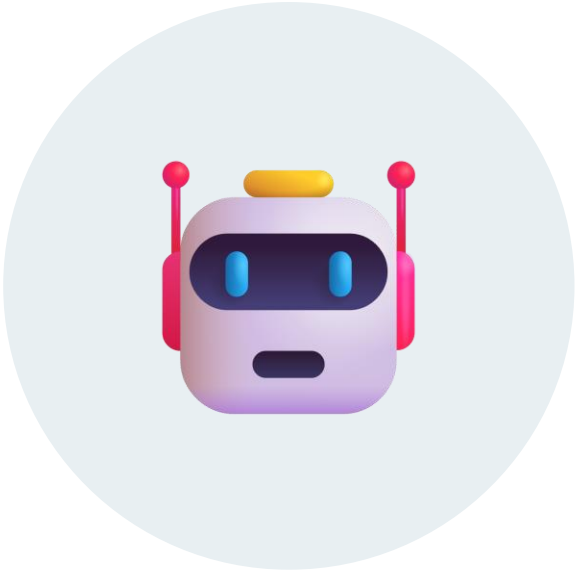
01

Agent

AI 에이전트란



🧠 스스로 문제를 분석하여

✏️ 쉽게 해결 가능한 작은 단위의 문제로 분리하고

🔧 외부 툴을 활용해 처리한 뒤

🤔 반복적으로 결과물을 검토하고

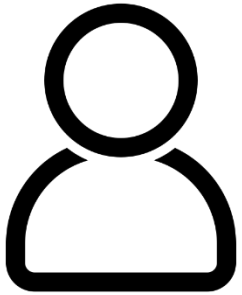
🧐 메모리에 저장해둔 사용자의 페르소나와 정보를 활용해

답을 내어주는 기술

Agent란

단순히 정보(혹은 질의응답)를 제공하는 것을 넘어서,

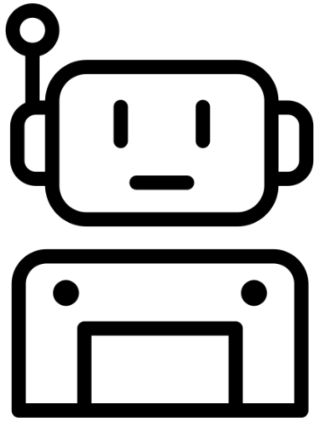
사용자의 요구를 더 정확하게 파악하고 만족시키는 것을 목표



User

"회의 일정을 잡아줘"

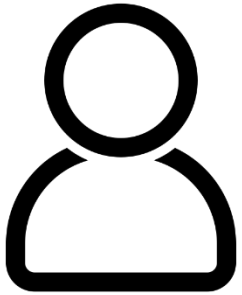
참석자들의 가능한 시간을 조율하고, 최적의 회의 시간을 찾아내어 일정을 설정하고 참석자들에게 초대장을 보내기 완료!



AI Agent를 활용하면?

단순히 정보(혹은 질의응답)를 제공하는 것을 넘어서,

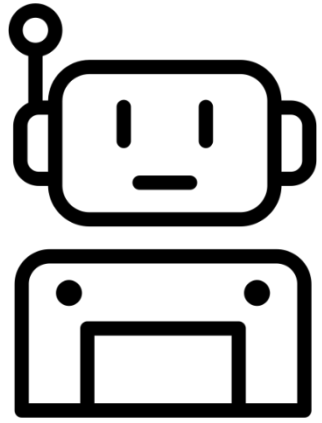
사용자의 요구를 더 정확하게 파악하고 만족시키는 것을 목표



User

"출퇴근용 자동차를 알아봐줘."

사용자의 상황과 요구 사항을 분석하여, 예산, 연비, 차량 유형 등을 고려한 맞춤형 자동차 추천 목록을 제공 완료!



AI agent vs. ChatGPT

자율성과 상호작용 능력

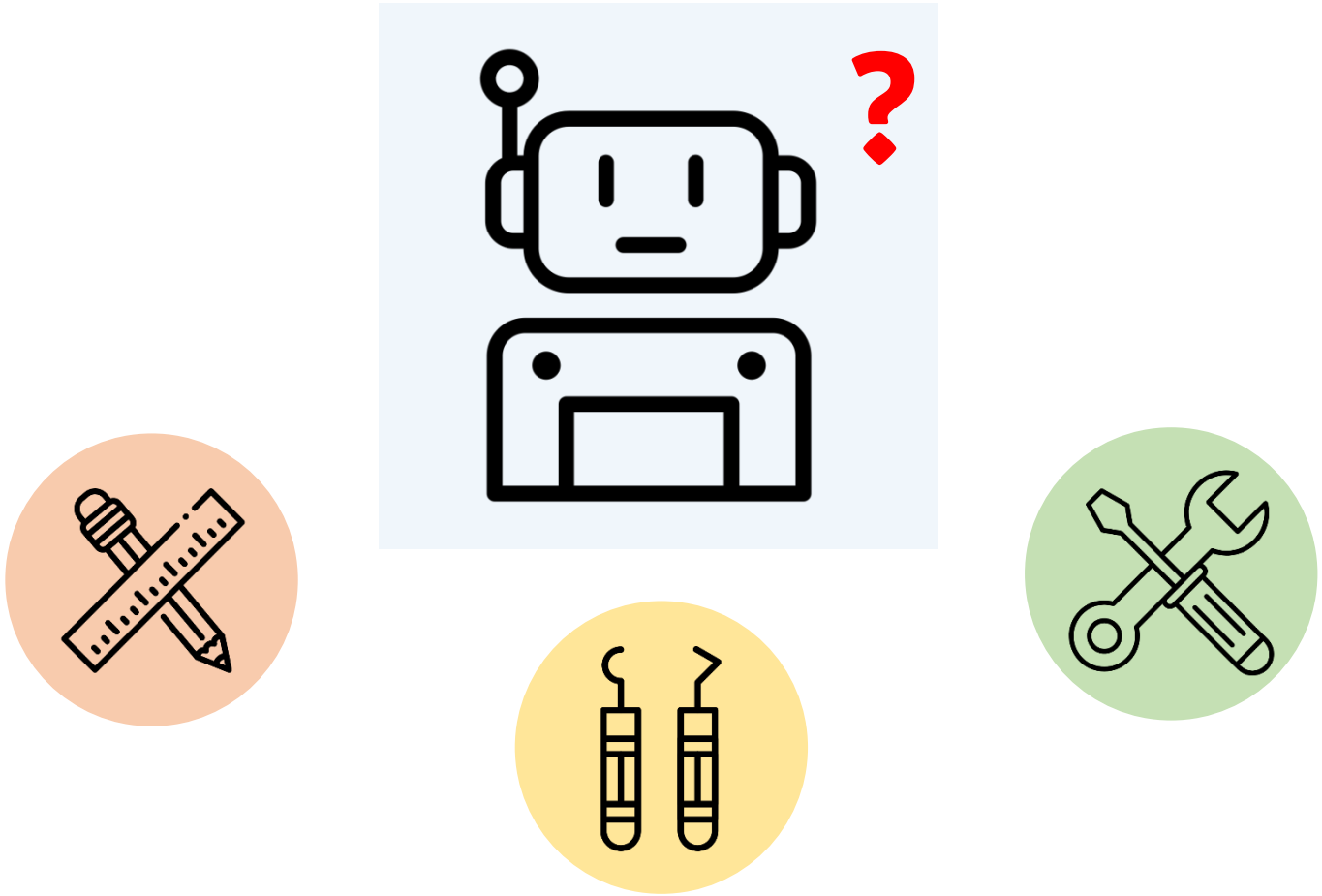
사용자가 요구한 작업(Task)의 완료를 위해
활용가능한 여러 도구(Tool)와의
상호작용(Interaction)을 연쇄적으로,
자율적으로(Autonomously) 수행할 수 있는 기술

