

LangChain

AI 서비스 모델 구축

실습 환경 설정

Anaconda 설치

Download from <https://www.anaconda.com/download>

The screenshot shows the Anaconda download page. At the top, there is a navigation bar with links for ANACONDA, Products (which is underlined), Solutions, Resources, Partners, and Company. To the right of the navigation bar are buttons for Free Download and Sign Up. Below the navigation bar, the word "Distribution" is prominently displayed in large black letters. Underneath it, the text "Free Download*" is shown. A registration form follows, asking for an email address and consent to receive communication. A "Submit" button is at the bottom of the form. A "Skip registration" link is located at the bottom right of the form. A red box highlights the "Skip registration" link, and a blue arrow points to it from the text "skip registration" on the right.

ANACONDA. [Products](#) Solutions Resources Partners Company

[Free Download](#) [Sign Up](#)

Distribution

Free Download*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- Easily search and install thousands of data science, machine learning, and AI packages
- Manage packages and environments from a desktop application or work from the command line
- Deploy across hardware and software platforms
- Distribution installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 200 employees requires a Business or Enterprise license. See Pricing

Provide email to download Distribution

Email Address:

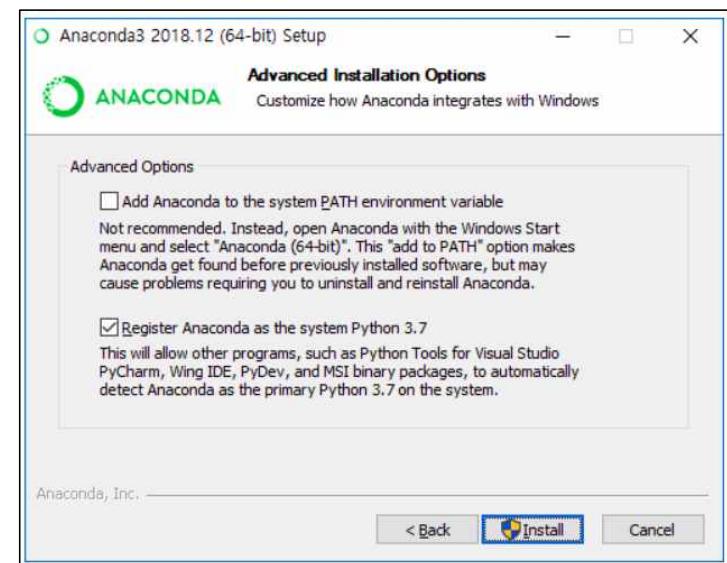
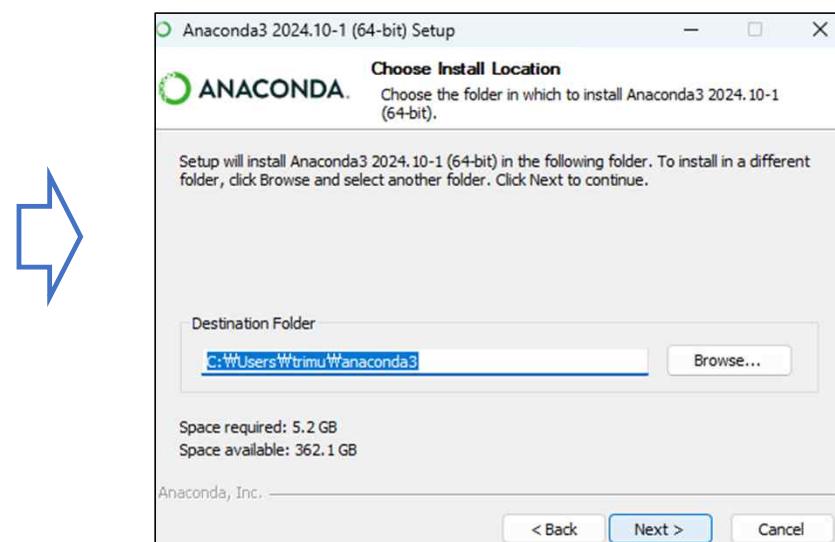
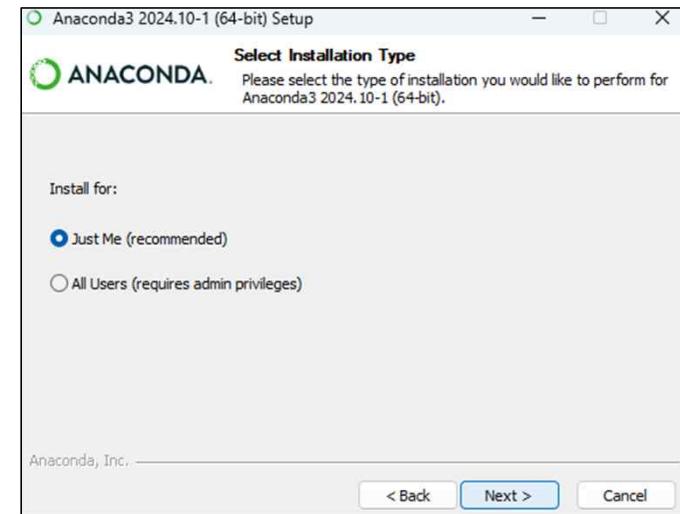
Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent [here](#) at any time.

By continuing, I agree to Anaconda's [Privacy Policy](#) and [Terms of Service](#).

[Submit >](#)

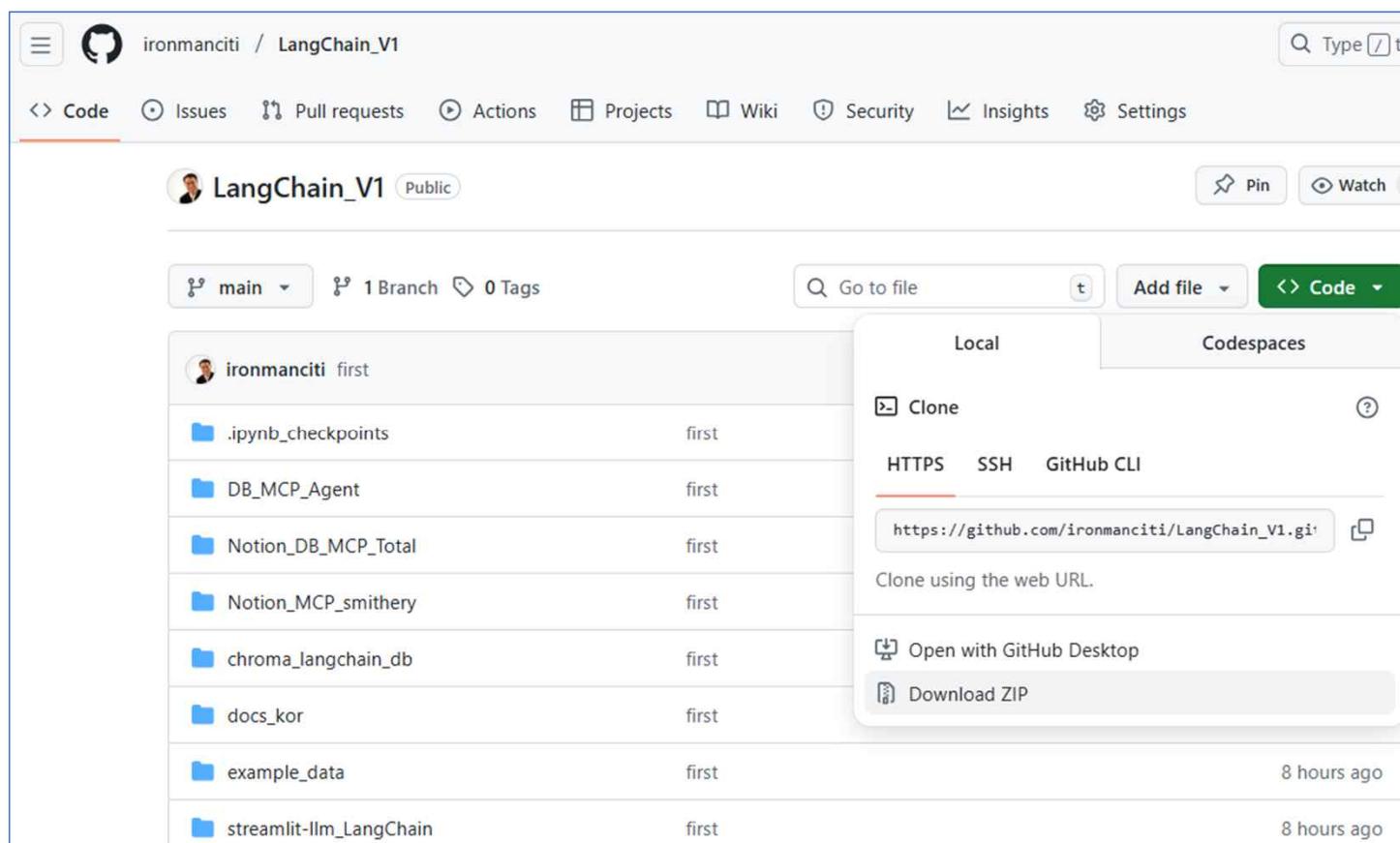
[Skip registration](#)

skip registration



Github Repository 에서 실습 code download

- https://github.com/ironmanciti/LangChain_V1



가상환경 생성 및 package 설치

```
conda env create -f environment.yml
```

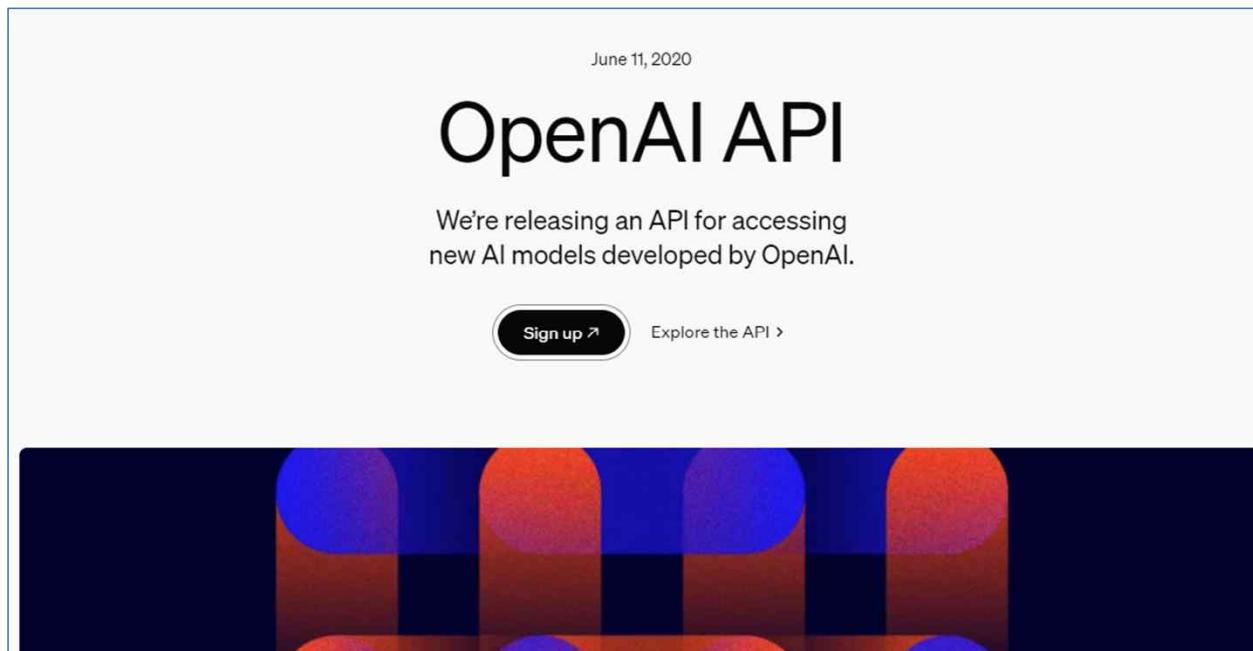
- package Install (수동 설치 경우)
 pip install langchain==1.0

```
pip install -U langchain-openai  
pip install -U langchain-anthropic  
pip install -U langchain-google-genai
```

- 다른 LLM 공급자는 [Integrations tab](#) 참조

OpenAI API Key 생성

<https://openai.com/index/openai-api/>



계정 만들기

이메일 주소* _____

계속

이미 계정이 있으신가요? [로그인](#)

_____ 또는 _____

Google로 계속하기

Microsoft 계정으로 계속하기

Apple로 계속하기

Billing 정보 등록 (settings에서 등록)

The screenshot shows the OpenAI developer platform interface. At the top, there's a navigation bar with a user icon, 'Personal' dropdown, 'Default project' dropdown, 'Playground', 'Dashboard', 'Docs', 'API reference', and a gear icon with a green checkmark. A blue arrow points from the right towards the gear icon. On the left, a sidebar titled 'GET STARTED' contains links for 'Overview' (which is highlighted in light purple), 'Quickstart', 'Models', 'Pricing', 'Changelog', and 'Terms and policies'. Below that, under 'CAPABILITIES', are links for 'Text generation', 'Vision', 'Image generation', 'Audio generation', and 'Text to speech'. The main content area features the title 'OpenAI developer platform' and a 'Developer quickstart' section with a brief description and a 5-minute estimate. To the right of this is a code snippet for Python:

```
python ◊
1  from openai import OpenAI
2  client = OpenAI()
3  completion = client.chat.completions.create(
4      model="gpt-4o",
5      store=True,
6      messages=[
7          {"role": "user", "content": "write a haiku about ai"}
8      ]
9  )
```

At the bottom of the main content area, there are 'Meet the models' and 'Pricing ↗' links.