주로 단일 플러그인을 사용하여 질문에 답변

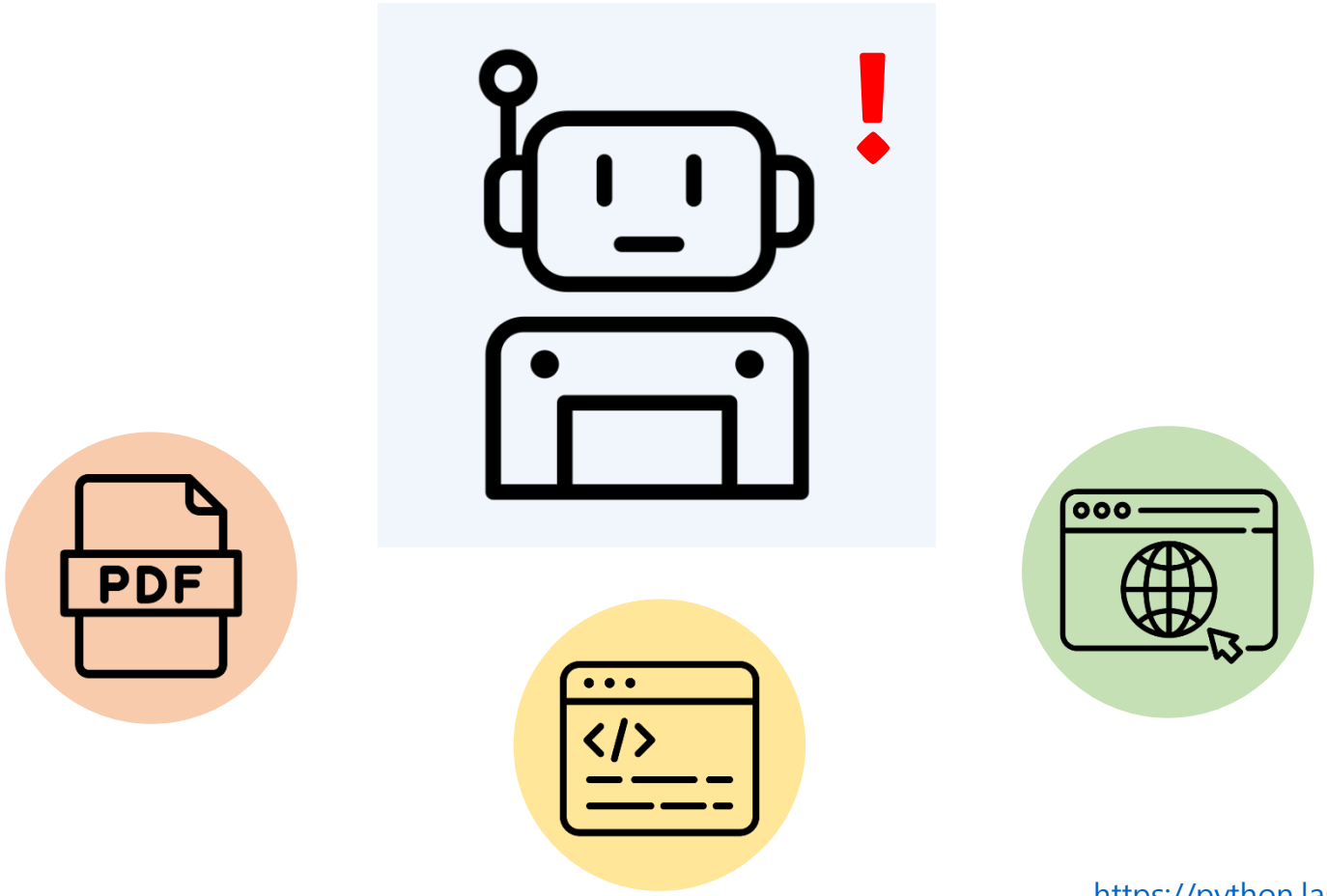
서비스 제공사에서 준비 및 제휴한 도구로,
그 기능이 한정적

AI Agent를 활용하면?