Billing 정보 등록 (Settings → Billing)

The screenshot shows the Billing settings page. On the left sidebar, under the SETTINGS section, the 'Billing' option is selected. In the main content area, the 'Overview' tab is active. The page displays a balance of \$10.50 and a note about auto-recharge being turned on. A callout box highlights the auto-recharge setting with Korean text explaining it.

P Personal / Default project

Playground Dashboard Docs API reference

SETTINGS

Your profile

ORGANIZATION

General

API keys

Admin keys

Members

Projects

Billing

Limits

Usage

Data controls •

Verifications

Billing

Overview Payment methods Billing history Preferences

Pay as you go → 사용한 만큼 지불하는 방식 (Pay as you go)

Credit balance ⓘ

\$10.50

Add to credit balance Cancel plan

• 자동 충전 설정: 켜져 있으며, 크레딧 잔액이 \$5.00에 도달하면 결제 방식이 사용되어 잔액이 \$15.00로 충전됩니다.

Auto recharge is on When your credit balance reaches \$5.00, your payment method will be charged to bring the balance up to \$15.00. **Modify**

Payment methods Add or change payment method

Billing history View past and current invoices

API key 생성 (Dashboard → API keys)

<https://platform.openai.com/api-keys>

The screenshot shows the OpenAI Platform dashboard for a 'Default project'. On the left sidebar, the 'API keys' option is selected. The main content area is titled 'API keys' and contains instructions for managing API keys. A green button at the top right says '+ Create new secret key'. A large blue arrow points from the bottom of the page towards this button. Below the button, there is a table with one row showing a test key:

NAME	SECRET KEY	LAST USED	CREATED BY	PERMISSIONS
My Test Key	sk-...7XGD	Never	YoungJea Oh	All

A callout box with Korean text appears over the table and the button:

이 프로젝트의 소유자로서 귀하는 이 프로젝트의 모든 API 키를 보고 관리할 수 있습니다.
API 키를 다른 사람과 공유하거나 브라우저 또는 기타 클라이언트 측 코드에 노출하지 마십시오. 귀하의 계정
보안을 보호하기 위해 OpenAI는 공개적으로 유출된 모든 API 키를 자동으로 비활성화할 수도 있습니다.
사용량 페이지에서 API 키별 사용량을 확인하세요.

Key는 한번만 보여주므로 메모장 등에 복사한 후 .env 파일로 저장

Create new secret key

Owned by
 You Service account

이 API 키는 사용자와 연결되어 있으며 선택한 프로젝트에 대해 요청할 수 있습니다. 조직이나 프로젝트에서 제거되면 이 키가 비활성화됩니다.

Name Optional
My Test Key

Project
Default project

Permissions
[All](#) [Restricted](#) [Read Only](#)

[Cancel](#) [Create secret key](#)

Save your key

Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

`3RuDSd7TQbDNViu39T3qT3B1bkFJEoTJIwcT8PhptPA:` [Copy](#)

Permissions
Read and write API resources

[Done](#)

이 프로젝트의 소유자로서 귀하는 이 프로젝트의 모든 API 키를 보고 관리할 수 있습니다.
API 키를 다른 사람과 공유하거나 브라우저 또는 기타 클라이언트 쪽 코드에 노출하지 마십시오. 귀하의 계정 보안을 보호하기 위해 OpenAI는 공개적으로 유출된 모든 API 키를 자동으로 비활성화할 수도 있습니다.
사용량 페이지 에서 API 키별 사용량을 확인하세요.
이름 비밀키 작성자: 권한 내 테스트 키 sk-...7XGD 오영재 모두 수정 삭제

생성된 API Key 관리

- 타인 노출 안되도록 조심
- 노출되면 반드시 삭제 후 재 생성

API keys

+ Create new secret key

As an owner of this project, you can view and manage all API keys in this project.

Do not share your API key with others or expose it in the browser or other client-side code. To protect your account's security, OpenAI may automatically disable any API key that has leaked publicly.

View usage per API key on the [Usage page](#).

NAME	SECRET KEY	LAST USED ⓘ	CREATED BY	PERMISSIONS
My Test Key	sk-...7XGD	Never	YoungJea Oh	All

Anthropic API Key 생성

The image consists of two side-by-side screenshots of the Anthropic website.

Left Screenshot (Homepage):

- The title "ANTHROPIC" is at the top left.
- The main heading is "Build with Claude".
- The subtext: "Create user-facing experiences, new products, and new ways to work with the most advanced AI models on the market."
- Two buttons at the bottom: "Start building" (highlighted with a red oval) and "Developer docs".

Right Screenshot (User Profile Page):

- The top navigation bar includes "Claude", "Research", "Company", and "Careers".
- A greeting message: "Good afternoon, YoungJea".
- A sidebar on the left shows a blue profile picture and the name "YoungJea".
- A list of options:
 - > Hi [Start prompting with Claude](#)
 - 📝 [Generate a prompt](#)
 - 👤 [Invite collaborators](#)
 - 🔑 [Get API keys](#) (highlighted with a white oval)
 - 📁 [Prompt library](#)
 - 📖 [Explore documentation](#)

Anthropic Billing Plan

The screenshot shows the Anthropic Settings page with a dark theme. On the left, a sidebar lists navigation options: Your organization, Members & invites, **Plans & billing** (which is selected and highlighted in grey), Rate limits, API keys, Logs, Usage, and Cost.

In the main content area, the top navigation bar includes links for Dashboard, Workbench, Settings, Docs, Feedback, and a user icon. The "Your plan" section is highlighted with a red box. It displays "Build Plan (Tier 1)" and a note: "Monthly usage limit: US\$0 used of US\$100. Limit resets on 2024년 9월 1일." Below this is a "Credit balance" section showing "AI" and "US\$10.00" as the Remaining Balance. To the right, it says "Charged to Visa •••• 5708" and has an "Add Funds" button. A message box states: "Auto reload is enabled. We will reload to US\$50 when the balance reaches US\$10 until your tier's monthly usage limit is reached." This message is also highlighted with a red box. An "Edit Settings" button is located next to the message.

At the bottom, there are sections for "Invoice History" and "Claude Instant Usage". The "Invoice History" table shows three entries:

Date Range	Status	Amount
Aug 01 - Sep 01, 2024 (UTC)	DRAFT	\$0.00 USD
Jul 17 - Aug 01, 2024 (UTC)	FINALIZED	\$0.00 USD
Jul 17, 2024 (UTC)	ISSUED	\$0.00 USD

The "Claude Instant Usage" section indicates "No charges with usage for this product".

Set up billing

A new account may use up to US\$100 of credits in a calendar month. This is calculated based on API usage and the age of your account. The monthly usage limit will increase over time and with usage. [Learn more](#)

Initial credit purchase ⓘ

\$10

Enter an amount between US\$5 and US\$100

Auto-reload credit when balance reaches a certain threshold. Enabling reloads is recommended to avoid service disruptions.

When credit balance reaches:

\$10

Bring credit balance back up to:

\$50

Subtotal	US\$10.00
Estimated taxes	US\$1.00
Total	US\$11.00

Charged to

Visa •••• 5708

By clicking "Purchase Credits" you agree to Anthropic's [Supplemental Credit Terms](#)

 Credits

Auto-reload toggle 선택

Auto reload settings

Auto-reload credit when balance reaches a certain threshold. Enabling reloads is recommended to avoid service disruptions.

When credit balance reaches:

\$

Bring credit balance back up to:

\$

Credit card to be charged

Visa •••• 5708

Cancel **Save Settings**

API key 생성

API keys				
NAME	KEY	CREATED BY	CREATED AT	UPDATED AT
test_key1	sk-ant-api03-lr...kgAA	ironmanciti@gmail.com	2024년 7월 17일	...

.env에 저장
ANTHROPIC_API_KEY=created-api-key

Create API key

Name your key

Create Key

Gemini API KEY 생성

The screenshot shows the Google AI for Developers homepage. At the top, there is a navigation bar with links for '모델' (Model), '솔루션' (Solution), '코드 지원' (Code Support), '쇼케이스' (Showcase), and '커뮤니티' (Community). A search bar and a language selector ('한국어') are also present. A banner at the top states '이제 Gemini 2.0 Flash 실험 버전을 사용할 수 있습니다.' (Now you can use the Gemini 2.0 Flash experimental version) with a '자세히 알아보기' (Learn more) button. Below the banner, a note says 'translated by Google' and '이 페이지는 Cloud Translation API를 통해 번역되었습니다.' (This page was translated using the Cloud Translation API). A 'Switch to English' button is available. The main content features a large blue circle graphic with the text 'Gemini API 시작하기' (Get started with Gemini API). Below this, a description reads: 'Google AI Studio는 Google의 차세대 멀티모달 생성형 AI 모델 제품군인 Gemini를 사용해 빌드할 수 있는 가장 빠른 방법입니다.' (Google AI Studio is the fastest way to build using Google's next-generation multi-modal generative AI model product line, Gemini). At the bottom, a blue button with the text 'Google AI Studio에 로그인' (Log in to Google AI Studio) is highlighted with a red rectangular border.

Google AI Studio

API 키 가져오기

Get API key

Create new prompt

New tuned model

My library

Allow Drive access

Getting started

Documentation

Prompt gallery

Gemini cookbook

Discourse forum

Build with Vertex AI on Google Cloud

API 키

아직 프로젝트가 없으면 새 프로젝트를 만들거나 기존 프로젝트에 API 키를 추가할 수 있습니다. 모든 프로젝트에는 새 프로젝트를 만들 때 동의하는 Google Cloud Platform 서비스 약관이 적용되며, Gemini API 및 Google AI Studio 사용 시에는 Gemini API 서비스 약관이 적용됩니다.

API 키는 안전하게 사용합니다. 다른 사람이 볼 수 있는 코드에 키를 공유하거나 삽입하지 마세요.

결제가 사용 설정된 프로젝트에서 Gemini API를 사용하는 경우 사용한 만큼만 지불하는 가격 책정 방식이 적용됩니다.

API 키 만들기

프로젝트 번호	프로젝트 ID	API 키	생성일	요금제
...4627	Generative Language Client	...Pwxg	2024. 7. 18.	무료 결제 설정 사용 데이터 보기

.env에 저장

GOOGLE_API_KEY=created-api-key

Google AI Studio

Studio ▾

Dashboard ^

API keys

Usage & Billing

Changelog

Documentation

Get API key

View status

Settings

API 키

Gemini API를 빠르게 테스트하세요

API 빠른 시작 가이드

Code

```
curl "https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent" \
-H 'Content-Type: application/json' \
-H 'X-goog-api-key: GEMINI_API_KEY' \
-X POST \
-d '{
  "contents": [
    {
      "parts": [
        {
          "text": "Explain how AI works in a few words"
        }
      ]
    }
  ]
}'
```

API 키는 아래와 같습니다. 또한 Google Cloud에서 프로젝트와 API 키를 확인하고 관리할 수 있습니다.

Look up API Key for project

프로젝트 번호	프로젝트 이름	API 키	생성일	요금제
...9728	Demo1	...odg0	2025. 7. 28.	Free 결제 설정 사용 데이터 보기



Google Cloud Demo-1 리소스, 문서, 제품 등 검색(/) 검색

API 사용자 인증 정보 + 사용자 인증 정보 만들기 ▾ 삭제 ↵ 삭제된 사용자 인증 정보 복원

사용 설정한 API에 액세스하려면 사용자 인증 정보를 만드세요. [자세히 알아보기](#)

API 키

이름	Bound account	생성일	제한사항
<input type="checkbox"/> ● MCP Test Key	—	2025. 9. 6.	YouTube Data API v3 ...
<input type="checkbox"/> ● Generative Language API Key	—	2025. 7. 28.	Generative Language API ...

OAuth 2.0 클라이언트 ID

이름	생성일	유형	클라이언트 ID
<input type="checkbox"/> 웹 클라이언트 1	2025. 7. 28.	웹 애플리케이션	91981399728-0

서비스 계정

이름
<input type="checkbox"/> 이메일

표시할 서비스 계정이 없습니다.

프로그램 내부에서 API Key 직접 사용 방법

- API library 설치 → pip install openai
- Program 내에서 직접 API Key assign

```
openai.api_key = "xxxxxxxx.....xxx"
```

- Key 노출 가능성이 크므로 가급적 사용 않음

API Key 환경 설정 방법

- pip install --upgrade openai
- API key 설정 (Mac OS)
 - 터미널 열기
 - Bash 프로필 편집
nano ~/.bash_profile 또는 (최신 MacOS 버전의 경우) nano ~/.zshrc
 - 환경 변수 추가 : 편집기에서 아래 줄 추가
`export OPENAI_API_KEY='your-api-key-here'`

API key 설정 (Windows 11)

- 현재 세션에만 설정 방법

setx OPENAI_API_KEY "your-api-key-here"

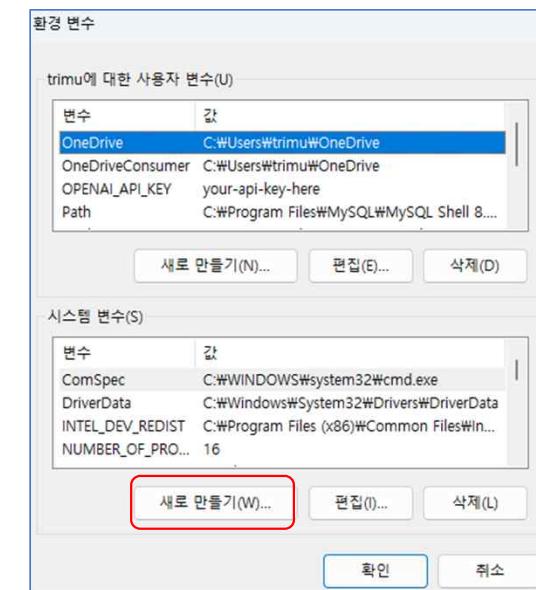
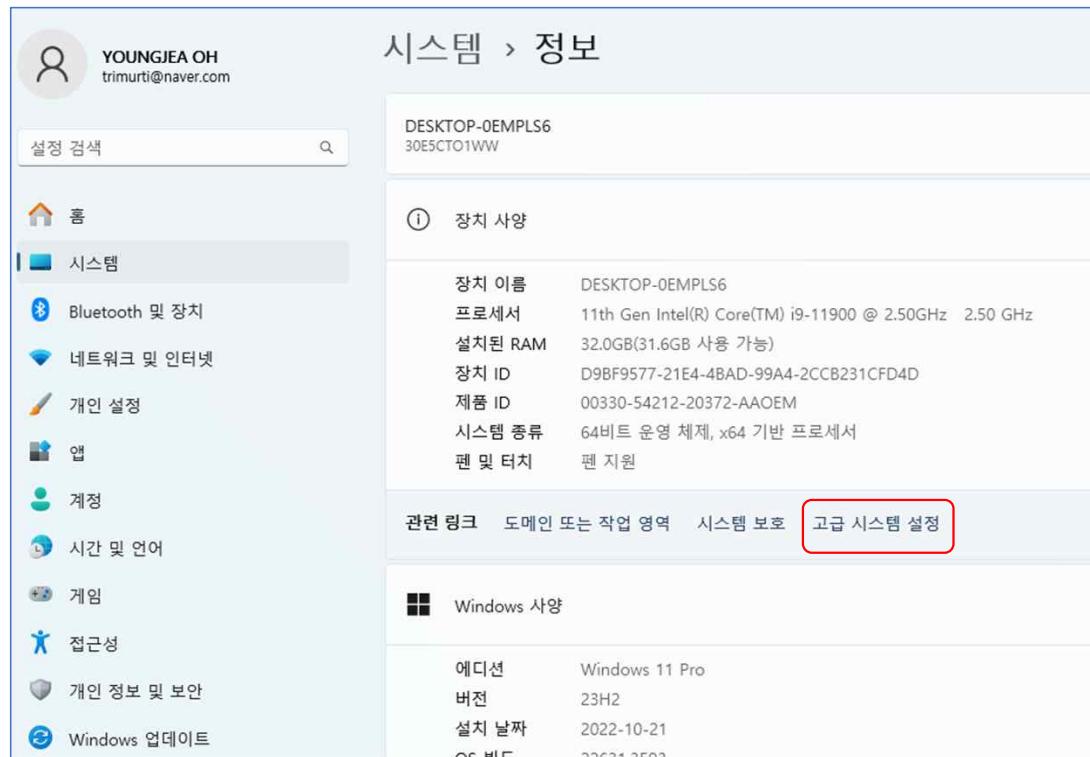
(* 현재 세션과 단일 프로젝트 .env 중복될 경우 현재 세션이 우선함)

- 환경 파일로 영구 설정 방법

설정 -> 시스템 -> 정보 -> 고급시스템설정 -> 환경변수 -> 시스템변수 -> 새로만들기
-> 변수 이름으로 OPENAI_API_KEY를 입력하고 변수 값으로 API 키를 입력

확인 : echo %OPENAI_API_KEY%

환경 파일로 영구 설정 방법 (windows 11)



단일 프로젝트에 대해 API key 설정 방법

- .env – API 키가 포함된 로컬 파일 생성

```
# .env 파일  
OPENAI_API_KEY=sk-vvDtll*****XiiIEpdjLhBJaH0f
```

- .gitignore 에 .env 파일 포함

```
# .env 파일을 git에서 무시  
.env
```

- Python code

```
pip install python-dotenv
```

```
from dotenv import load_dotenv, find_dotenv  
_ = load_dotenv(find_dotenv()) # local .env file을 읽어서 os.environ에 OPENAI_API_KEY 추가
```

```
from openai import OpenAI  
client = OpenAI() # os.environ.get("OPENAI_API_KEY")을 default로 사용하여 API Key 이용
```

LangChain

LangChain이란?

- 대규모 언어 모델(LLM)을 활용한 애플리케이션을 개발하기 위한 프레임워크
- LLM 애플리케이션의 전체 라이프사이클을 간소화
- Python 및 TypeScript 지원
- LangChain
 - 원하는 모델 제공자(model provider)를 사용해 에이전트를 빠르고 손쉽게 구축할 수 있도록 도와줍니다.
- LangGraph
 - 에이전트의 모든 단계를 세밀하게 제어할 수 있는 저수준 오케스트레이션(low-level orchestration), 메모리(memory), Human-in-the-loop(인간 개입) 기능을 제공

LangChain History

버전	출시 시기	주요 언어 지원	비고
v0.0	2022년 10월	Python	초기 프로토타입 버전 — LangChain 첫 공개
v0.1	2024년 1월	Python, JavaScript	안정화 및 JS SDK 공개 (LangChain.js 시작)
v0.2	2024년 5월	Python, JavaScript	구조 개편, 통합 인터페이스, LangGraph 도입 – 실험적 프로젝트
v0.3	2024년 9월	Python, TypeScript	완전한 모듈화 구조, TypeScript 정식 도입 LangGraph 를 내부 핵심 엔진으로 통합
v1.0	2025년 10월	Python, TypeScript	LangGraph는 LangChain 1.0의 기본 런타임 구조로 통합되었으며, 모든 에이전트 실행은 Graph 기반의 상태(StateGraph)로 처리

LangChain 주요 개념

1. **Chat models** - 메시지의 시퀀스를 입력으로 받아 하나의 메시지를 출력하는 채팅 API를 통해 노출되는 LLM (gpt-4o, claude-3, gemini, 등)을 의미
2. **메시지(Messages)** – 채팅 모델의 입력과 출력
3. **Chat History (대화 기록)** - 사용자 메시지와 모델 응답이 번갈아 가며 나타나는 메시지 시퀀스로 표현된 대화 기록
4. **Tools (도구)** - 언어 모델에 전달할 호출할 함수 이름(name), 설명(description), 인자(arguments)를 정의하는 스키마를 포함하는 기능
5. **도구 호출(Tool Calling)** - 메시지와 함께 도구 스키마를 입력으로 받아, 도구 호출 결과를 출력 메시지의 일부로 반환하는 채팅 모델 API 유형
6. **구조화된 출력(Structured Output)** - 채팅 모델이 지정된 스키마(JSON 등)에 맞는 구조화된 형식으로 응답하도록 하는 기술

7. **메모리(Memory)** - 대화의 정보를 저장(Persisted)하여 향후 대화에서 사용할 수 있도록 유지
8. **다중 모달리티(Multimodality)** - 텍스트, 오디오, 이미지, 비디오 등 다양한 형태의 데이터를 처리하는 능력
9. **Runnable** – invoke, batch, stream method로 실행 가능한 객체
10. **스트리밍(Streaming)** - 결과가 생성되는 즉시 부분적으로 출력
11. **Document loaders (문서 로더)** - 다양한 소스에서 문서를 불러오는 역할
12. **Retrievers (검색기)** - 질의(Query)에 대한 관련 문서를 지식 베이스에서 반환
13. **검색 증강 생성(Retrieval Augmented Generation, RAG)**
 - 언어 모델이 외부 지식 베이스와 결합하여 성능을 향상시키는 기술
14. **에이전트(Agents)** - 언어 모델이 작업 순서를 결정하고 외부 도구와 상호작용하여 목표를 달성

- 15. Prompt Templates** - 사용자 입력을 언어 모델에 전달할 수 있는 형식으로 포맷
- 16. Output parsers** - LLM의 출력을 받아 더 구조화된 형식으로 변환
- 17. Text splitters (텍스트 분할기)** - 문서를 검색에 사용할 수 있는 여러 조각으로 나누는 역할
- 18. Embedding models (임베딩 모델)** - 텍스트 조각을 받아 숫자 벡터로 변환
- 19. Vector stores (벡터 스토어)** - 임베딩을 효율적으로 저장하고 검색할 수 있는 DB
- 20. Indexing (인덱싱)** - 벡터 스토어를 기본 데이터 소스와 동기화 상태로 유지하는 과정

LangChain의 장점과 단점

- 장점

- 오픈 소스이므로 누구나 자유롭게 이용 가능하고 빠른 업데이트 가능
- 다양한 LLM에 표준화된 인터페이스 지원 → LLM 공급자 전환 용이
- 모듈화된 설계: 프롬프트 템플릿, 문서 로더, 출력 파서 등 다양한 모듈 제공
- 관찰 가능성 및 평가 기능 제공 (LangSmith)

- 단점

- 언어 지원 제한: Python 및 TypeScript 이외의 언어 지원 부족
- 복잡한 초기 설정과 다양한 옵션으로 초보자에게 진입 장벽
- 복잡한 기능과 다양한 옵션으로 인해 학습 곡선이 가파를 수 있음
- 버전간 코드 차이가 크며 공식 문서가 최신 코드와 불일치 하는 경우 있음
- API 업데이트 반영 지연: OpenAI 등의 새로운 기능이 즉각 반영되지 않을 수 있음

Chat Models (채팅 모델)

- 대규모 언어 모델(Large Language Models, LLMs)은 특정 작업에 대한 별도의 미세 조정 없이도 다양한 시나리오에서 우수한 성능을 발휘
- 채팅 모델 – 채팅에 특화된 LLM
- 메시지 목록을 입력으로 받아 메시지를 출력. 다음과 같은 추가 기능 제공:
 - 도구 호출: 외부 서비스, API, 데이터베이스와 상호 작용할 수 있는 도구 호출 API 제공
 - 구조화된 출력: JSON과 같은 구조화된 형식으로 응답하도록 하는 기술
 - 멀티모달 처리: 텍스트 외에 이미지, 오디오, 비디오 등을 처리할 수 있는 능력

대표적 Chat Models

- OpenAI – GPT-5 / Mini / Nano 등
- Google – Gemini 2.5 Pro / Flash / Flash-Lite 등
- Anthropic – Claude Opus 4.1 , Claude 4 / Sonnet 4 / Haiku등
- xAI – Grok 4 / Grok 4 Heavy 등
- Naver -HyperCLOVA X / CLOVA X 등
- 기타 - <https://python.langchain.com/docs/integrations/chat/>

Prompt

- LLM에 넣어 주는 텍스트 입력을 의미
- model을 프로그래밍할 수 있도록 여러가지 구성이 가능한 기능 제공
- 프롬프트의 주요 구성 요소:
 - 지시사항(Instruction): 모델이 수행해야 할 작업에 대한 명확한 설명.
 - 맥락(Context): 모델이 더 나은 답변을 제공하도록 돋는 배경 정보.
 - 입력 데이터(Input Data): 해결하려는 문제나 질문에 대한 구체적인 데이터.
 - 출력 지시자(Output Indicator): 원하는 결과물의 형식이나 구조에 대한 명시.