주어진 쿼리에 대해 스스로 생각하여 적절한 행동을 선택하는 대리인

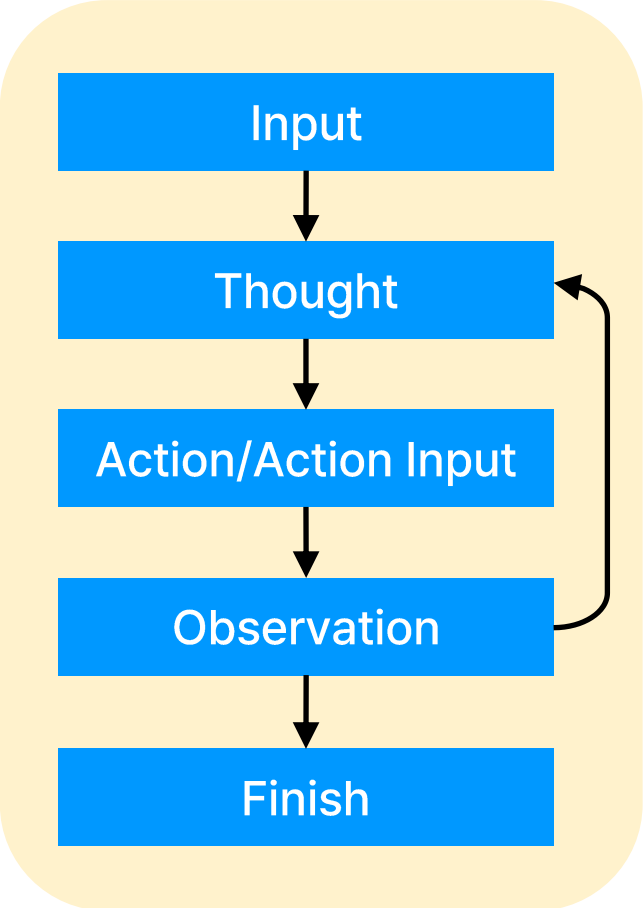


에이전트가 활용할 수 있는 기능적 요소



https://python.langchain.com/v0.1/docs/integrations/tools/

Agent의 작업 처리 순서도



1. Agent Input 사용자가 Agent에게 작업을 할당

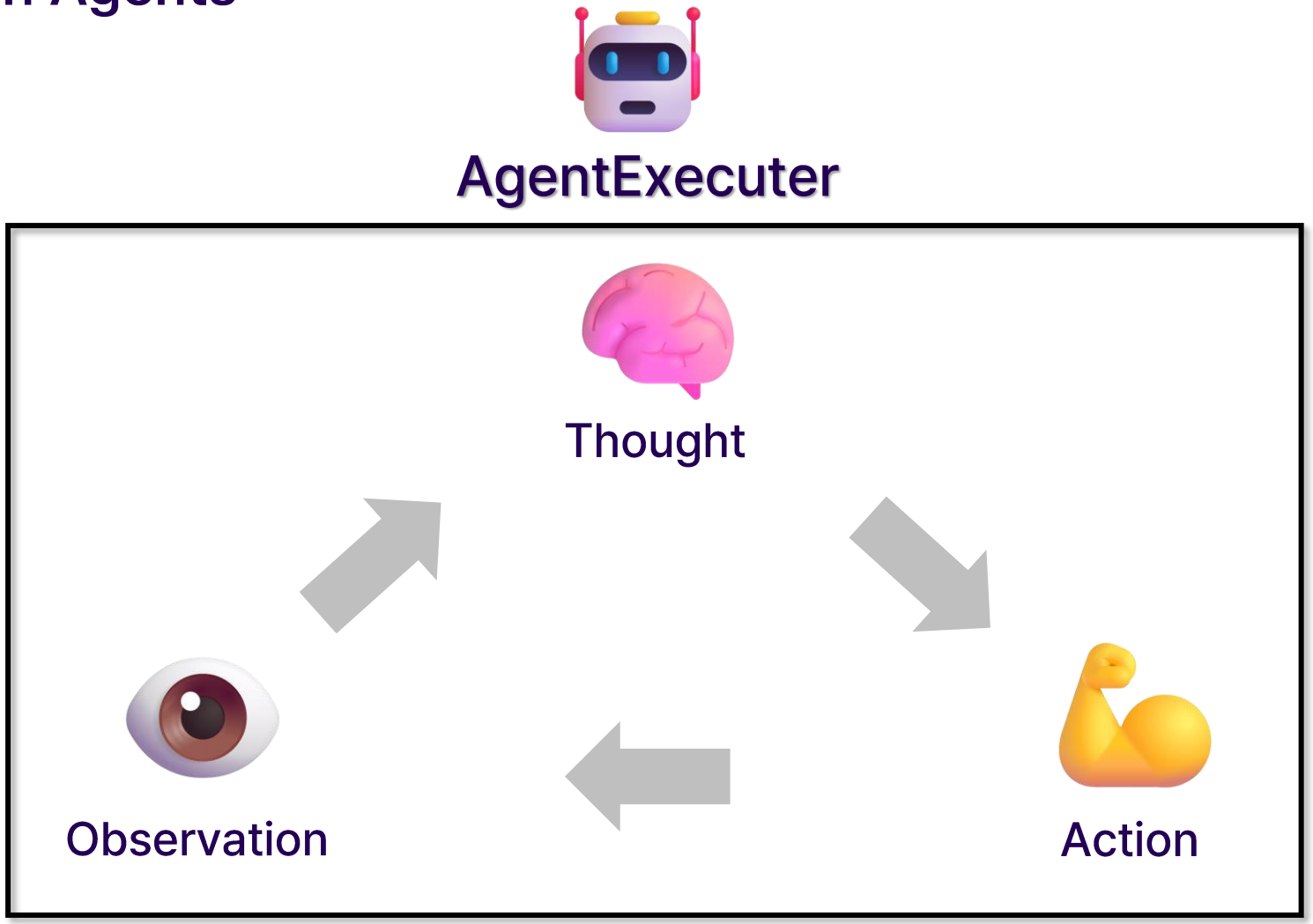
2. Thought Agent가 작업을 완수하기 위해 무엇을 할지 생각

3. Action/Action Input 사용할 도구를 결정하고, 도구의 입력(함수의 입력값)을 결정

4. Observation 도구의 출력 결과를 관찰

5. Finish 관찰 결과 작업을 완료했다는 판단에 도달할 때까지 2~4번 과정을 반복

Action Agents



Action 출력

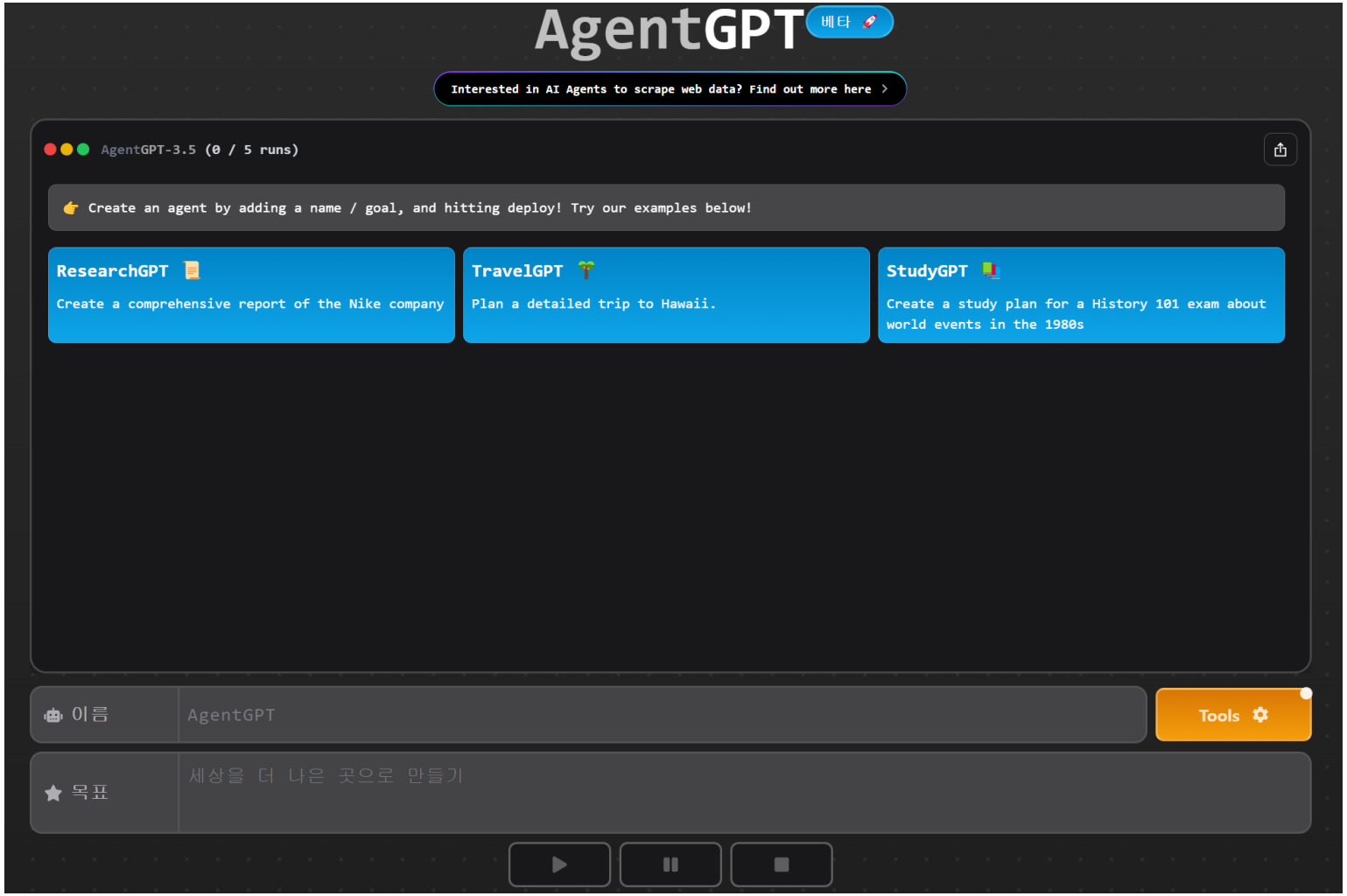
Observation 출력

Action 출력

Observation 출력

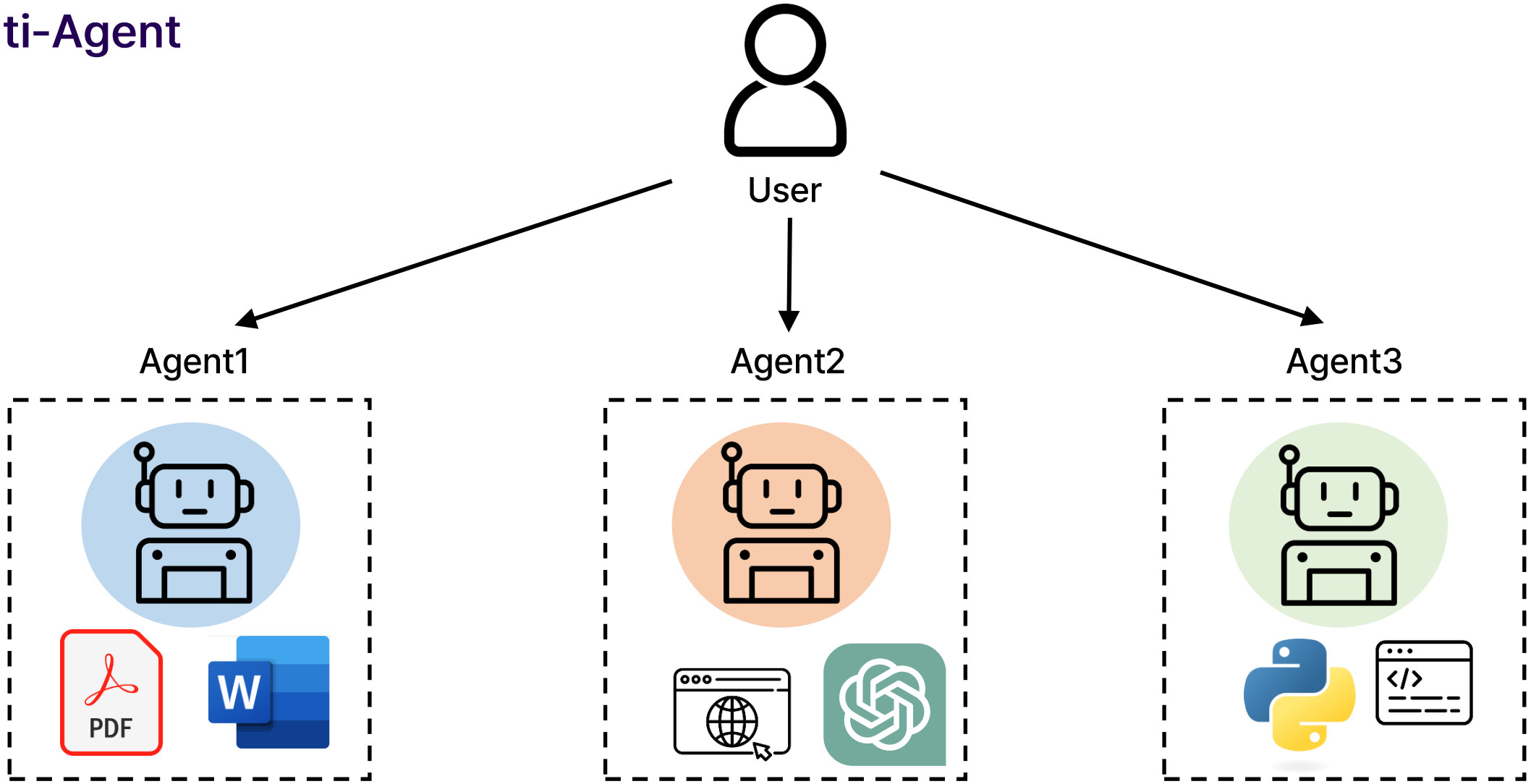
... (목표 달성까지 계속) ...