Messages

- 메시지 구성 요소
 - Role (역할) → 메시지의 유형 식별, 예: system, user, assistant 등
 - Content (내용) → 메시지의 실제 내용 (텍스트, 이미지, 오디오 등)
 - Metadata (메타데이터) → 선택적(optional) 필드.
응답 정보(response info), 메시지 ID, 토큰 사용량(token usage) 등

메시지 유형 (Message types)

- System message (시스템 메시지)
→ 모델이 어떻게 행동해야 하는지 지시하고, 상호작용의 맥락(context) 을 제공
- Human message (사용자 메시지)
→ 사용자의 입력을 나타내며, 모델과의 대화(interaction) 를 구성
- AI message (AI 메시지)
→ 모델이 생성한 응답으로, 텍스트 내용(text content) 뿐 아니라 도구 호출(tool calls) 및 메타데이터(metadata) 를 포함할
- Tool message (도구 메시지)
→ 모델이 호출한 도구의 실행 결과(outputs)

실습: 010. Models & Messages

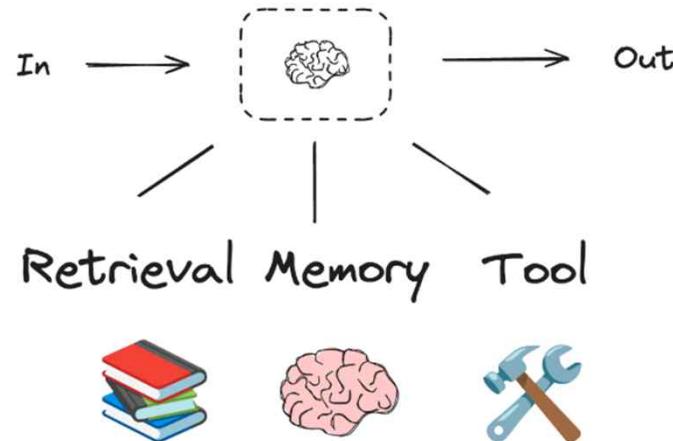
- 언어 모델 사용하기
- 메시지
 - text prompt 형식
 - dictionary 형식
- 메시지 유형
 - System Message
 - Human Message
 - AI Message
 - Tool Message
- 도구 호출
 - Token 사용량 계산
 - Batch interface
 - Tool Message

Agent 란 ?

- LLM을 핵심 엔진으로 사용하여 주어진 목표를 달성하기 위해 독립적으로 작업(추론, 판단, 실행, 피드백)을 수행하는 인공지능 시스템
- 주로 LLM(대규모 언어 모델)의 능력을 기반으로 동작하며, 사용자의 명령을 이해하고, 판단하며, 실행
- 즉, LLM은 두뇌, Agent는 이 두뇌를 활용해 행동을 실행하는 작업자 역할
- 활용 사례
 - 고객 지원 챗봇: 전자상거래 플랫폼에서 환불 요청을 처리하거나 배송 상태 확인
 - 연구 도우미: 과학 논문에서 특정 주제에 대한 핵심 내용 추출
 - 개인 비서: 이메일 초안을 작성하고, 미팅 일정 조율
 - 데이터 분석 에이전트: 주식 시장 데이터를 분석해 투자 전략 제안
 - 교육용 튜터: 수학 문제 풀이 과정을 단계별로 설명

Agent의 핵심 기술 스택

- LLM (예: GPT, Claude): 언어 이해 및 생성
- LangChain / LangGraph: 에이전트 워크플로우 및 상태 관리
- Vector Database (예: Chroma, Pinecone): 정보 저장 및 검색
- Memory: 상태 유지 및 맥락 관리
- API 통합 (예: Tavily, SerpAPI): 외부 도구와 연결

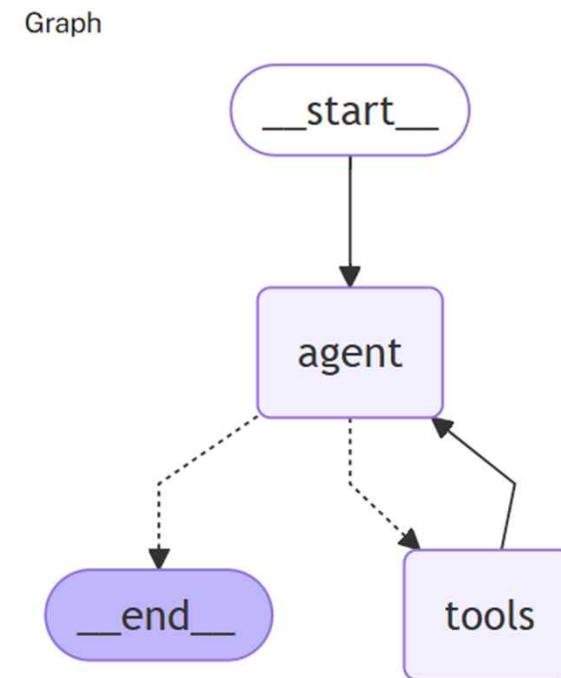
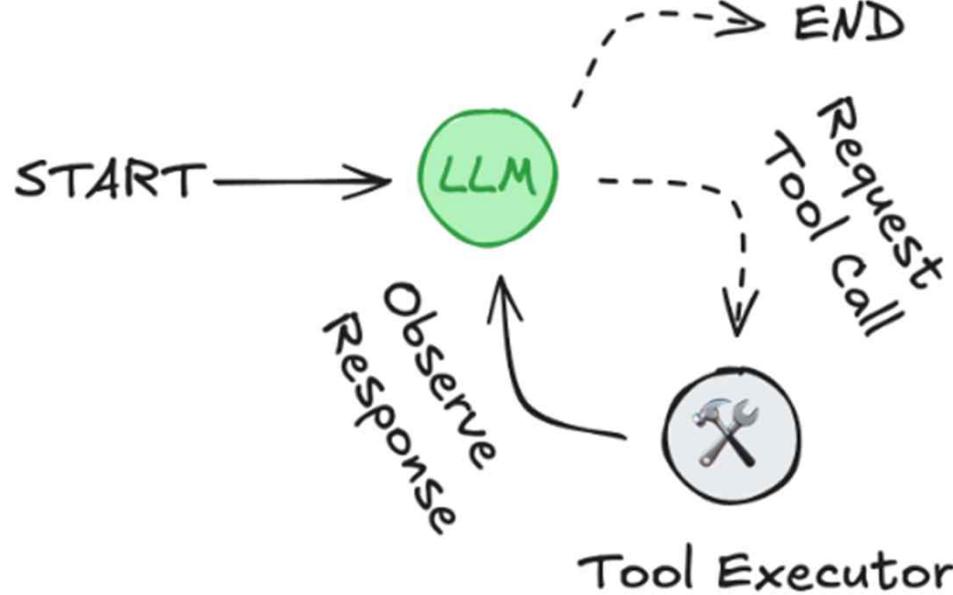


ReAct (Reasoning + Acting)

- LLM이 추론(Reasoning)과 행동(Acting)을 결합하여 더 강력한 성능을 발휘하도록 하는 프레임워크. 즉, LLM이 논리적으로 사고하면서 필요한 경우 외부 도구를 활용할 수 있도록 설계된 방법론
- Reasoning (추론): 자연어 기반으로 문제 해결을 위한 논리적 사고 수행
- Acting (행동): API 호출, 데이터베이스 검색, 웹 검색 등 외부 도구를 활용하여 필요한 정보 획득
- 예) 내가 사는 곳의 날씨를 알려줘
 - Reasoning(추론) - 사용자의 위치를 알아야 하고, 날씨 API에서 날씨 정보를 가져와야 한다. (LLM)
 - Acting(행동) - TavilySearchResults API 호출 (LangChain)
 - Observation(관찰) - API로부터 받은 정보를 LLM에 다시 전달. (LangChain)
 - Reasoning(추론) – 추가적인 정보를 이용하여 추론 (LLM)
 - Answer(응답) - "서울의 날씨는 맑음입니다." (LLM)

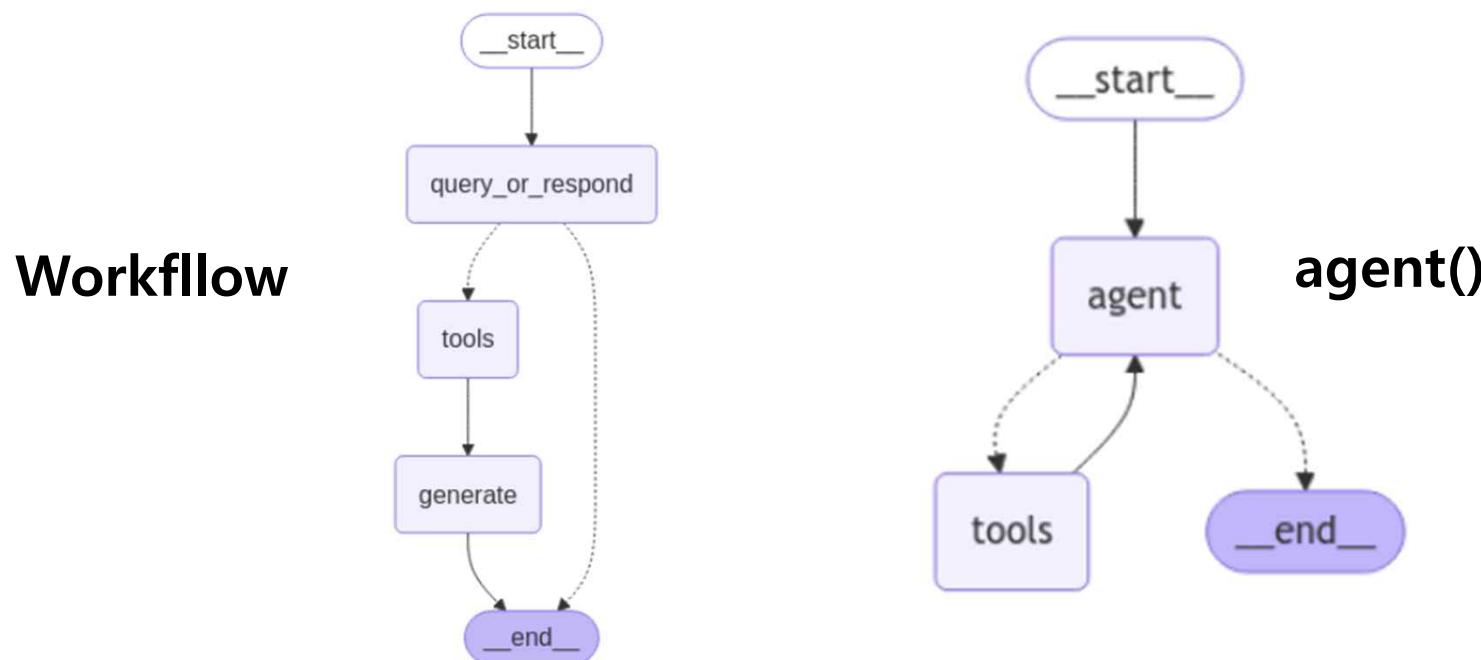
에이전트 루프(Agent loop)

- LLM이 도구들을 선택하고, 그 도구들의 출력을 활용하여 사용자의 요청을 수행



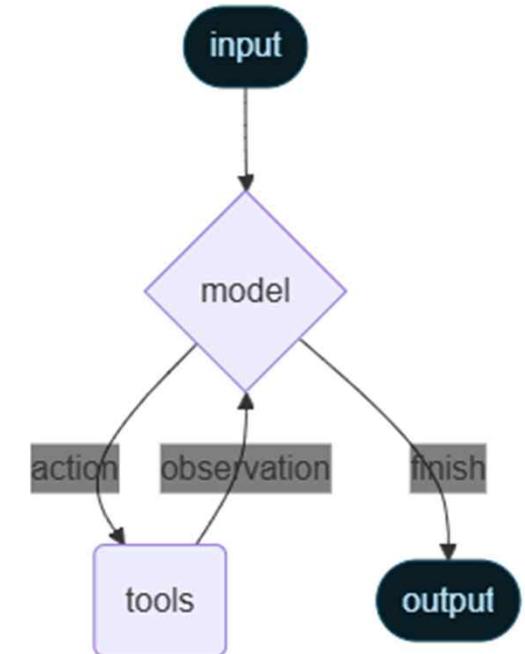
Workflow vs. Agent

- Workflow – 고정된 작업 흐름
- 에이전트(Agents) - LLM이 필요에 따라 여러 번의 검색 단계 (다중 단계 검색)를 수행하도록 자유롭게 설정 가능



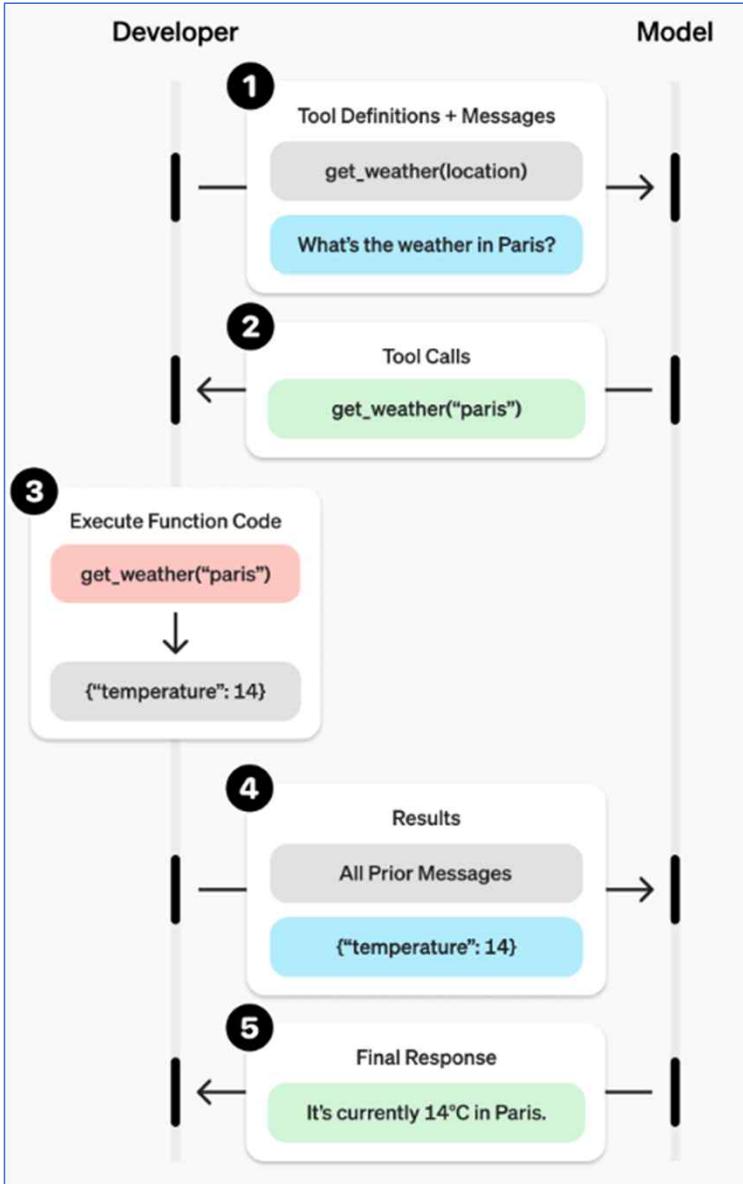
실습: 011_Agent Overview

- LLM 연결 – OpenAI, Gemini 등
- ReAct Agent 생성
- 도구(Tools) 추가
- 동적 모델 선택 (Dynamic Model)
- 도구 오류 처리 (Tool error handling)
- System prompt 제공 – 정적/동적 system prompt
- Agent 호출
- 구조화된 출력 (Structured Output)



LLM의 함수 호출 방법

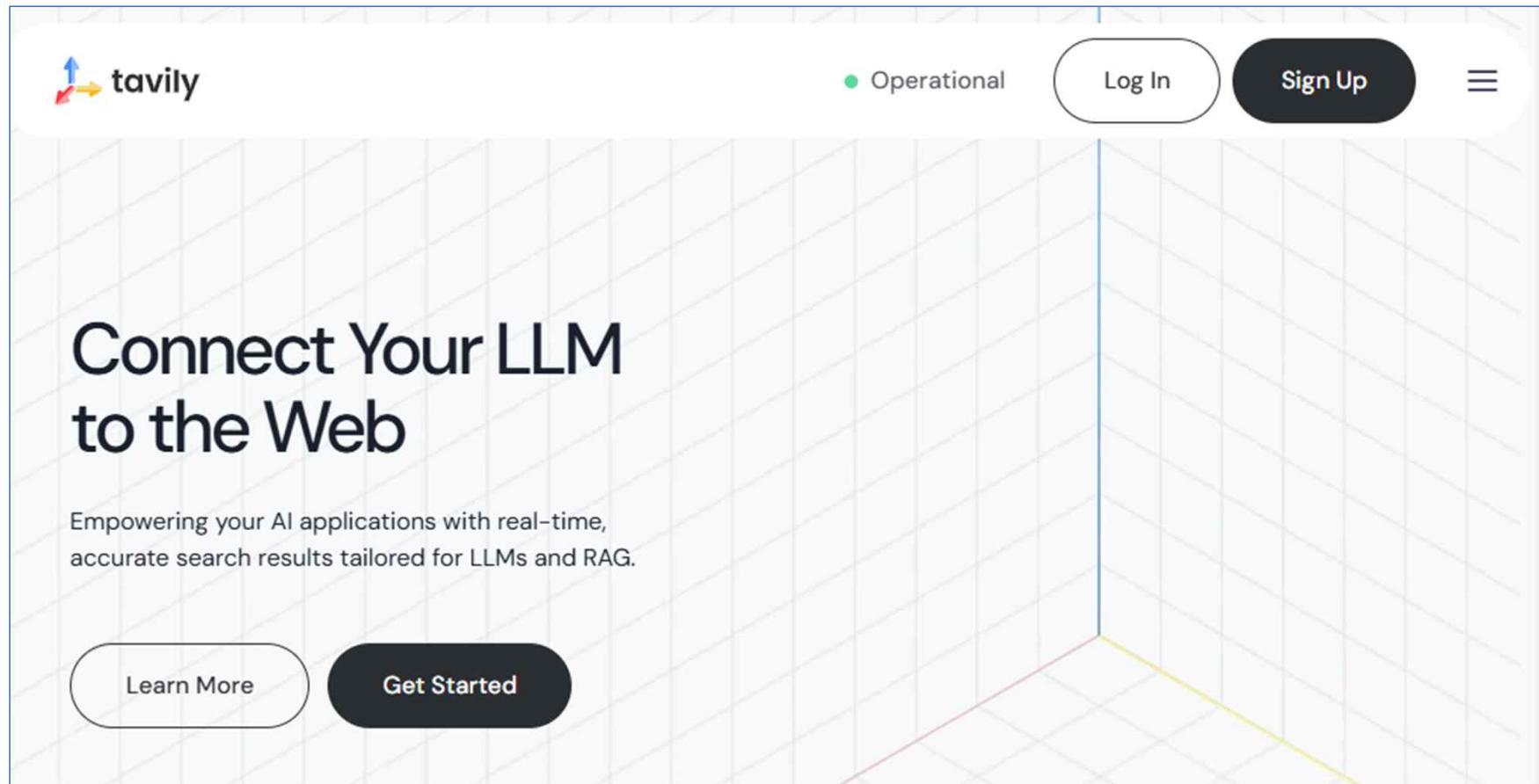
- LLM은 함수 호출(Function Calling) 기능을 통해 외부 도구 및 API와 안정적으로 통합될 수 있습니다.
- JSON 기반 함수 호출:
 - 모델은 함수 시그니처를 따르는 JSON 객체로 응답합니다.
 - 개발자는 함수의 입력 파라미터를 JSON 스키마로 설명합니다.
- 자동 함수 호출 결정:
 - 모델은 입력에 따라 적절한 함수를 호출할 시점을 감지합니다.
- 사용 사례:
 - 외부 도구 호출: API, 플러그인 등과 연결해 작업 수행.
 - 예: get_current_weather(location: string, unit: 'celsius' | 'fahrenheit')



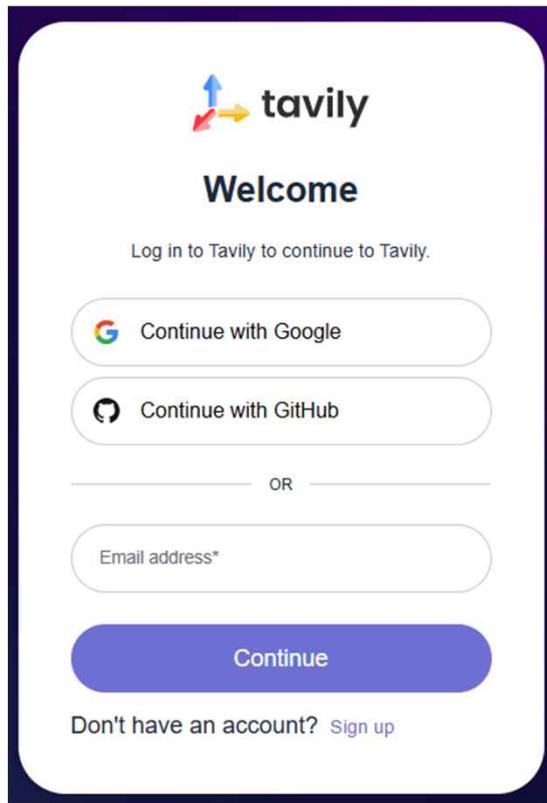
• 함수 호출 단계

- (1) 함수 정보와 함께 LLM 모델 호출
- (2) 모델이 함수 호출을 결정 – 함수명과 argument 반환
- (3) 함수 호출 – 모델이 만들어준 argument 이용
- (4) 함수 호출 결과를 모델에 제공
- (5) 모델이 최종 응답 생성

Tavily – AI Agent 용 검색 엔진



tavily API Key



API Keys			
The key is used to authenticate your requests to the Research API . To learn more, see the documentation page.			
NAME	USAGE	KEY	OPTIONS
default	0	tvly-*****	

TAVILY_API_KEY=tvly-*****Oh

실습: 012_Tools - LLM의 함수 호출

- LLM이 직접 계산하거나 외부 정보에 접근하지 않고, 정의된 함수/도구를 호출해 결과를 얻도록 유도
- @tool 데코레이터로 Python 함수를 도구 등록
- TavilySearch 웹 검색 툴 생성 → tools 리스트에 등록
- create_agent의 tools 파라미터로 모델에 연결
- 어떤 도구를 언제 호출할지는 LLM이 결정
- 함수 실행 결과는 ToolMessage로 agent가 모델에 재 전달
- 모델이 도구 호출 결과를 반영해 최종 답변 작성

단기 메모리 (Short-term Memory)

- AI가 이전 상호작용의 정보를 기억하는 시스템
- 에이전트가 여러 번의 사용자 상호작용을 포함한 복잡한 작업을 수행 가능
- 에이전트는 AgentState 를 사용하여 단기 메모리(short-term memory) 를 관리합니다. 특히, 대화 이력(conversation history)은 messages 키를 통해 저장

실습: 013_Memory – 단기 메모리

- LangChain 에이전트 단기 메모리 구현
- InMemorySaver 또는 SqliteSaver로 thread 단위 대화 이력 지속성 구현
- 메시지 trimming/삭제/요약 기능
- 메모리 접근/수정
 - 커스텀 AgentState로 메모리 확장
 - @before_model 미들웨어로 민감정보 차단
 - @after_model 미들웨어로 부적절한 응답 제거

스트리밍 (Streaming)