AgentGPT

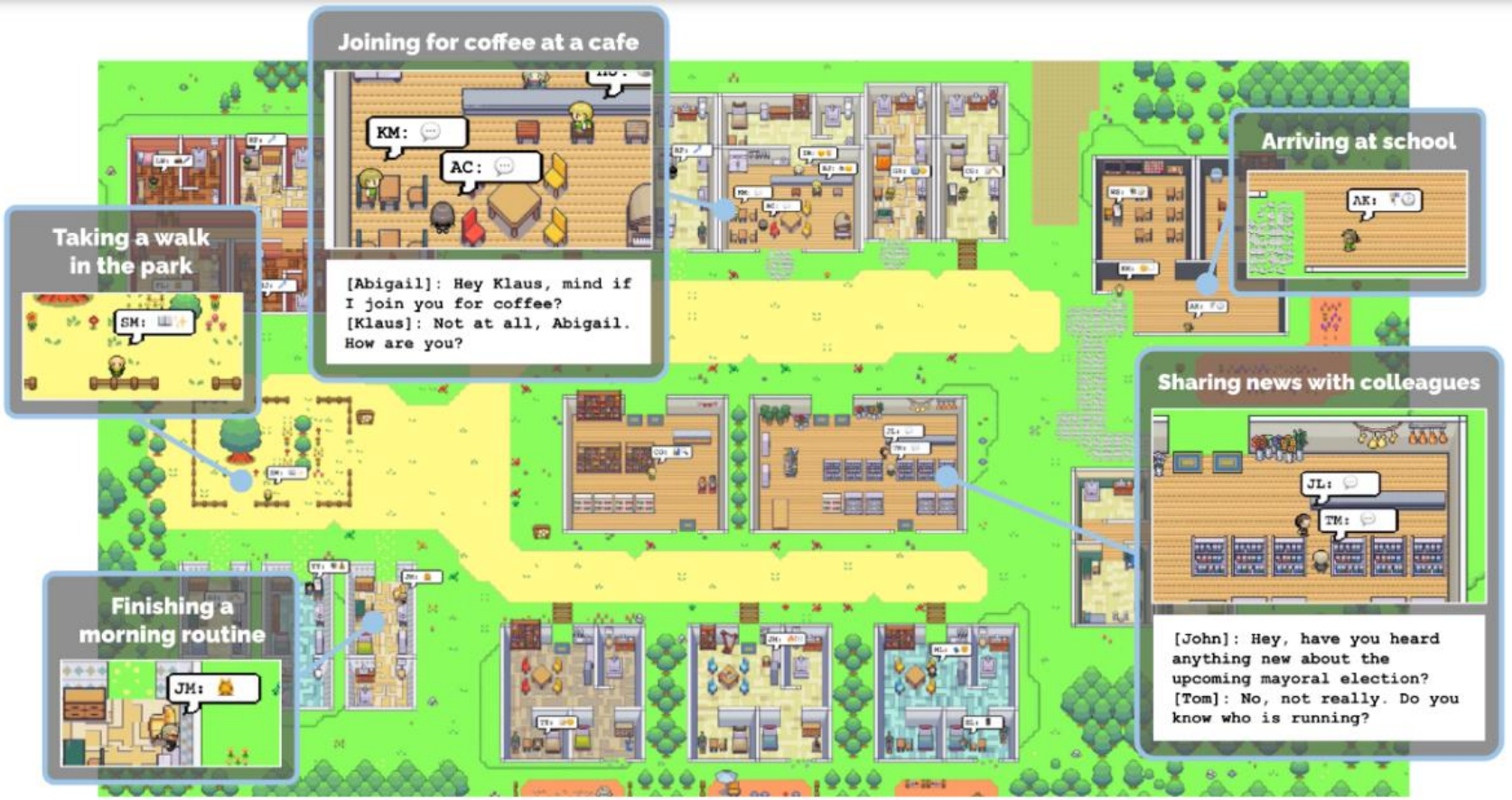


AgentGPT 링크

Multi-Agent



에이전트들의 행동과 상호 작용



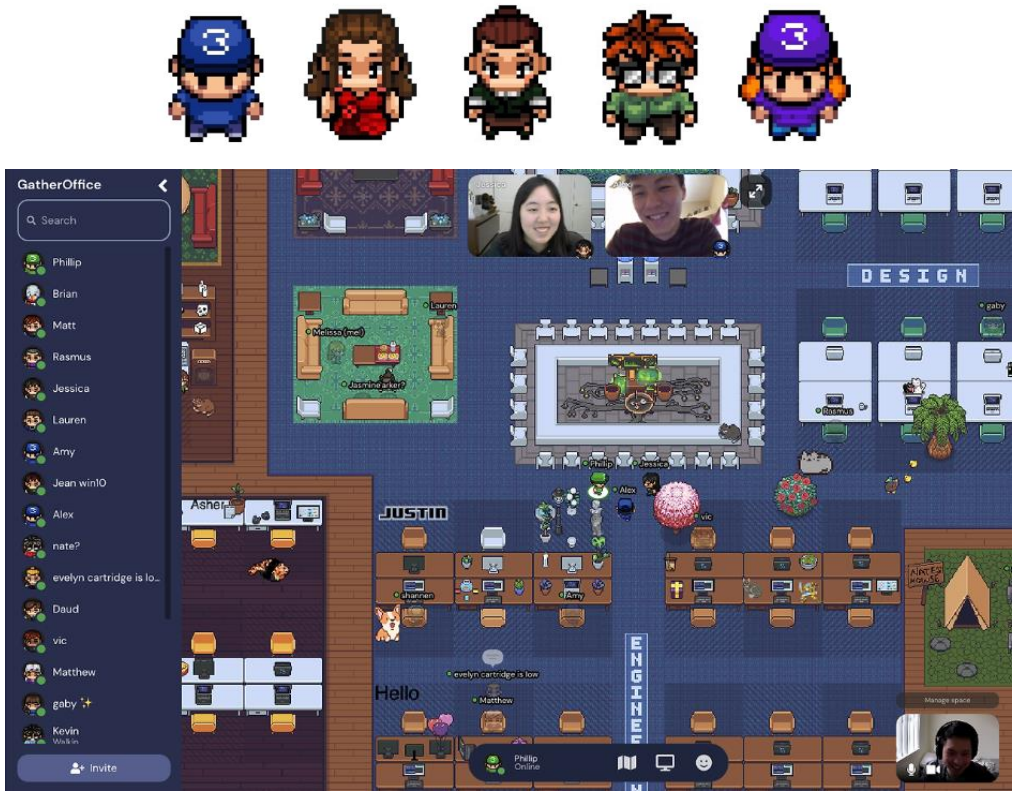
source: "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond"

AI가 사는 마을 Generative Agent

SIMS에서 착안한 가상 세계와 에이전트들에 대한 환경

에이전트는 그들이 목격한 object들을 기억

< SIMS4 >



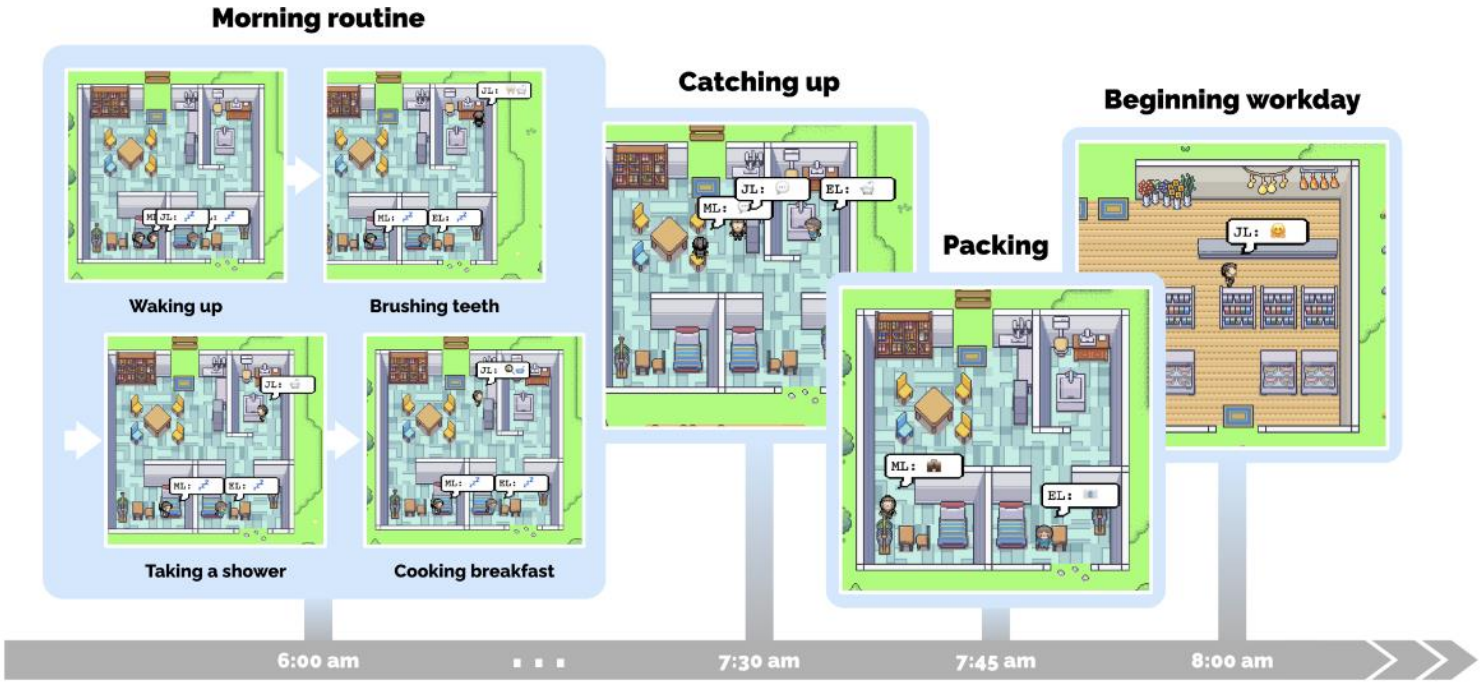
< Gather Town >

source: https://sensortower.com/blog/the-sims-mobile-revenue-25-million, https://brunch.co.kr/@uniher/10?pidx=1