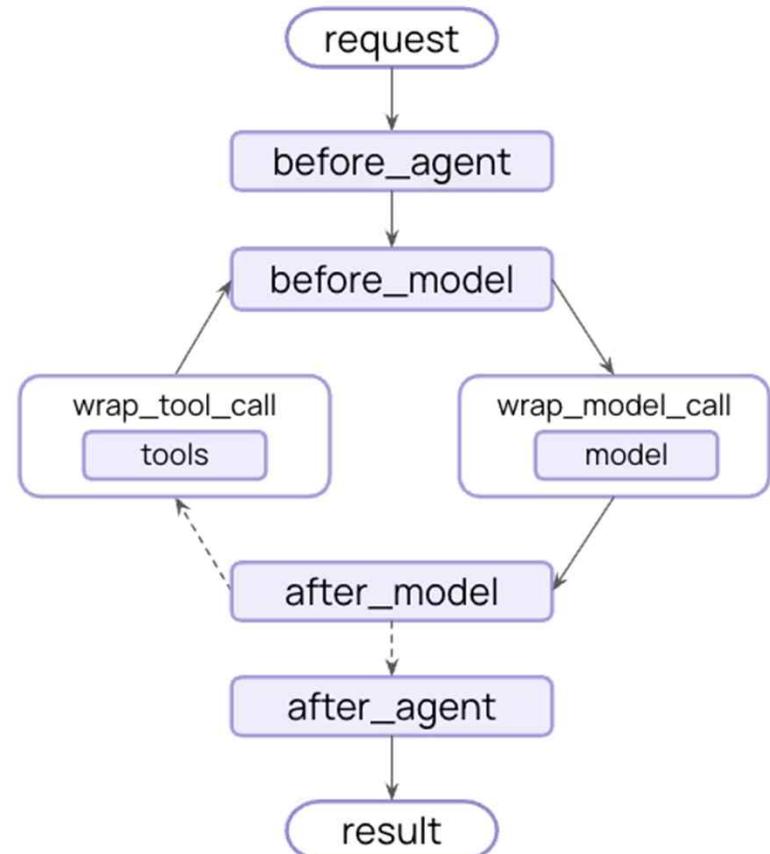
- 스트리밍은 LLM 기반 애플리케이션의 응답성을 향상시키는 데 매우 중요
- 완전한 응답이 준비되기 전에도 출력을 점진적으로 표시함으로써 스트리밍은 사용자 경험(UX)을 크게 향상시키며, 특히 LLM의 자연 시간을 처리 할 때 더욱 그렇다.
- 스트리밍 시스템을 사용하면 에이전트 실행에 대한 실시간 피드백을 애플리케이션에 표시 가능

실습: 014_Streaming

- LLM의 전체 응답을 기다리지 않고, 진행 중인 결과를 실시간으로 표시
- 기능 종류
- Agent 단계별 진행 상황 스트리밍
- LLM 토큰 생성 실시간 스트리밍
- 모드 종류
 - "updates": 단계별 상태 업데이트
 - "messages": 토큰 단위 실시간 생성
 - "values": LLM의 최종 응답 객체들을 순서대로 반환
- 효과 – “점진적 응답” 제공으로 대화형 AI 반응성 향상
활용 예시 – 실시간 챗봇, 진행률 표시, 대화 로그 스트림

미들웨어 (Middleware)

- 에이전트 내부에서 일어나는 일을 더 세밀하게 제어
- 핵심 실행 루프의 각 단계(모델 호출, 도구 실행, 종료 처리)의 이전(before)과 이후(after) 시점에 개입할 수 있는 흙(hook)을 제공
- hook(흙) - 프로그래밍에서 특정 동작이 일어날 때 자동으로 실행되도록 연결된 코드 조각



실습: 015_Middleware

- 에이전트의 실행 과정(모델 호출 → 도구 실행 → 종료)을 세밀하게 제어
- 주요 내장 미들웨어
 - SummarizationMiddleware: 토큰 한도 도달 시 대화 자동 요약
 - HumanInTheLoopMiddleware: 도구 실행 전 사용자 승인/수정/거부 대기
- 인간 개입 (Human-in-the-loop)
- 호출 제한/재시도
 - ModelCallLimitMiddleware: 모델 호출 횟수 제한 (무한루프/비용 방지)
 - ToolCallLimitMiddleware: 특정 도구 또는 모든 도구에 대해 호출 횟수 제한
 - ToolRetryMiddleware: 도구 실패 시 자동 재시도 (백오프 전략)
- PII 감지 (PII Protection)
 - PIIMiddleware: 개인정보(이메일, 카드번호, API키) 자동 탐지/마스킹

구조화된 출력 (Structured Output)

- 모델이 특정한 출력 구조를 따르도록 지시 받는 것
- 모델의 응답이 데이터베이스 스키마나 다른 정해진 형식에 부합하도록 보장
→ 모델 출력의 일관성 유지
- 구조화된 출력 구현 방법 → 출력 구조를 스키마로 정의
- 스키마는 JSON과 같은 단순한 형식으로 표현
- Pydantic과 같은 도구를 사용하여 타입 힌트와 유효성 검사를 포함한 스키마 정의
- LangChain은 `with_structured_output()` 메서드를 제공하여 스키마를 모델에 바인딩하고, 모델이 구조화된 출력을 생성하도록 지원

Pydantic 활용

- 데이터 검증 및 설정 관리를 도와주는 라이브러리
- JSON 데이터를 Python 객체로 변환하는 데 유용
- Pydantic.BaseModel의 역할
 - 데이터 유효성 검사 - 입력값이 올바른 형식인지 확인
 - 자동 JSON 변환 - 객체를 JSON 형식으로 직렬화(Serialization)
 - 간결한 코드 - 복잡한 데이터 처리를 쉽게 구현 가능
- 동작 방식
 - LLM의 구조화된 출력(Structured Outputs) 기능 사용
 - JSON 형식의 응답을 생성하도록 유도
 - Python의 pydantic은 JSON 데이터를 Python 객체로 변환하고 검증하는 역할

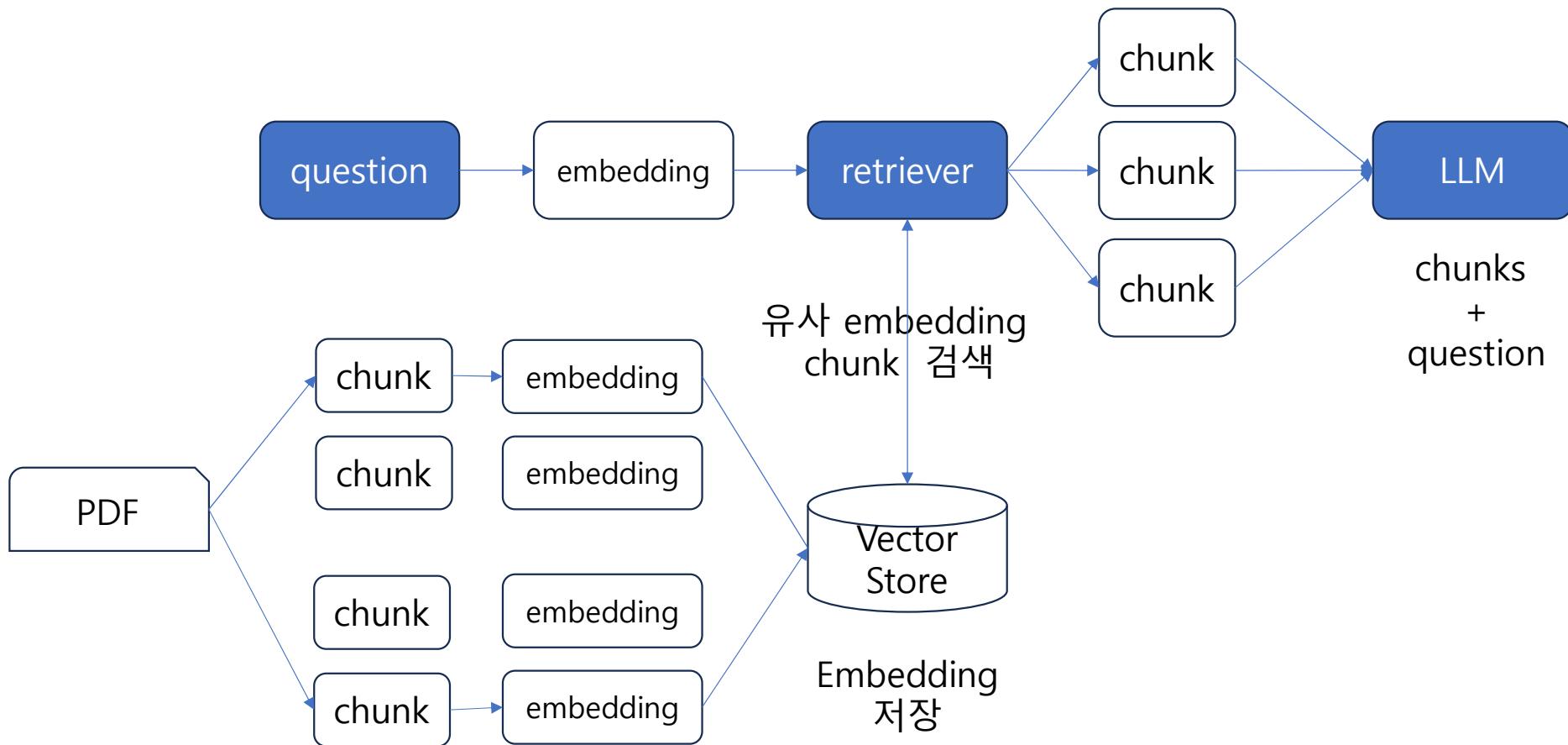
실습: 016._Structured Output

- Schema 정의와 도구 호출을 활용한 분류
- Schema의 세밀한 정의 – pydantic 사용
 - Literal, Enum, Optional, Field 등을 활용하여 LLM의 출력을 더 엄격히 제어
- 구조화된 출력 (Structured Outputs) - model.with_structured_output 사용
 - 모델이 직접 JSON 응답을 생성하며, 파싱 오류 없이 Pydantic으로 자동 검증됨
- 스키마를 도구로서 모델에 바인딩 – model.bind_tools 사용
 - Pydantic 모델을 도구(Tool)나 출력 포맷으로 모델에 바인딩하여 자동 검증/파싱 수행

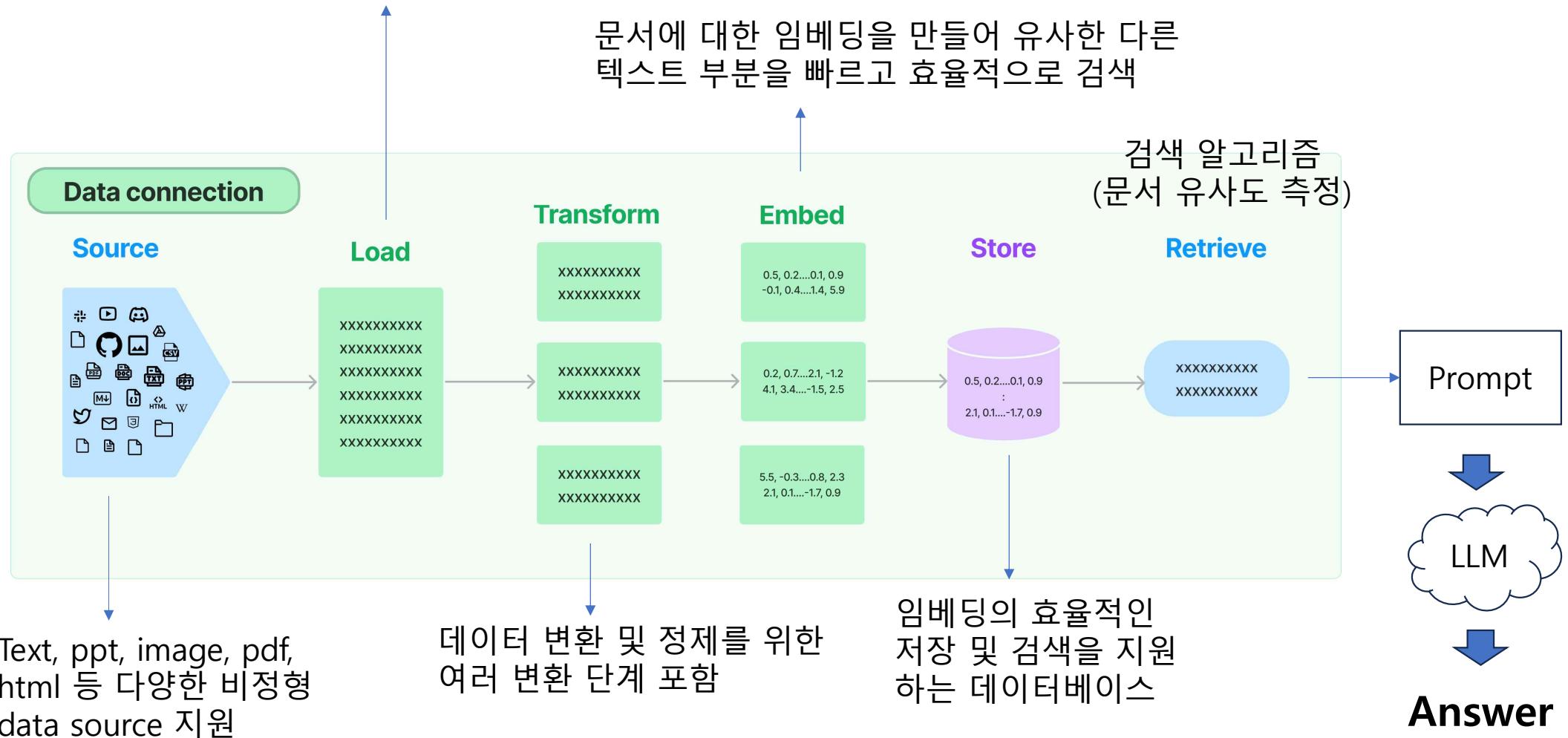
검색 증강 생성(Retrieval-Augmented Generation, RAG)

- 대규모 언어 모델(LLM)의 한계를 보완하기 위해 외부 지식 베이스를 활용하여 텍스트 생성의 정확성과 최신성을 향상시키는 기술
- 구성 요소
 - 검색기(Retriever): 사용자 입력에 따라 관련 정보를 외부 데이터베이스에서 검색
 - 생성기(Generator): 검색된 정보를 기반으로 사용자에게 응답을 생성
- 작동 순서
 1. 질의 처리: 사용자의 질문을 분석
 2. 정보 검색:
*검색기가 외부 지식 베이스에서 관련 정보를 검색
 3. 응답 생성: 생성기가 검색된 정보를 활용하여 정확하고 풍부한 응답을 생성

RAG Flow



다양한 소스(Web Site, DB, YouTube 등)에서 문서 (HTML, PDF, JSON, Word, ppt, 코드 등) 로드



LangChain을 이용한 검색 (retrieval)

- 검색 엔진 구축에 필요한 LangChain components
 - 문서 및 문서 로더 (Document and Document Loader)
 - 텍스트 분할기 (Text splitters)
 - 임베딩 (Embeddings)
 - 벡터 스토어 및 검색기 (Vector stores and retrievers)

Document Loader

- RAG 시스템에서 다양한 소스의 문서를 불러와 처리 가능한 형식으로 변환하는 기능을 수행. 이를 통해 외부 데이터 소스에서 정보를 효율적으로 수집하고, LLM(대규모 언어 모델)이 활용할 수 있도록 지원
- 주요 기능
 - 다양한 소스 지원 (웹 페이지, PDF, CSV, 텍스트 파일 등)
 - 불러온 문서를 분석하고 처리하여, 다른 모듈이 활용하기 쉬운 형태로 변환
 - 대량의 문서를 효율적으로 관리하고, 필요 시 빠르게 접근할 수 있도록 지원
- 종류 - 텍스트 파일 로더, CSV 파일 로더, PDF 파일 로더, 웹 페이지 로더, 디렉토리 로더

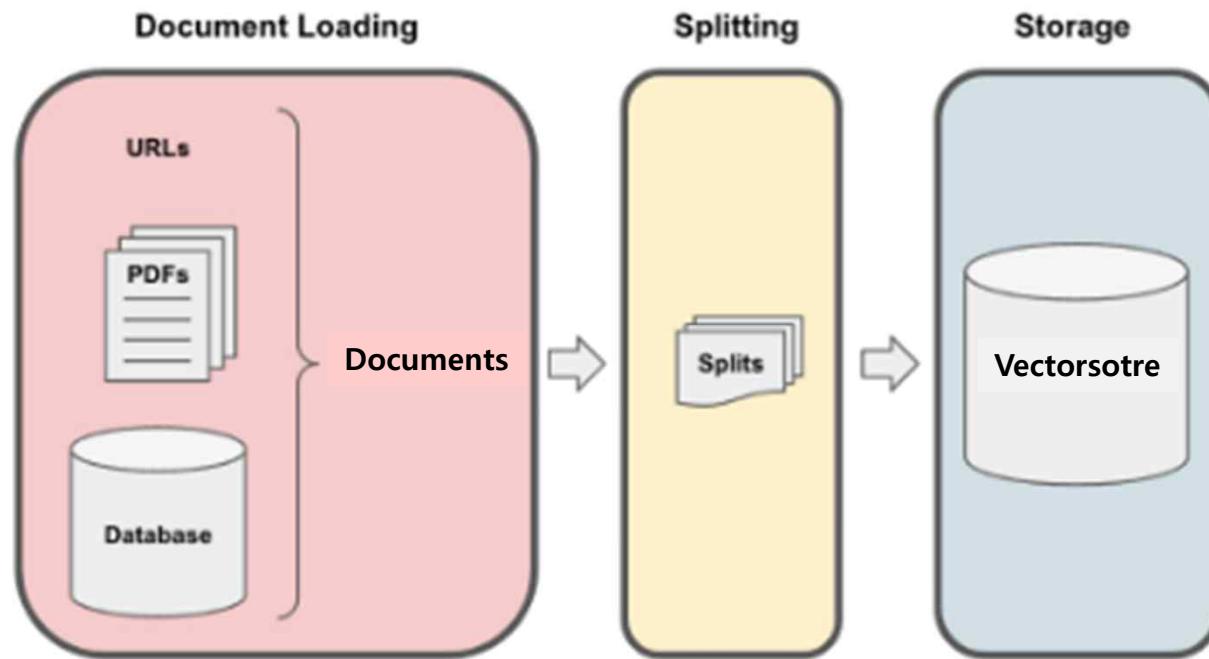
문서 분할 (Document Splitting)

- 문서 분할은 많은 애플리케이션에서 중요한 전처리 단계
- 이 과정은 큰 텍스트를 더 작고 관리하기 쉬운 청크로 나누는 것을 포함
- 다양한 문서 길이를 일관되게 처리하고, 모델의 입력 크기 제한을 극복하며, 검색 시스템에서 사용되는 텍스트 표현의 품질을 향상시키는 등 여러 가지 이점을 제공



텍스트 분할기 (Text Splitters)

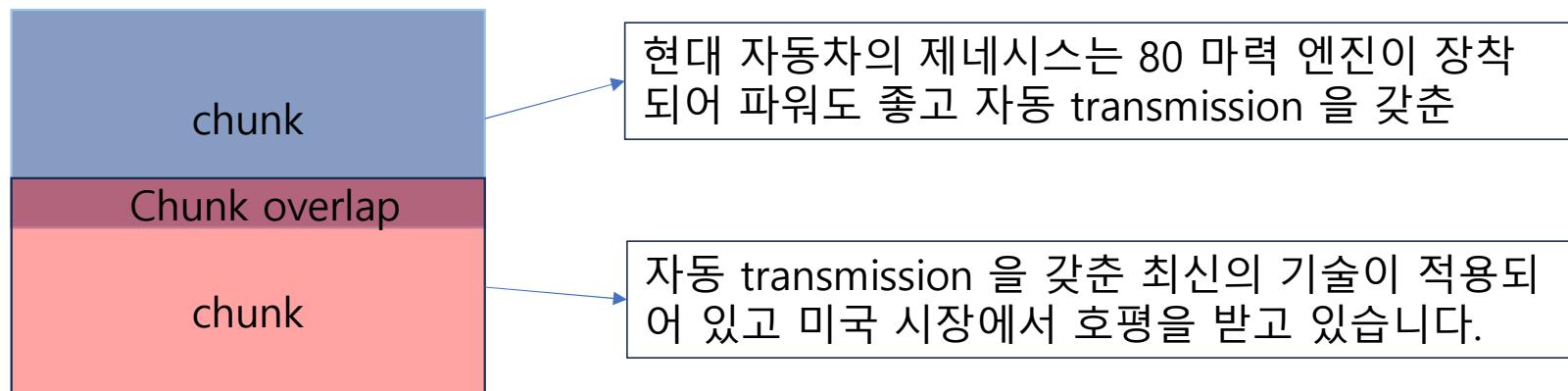
- 긴 텍스트 조각을 처리하려면 해당 텍스트를 여러 덩어리(chunk)로 분할



Chunk Overlap

- 텍스트 분할기는 일부 청크가 겹치는 청크 분할 포함

Ex) 현대 자동차의 제네시스는 80 마력 엔진이 장착되어 파워도 좋고 자동 transmission 을 갖춘 최신의 기술이 적용되어 있고 미국 시장에서 호평을 받고 있습니다.



질문 : 제네시스의 자동차 사양에 대해 설명해줘

텍스트 분할 방식

1. 길이 기준 (Length-based) 분할

- 구현이 간단하며 일관된 청크 크기 유지
- 다양한 모델 요구 사항에 쉽게 적응

2. 텍스트 구조 기반(Text-structured based) 분할

- 단락, 문장, 단어와 같은 계층적 단위로 구성된 고유한 구조를 활용하여 분할
- 자연스러운 언어 흐름을 유지하고, 분할 내 의미적 일관성을 보장

3. 문서 구조 기반 (Document-structured based) 분할

- HTML, Markdown, JSON 등의 고유한 구조에 따라 분할
- 문서의 원래의 논리적 구조가 보존되고 검색, 요약 등 후속 작업에 더 효과적

Splitter 종류 (langchain_text_splitters)

- CharacterTextSplitter() – character 단위로 분할(기본적으로 "\n\n")
- RecursiveCharacterTextSplitter() – character 단위로 분할(["\n\n", "\n", " ", ""])
- MarkdownHeaderTextSplitter() – 특정 header 기준으로 markdown 파일 분할
- SentenceTransformersTokenTextSplitter() – token 기준으로 분할

What is embedding ?

- 정의 - 고차원의 데이터를 저차원으로 매핑하면서, 그 의미를 보존하는 것
- 글, 이미지나 동영상같은 비정형 데이터를 숫자로 표현
- 큰 데이터를 저차원으로 압축하여 효율적으로 저장하며 의미 보존
- 인공지능 모델이 아주 잘 다룰 수 있는 표현 형식
- 데이터 간의 유사도 측정 방법
 - 코사인 유사도: -1~1 사이의 범위
 - Dot product
 - 유클리드 거리: 두 벡터 사이의 직선 거리 (L_2)

Word Embedding (Feature 표시)

dimension	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
성별	-1	1	-0.95	0.97	0.00	0.01
귀족	0.01	0.02	0.93	0.95	-0.01	0.00
나이	0.03	0.02	0.7	0.69	0.03	-0.02
음식	0.04	0.01	0.02	0.01	0.95	0.97



King (4914) 의 4 dimension vector 표시

Man (5391) 의 4 dimension vector 표시

Embedding matrix (example)

학습된 300 차원 features

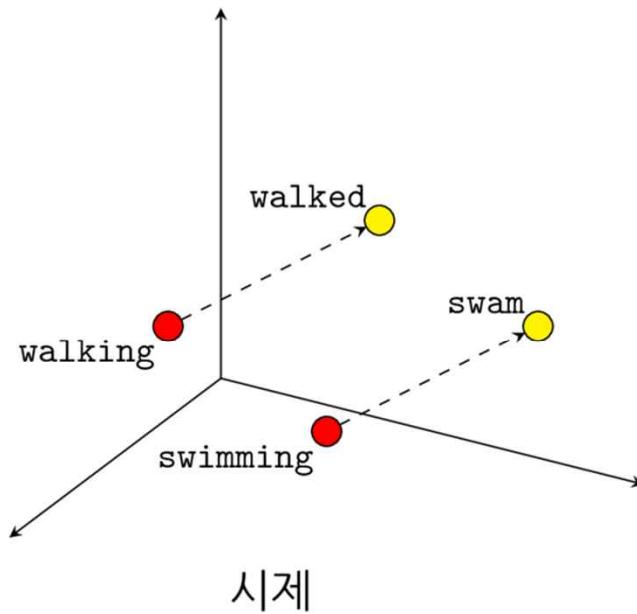
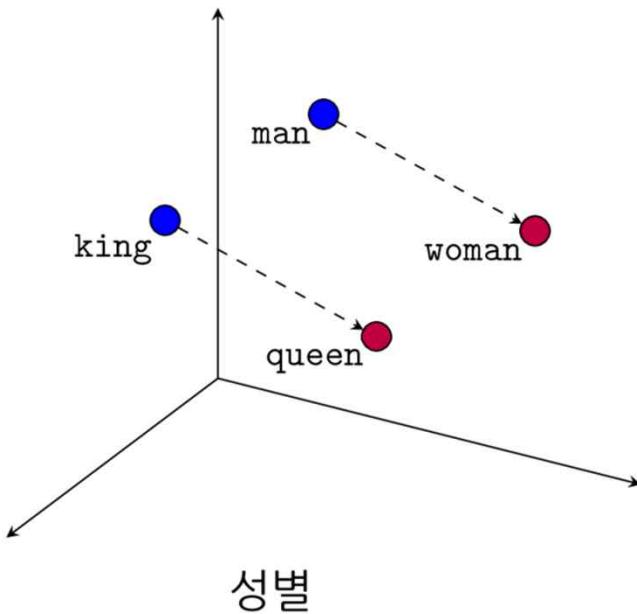
Words