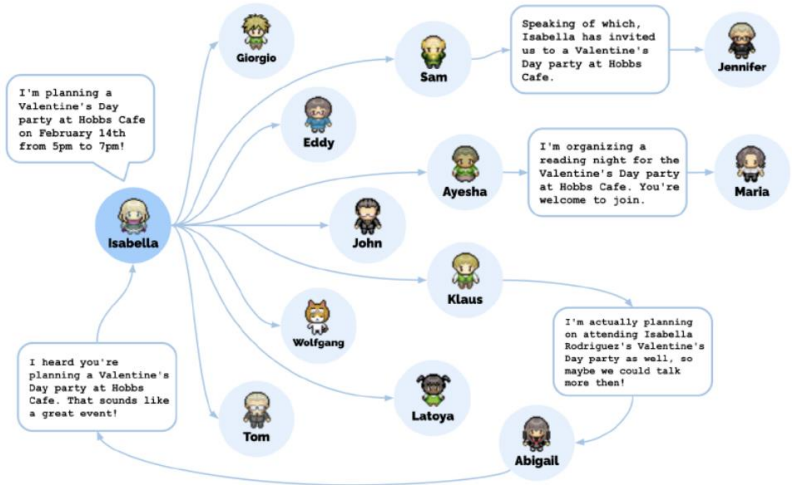
AI가 사는 마을 Generative Agent

캐릭터들이 인간과 유사한 행동을 할 수 있게 환경과 캐릭터와의 상호작용 구성

< 환경과의 상호작용 – 하루 일과 예시 >



< 사회적 행동 – 정보 확산 / 관계 기억>



02

LangGarph

RAG의 한계

문서 내 질문에 대한 답변이 존재하지 않는 경우

부족한 정보를 Web 검색하여 문서에 추가하는 로직을 추가

만약 검색결과에 잘못된 정보가 포함되거나 혹은 검색 결과에 없다면?

잘못된 검색결과가 결국 Hallucination으로 이어진다면

RAG의 한계

🤔 검색이 제대로 나올 때까지 반복하여 검색 해볼까?

Hallucination을 방지하는 LLM을 추가해야 하나?

기존 전통적인 RAG의 문제점

1. 문서 업로드

Document Loader

사전에 정의된

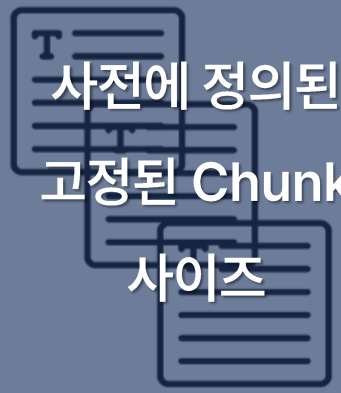
데이터 소싱 자원

(PDF, DB 등)

PyPDFLoader를 활용한 문서 가져오기

2. 문서 분할

Text Splitter



사전에 정의된

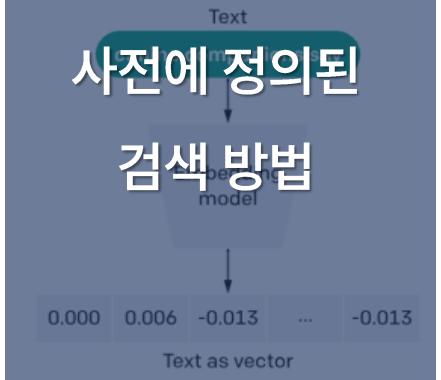
고정된 Chunk

사이즈

PDF 문서를 여러 문서로 분할

3. 문서 임베딩

Embed to Vectorstore



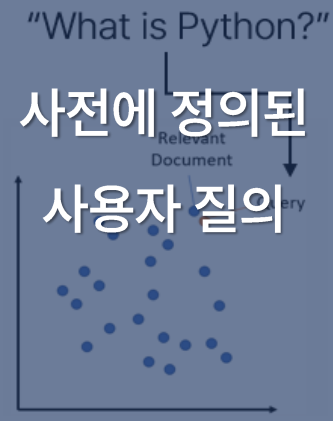
사전에 정의된

검색 방법

LLM이 이해할 수 있도록 문서 수치화

4. 임베딩 검색

VectorStore Retriever



사전에 정의된

사용자 질의

질문과 연관성이 높은 문서 추출

5. 답변 생성

QA Chain

답변 결과에 대한

문서와의 관련성

Prompt 1

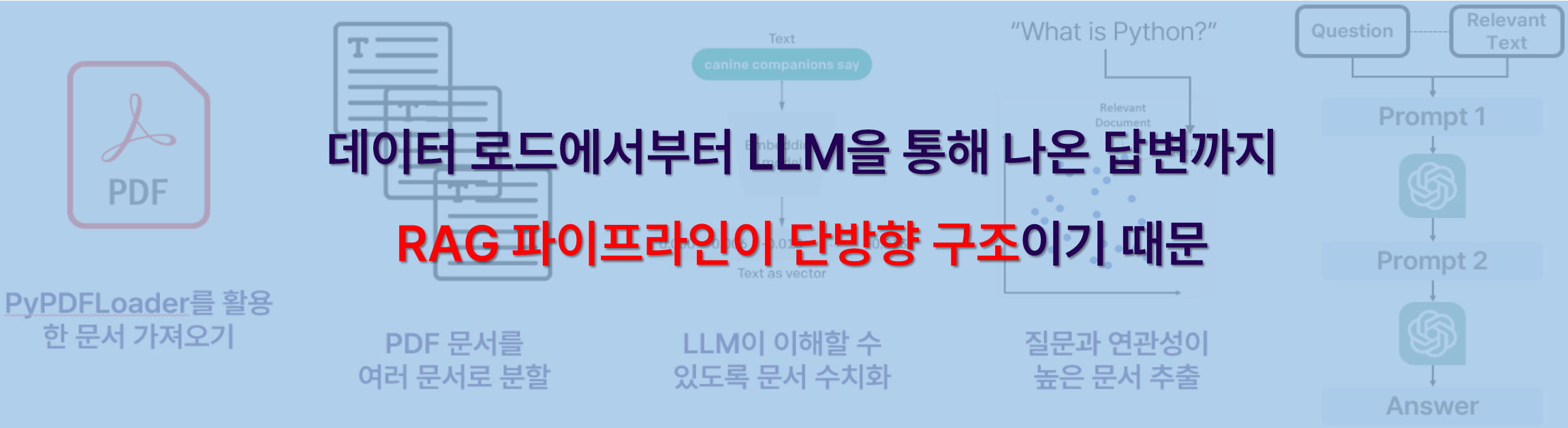
Prompt 2

신뢰하기 어려운

LLM or Agent

Answer

기존 전통적인 RAG의 문제점



데이터 로드에서부터 LLM을 통해 나온 답변까지

RAG 파이프라인이 단방향 구조이기 때문

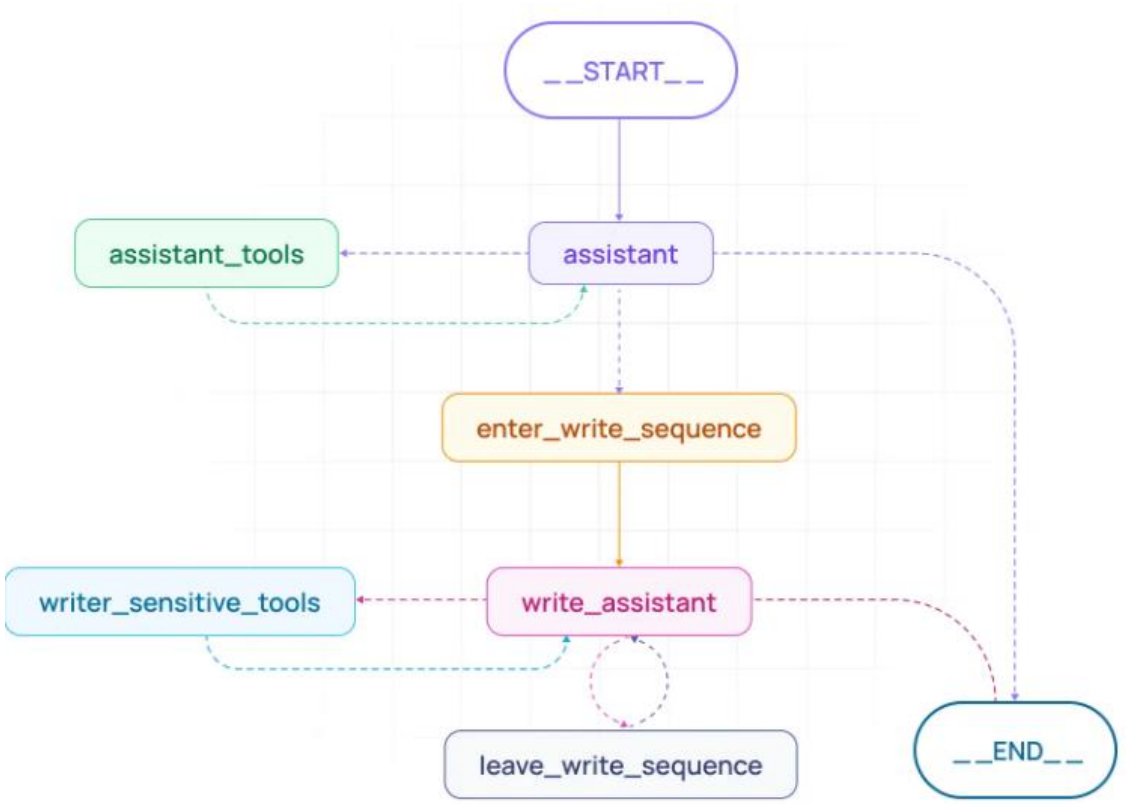
LangGraph

LangGraph은

다양한 색(state)으로 표현된 점(node)이

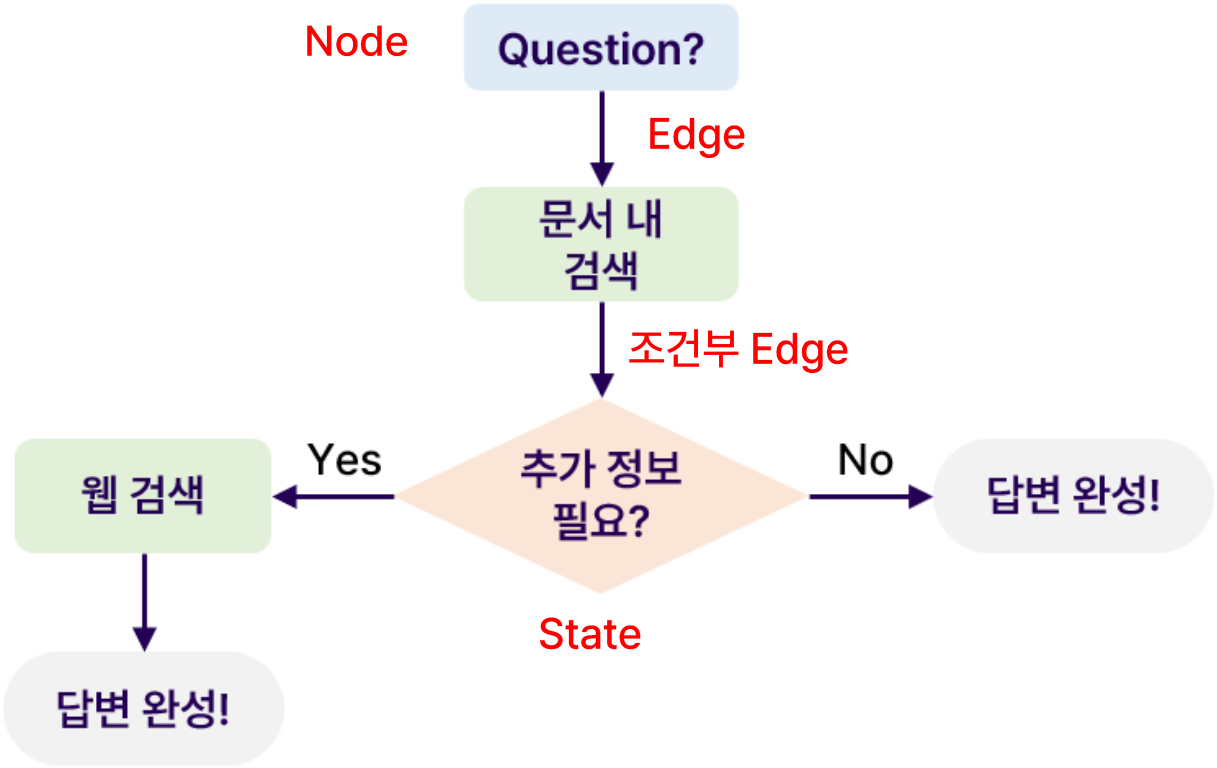
방향(sign)을 갖은 선(edge)과 만나

구성된 다이어그램(graph)이다.



source: https://www.langchain.com/langgraph

LangGraph

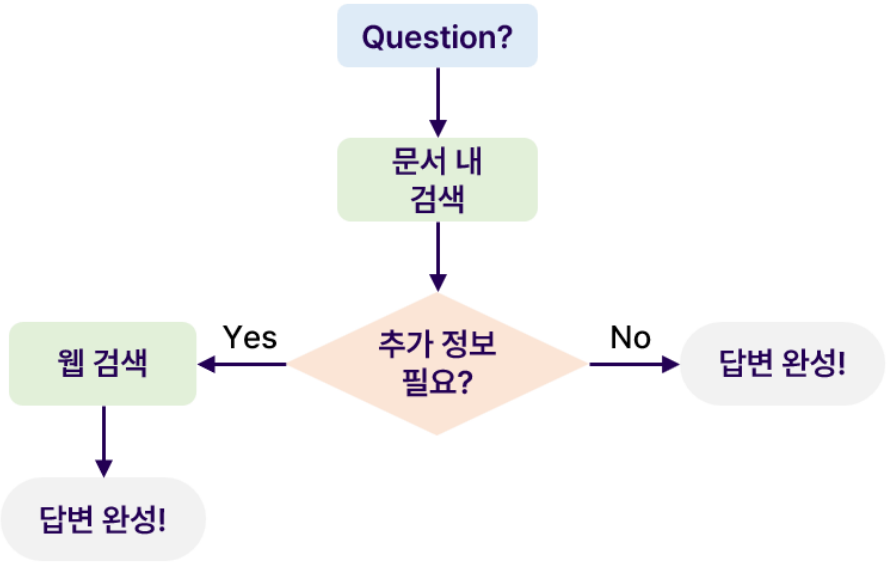


- **Node** 수행하고자 하는 작업 내용
- **Edge** 노드와 노드 사이의 연결
- **조건부 Edge** 를 통해 분기 처리
- **State** 현재의 상태 값을 저장 및 전달하는데 활용

➡️ RAG 파이프라인을 유연하게 설계 가능

source: https://www.langchain.com/langgraph

LLM 을 활용한 워크플로우에 순환(Cycle) 연산 기능을 추가하여 손 쉽게 흐름을 제어



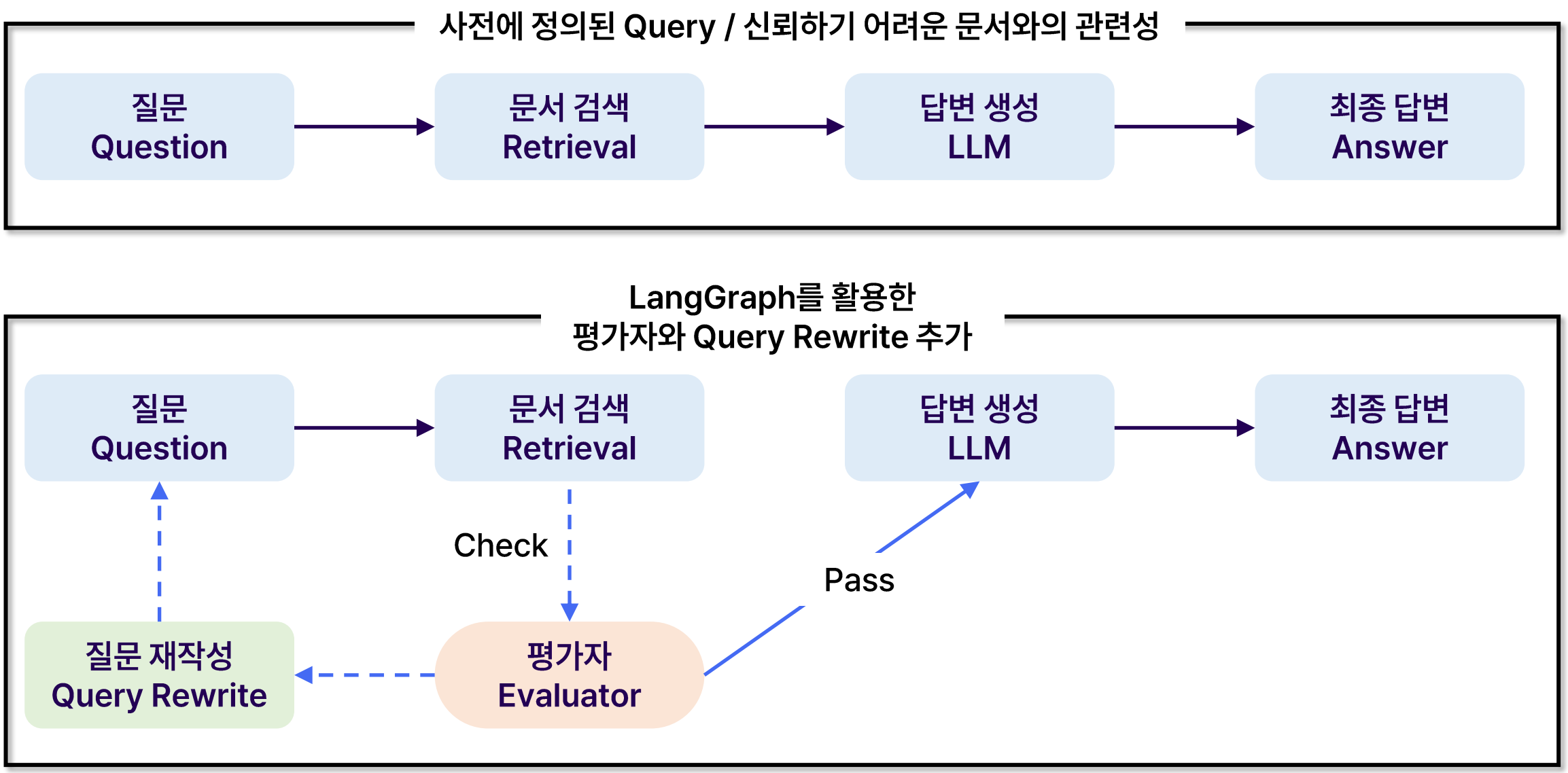
RAG 파이프라인의 세부 단계별 흐름제어가 가능

Conditional Edge: 조건부 (if, elif, else 와 같은..) 흐름 제어

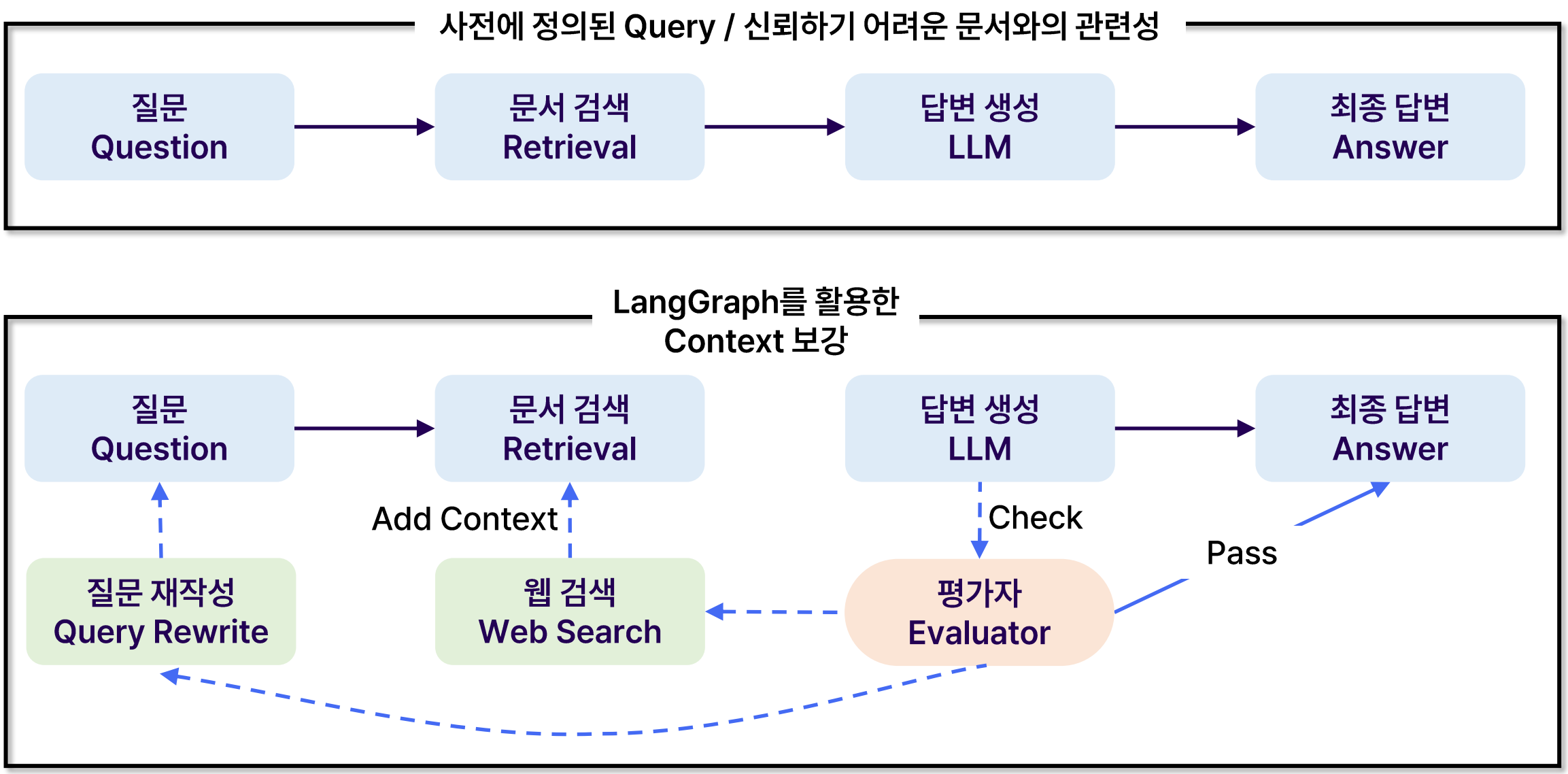
Human-in-the-loop: 필요시 중간 개입하여 다음 단계를 결정

Checkpointer: 과거 실행 과정에 대한 “수정” & “리플레이” 기능

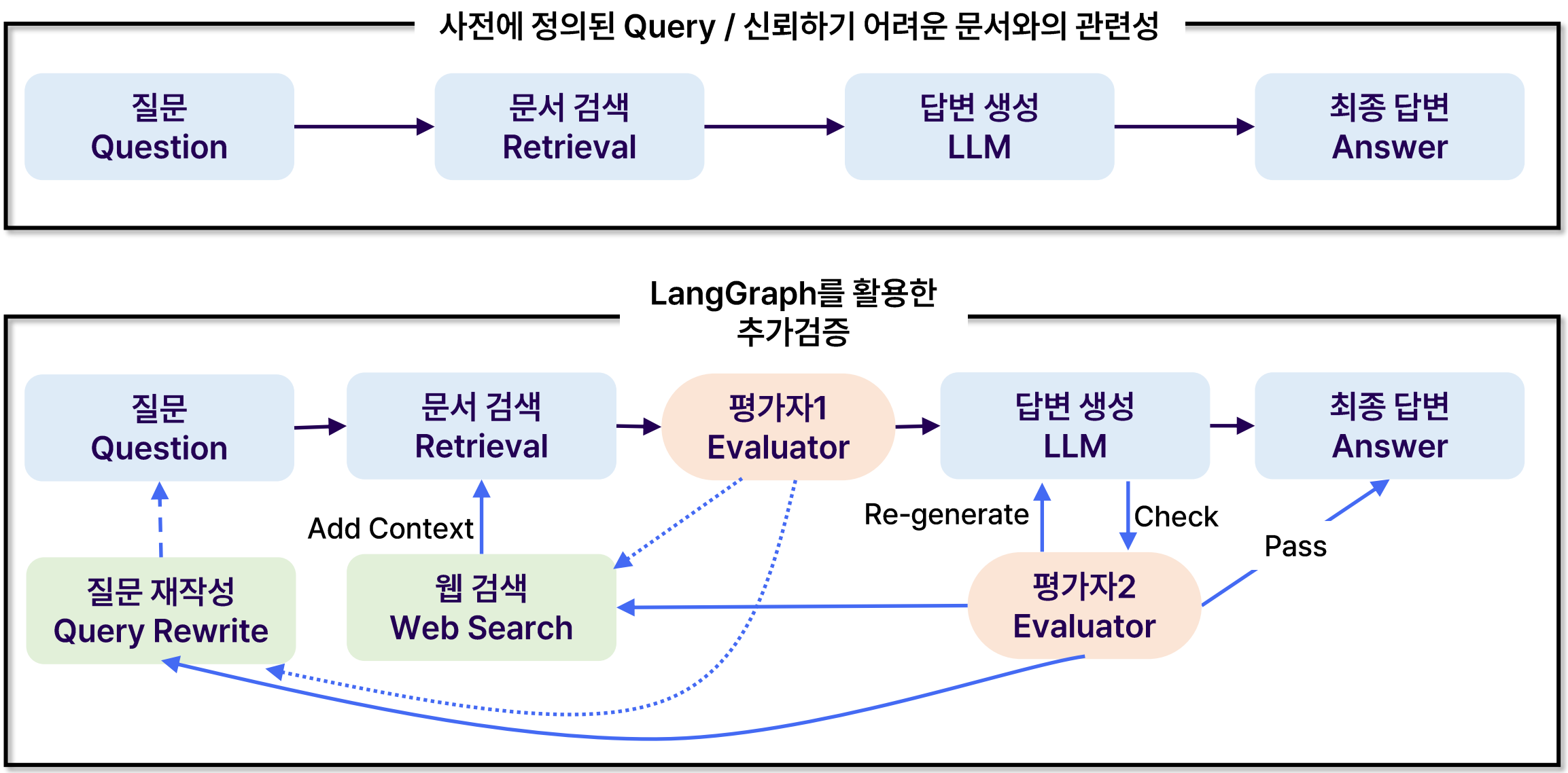
LangGraph를 활용하면?



LangGraph를 활용하면?



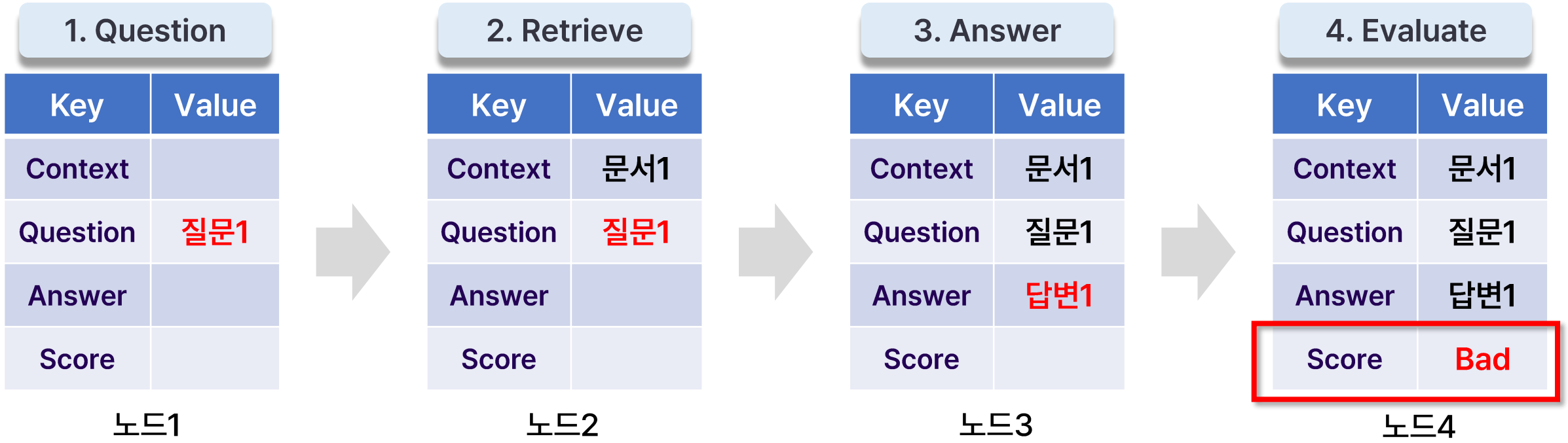
LangGraph를 활용하면?



상태(State)

: 노드와 노드 간 정보 전달 시 State 객체에 담아 전달

Dictionary 형태와 유사하고 새로운 노드에 값을 덮어쓰는 방식(Overwrite)으로 진행



노드1: 질문 재작성 요청

5. Question

Key	Value
Context	문서1
Question	질문2
Answer	답변1
Score	Bad

노드1

6. Retrieve

Key	Value
Context	문서2
Question	질문2
Answer	답변1
Score	Bad

노드2

7. Answer

Key	Value
Context	문서2
Question	질문2
Answer	답변2
Score	Bad

노드3

8. Evaluate

Key	Value
Context	문서2
Question	질문2
Answer	답변2
Score	Good

노드4

Question Transform: 질문 재작성

노드2: 문서 검색을 재요청

1. Question

Key	Value
Context	
Question	질문1
Answer	
Score	

노드1

5. Retrieve

Key	Value
Context	문서2
Question	질문1
Answer	답변1
Score	Bad

노드2

6. Answer

Key	Value
Context	문서2
Question	질문1
Answer	답변2
Score	Bad

노드3

7. Evaluate

Key	Value
Context	문서2
Question	질문1
Answer	답변2
Score	Good

노드4

Context Retrieval 조정
(Chunk, 다른 검색기, Web Search, ...)

노드3: 답변 재생성 요청

1. Question

Key	Value
Context	
Question	질문1
Answer	
Score	

노드1

2. Retrieve

Key	Value
Context	문서1
Question	질문1
Answer	
Score	

노드2

5. Answer

Key	Value
Context	문서1
Question	질문1
Answer	답변2
Score	Bad

노드3

6. Evaluate

Key	Value
Context	문서1
Question	질문1
Answer	답변2
Score	Good

노드4

프롬프트 조정 & 다른 LLM 사용
(GPT, Claude, ...)

03

올라마 (Ollama)

로컬 LLM을 사용하는 이유

Ollama

LM studio

1. **개인정보 보안:** 인터넷 연결 없이도 입력한 데이터가 컴퓨터 외부로 반출되지 않아 정보 보호 가능
2. **사용자 맞춤 옵션:** CPU 사용량, 대화의 길이, 온도(모델의 창의성 조절) 등 다양한 기능을 사용자 맞춤으로 조정 가능
3. **비용 절감:** 월 구독료 없이 무료로 사용 가능
4. **연결 문제 해결:** 인터넷 연결 없이도 사용 가능하기 때문에 연결 상태에 영향 받지 않음

각광 받는 LLM 모델

ChatGPT-4o

Hello GPT-4o

- OpenAI에서 개발
- 다양한 언어와 주제에 대해 높은 수준의 대화 가능

Claude

Claude

- AI 스타트업 엔트로픽에서 개발
- 사용자의 의도를 잘 파악하고, 안전하고 윤리적인 대화 가능
- 사용자 친화적 대화와 복잡한 문제 해결 가능

Llama

- Meta에서 개발
- 자원 효율성이 높아서 작은 하드웨어에서도 우수한 성능
- 오픈 소스 LLM

로컬 LLM 도구 Ollama

대규모 언어 모델(LLM)을

로컬 머신 상에서 실행하기 위한 강력한 도구

Models

다양한 모델 보유

Filter by name...

Featured

llama3.2

Meta's Llama 3.2 goes small with 1B and 3B models.

Tools 1B 3B

1.1M Pulls 63 Tags Updated 3 weeks ago

llama3.1

Llama 3.1 is a new state-of-the-art model from Meta available in 8B, 70B and 405B parameter sizes.

Tools 8B 70B 405B

6.6M Pulls 94 Tags Updated 4 weeks ago

gemma2

Google Gemma 2 is a high-performing and efficient model available in three sizes: 2B, 9B, and 27B.

2B 9B 27B

source: https://ollama.com/library

Ollama 시작하기

1. 컴퓨터에 맞는 버전을 다운로드

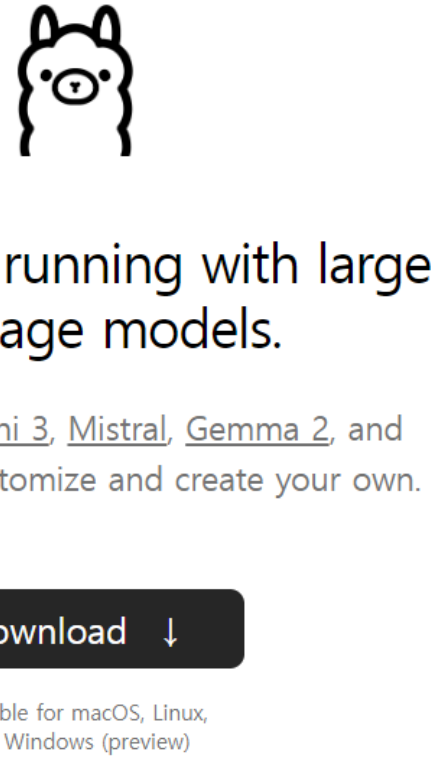
   https://ollama.com/

2. 특정 LLM을 실행하기 위해 다음을 사용하여 다운로드

   `ollama pull 모델명`

3. 모델 실행 시 run 명령어 사용

   `ollama pull llama`



source: https://ollama.com/