	0	1	2	3	4	5	6	7	8	9	...	290	291	292
fox	-0.348680	-0.077720	0.177750	-0.094953	-0.452890	0.237790	0.209440	0.037886	0.035064	0.899010	...	-0.283050	0.270240	-0.654800
ham	-0.773320	-0.282540	0.580760	0.841480	0.258540	0.585210	-0.021890	-0.463680	0.139070	0.658720	...	0.464470	0.481400	-0.829200
brown	-0.374120	-0.076264	0.109260	0.186620	0.029943	0.182700	-0.631980	0.133060	-0.128980	0.603430	...	-0.015404	0.392890	-0.034826
beautiful	0.171200	0.534390	-0.348540	-0.097234	0.101800	-0.170860	0.295650	-0.041816	-0.516550	2.117200	...	-0.285540	0.104670	0.126310
jumps	-0.334840	0.215990	-0.350440	-0.260020	0.411070	0.154010	-0.386110	0.206380	0.386700	1.460500	...	-0.107030	-0.279480	-0.186200
eggs	-0.417810	-0.035192	-0.126150	-0.215930	-0.669740	0.513250	-0.797090	-0.068611	0.634660	1.256300	...	-0.232860	-0.139740	-0.681080
beans	-0.423290	-0.264500	0.200870	0.082187	0.066944	1.027600	-0.989140	-0.259950	0.145960	0.766450	...	0.048760	0.351680	-0.786260
sky	0.312550	-0.303080	0.019587	-0.354940	0.100180	-0.141530	-0.514270	0.886110	-0.530540	1.556600	...	-0.667050	0.279110	0.500970
bacon	-0.430730	-0.016025	0.484620	0.101390	-0.299200	0.761820	-0.353130	-0.325290	0.156730	0.873210	...	0.304240	0.413440	-0.540730
breakfast	0.073378	0.227670	0.208420	-0.456790	-0.078219	0.601960	-0.024494	-0.467980	0.054627	2.283700	...	0.647710	0.373820	0.019931
toast	0.130740	-0.193730	0.253270	0.090102	-0.272580	-0.030571	0.096945	-0.115060	0.484000	0.848380	...	0.142080	0.481910	0.045167
today	-0.156570	0.594890	-0.031445	-0.077586	0.278630	-0.509210	-0.066350	-0.081890	-0.047986	2.803600	...	-0.326580	-0.413380	0.367910
blue	0.129450	0.036518	0.032298	-0.060034	0.399840	-0.103020	-0.507880	0.076630	-0.422920	0.815730	...	-0.501280	0.169010	0.548250
green	-0.072368	0.233200	0.137260	-0.156630	0.248440	0.349870	-0.241700	-0.091426	-0.530150	1.341300	...	-0.405170	0.243570	0.437300
kings	0.259230	-0.854690	0.360010	-0.642000	0.568530	-0.321420	0.173250	0.133030	-0.089720	1.528600	...	-0.470090	0.063743	-0.545210
dog	-0.057120	0.052685	0.003026	-0.048517	0.007043	0.041856	-0.024704	-0.039783	0.009614	0.308416	...	0.003257	-0.036864	-0.043878
sausages	-0.174290	-0.064869	-0.046976	0.287420	-0.128150	0.647630	0.056315	-0.240440	-0.025094	0.502220	...	0.302240	0.195470	-0.653980
lazy	-0.353320	-0.299710	-0.176230	-0.321940	-0.385640	0.586110	0.411160	-0.418680	0.073093	1.486500	...	0.402310	-0.038554	-0.288670
love	0.139490	0.534530	-0.252470	-0.125650	0.048748	0.152440	0.199060	-0.065970	0.128830	2.055900	...	-0.124380	0.178440	-0.099469
quick	-0.445630	0.191510	-0.249210	0.465900	0.161950	0.212780	-0.046480	0.021170	0.417660	1.686900	...	-0.329460	0.421860	-0.039543

20 rows × 300 columns

Word Embedding 의 결과

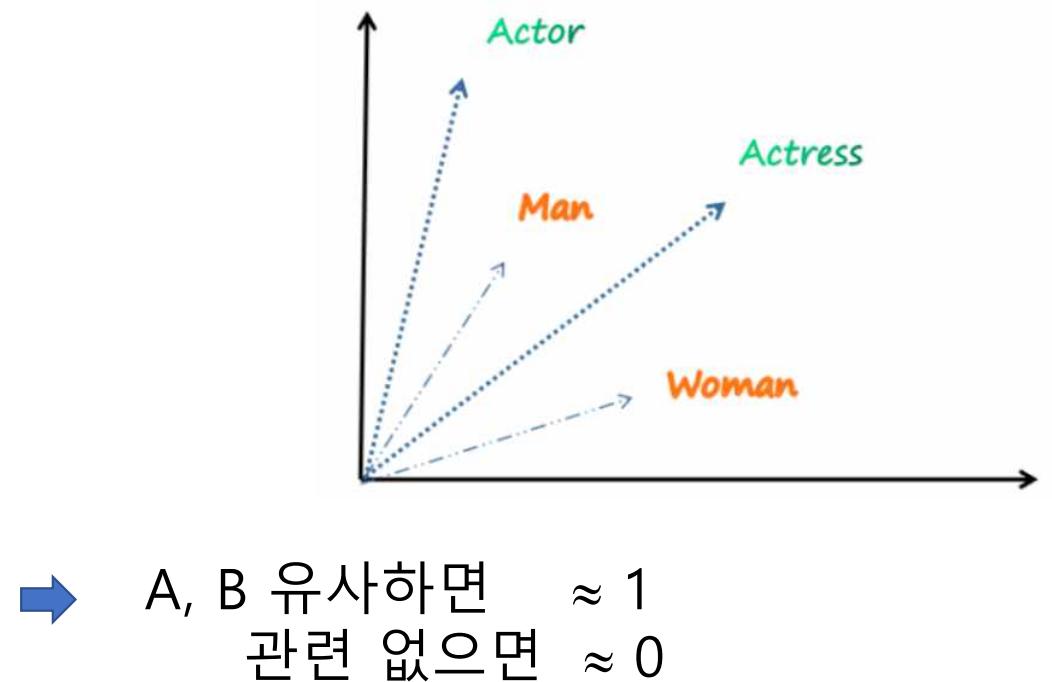
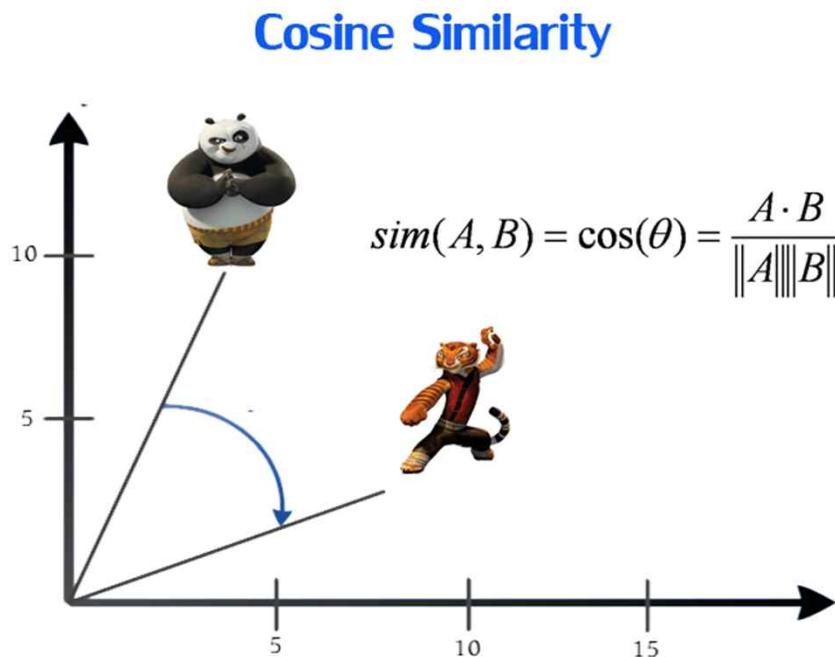
king – queen ≈ man - woman



Spain	Madrid
Italy	Rome
Germany	Berlin
Turkey	Ankara
Russia	Moscow
Canada	Ottawa
Japan	Tokyo
Vietnam	Honoi
China	Beiging

국가-수도

유사도 측정 (Cosine Similarity)

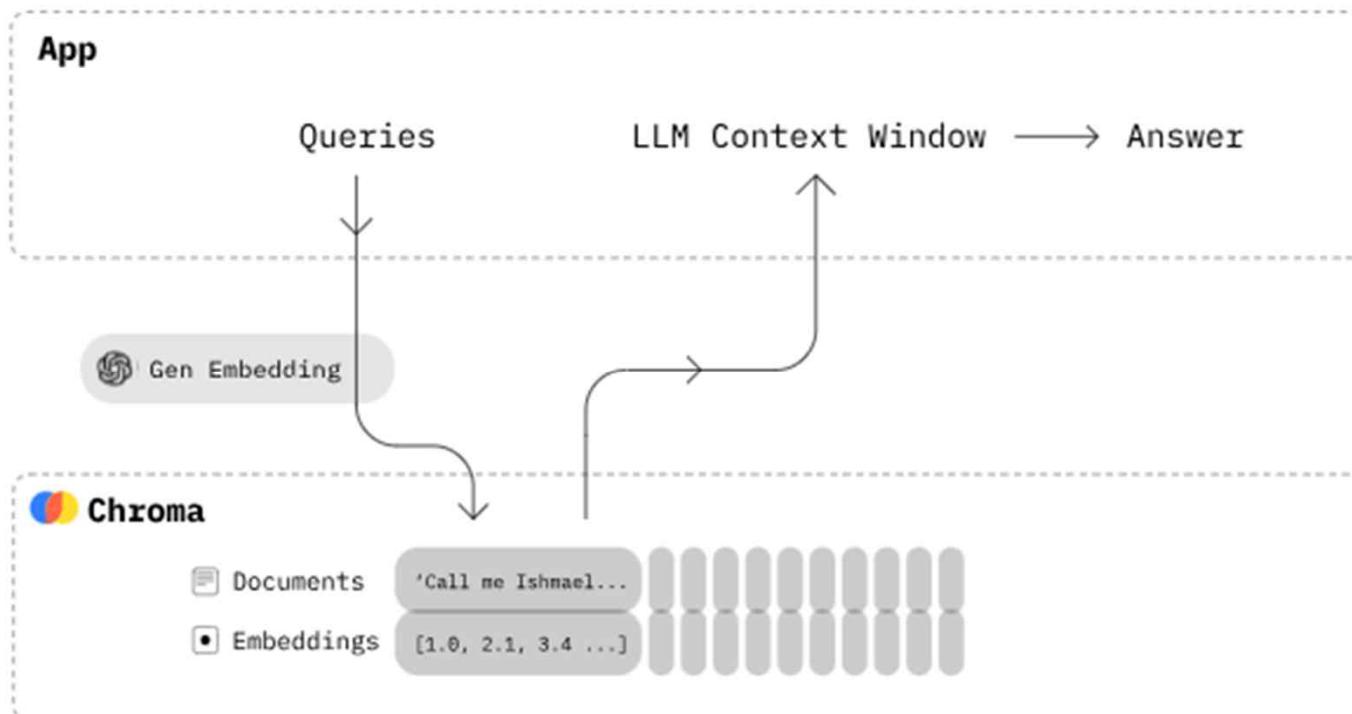


벡터 스토어 (Vector Stores)

- VectorStore 객체는 텍스트 및 문서 객체를 스토어에 추가하고, 다양한 유사도 측정 지표(similarity metrics)를 사용하여 쿼리할 수 있는 메서드 포함
- 벡터 스토어는 대개 임베딩 모델(embedding models)과 함께 초기화
- 벡터 스토어 종류
 - 호스팅형 – Cloud 업체에서 제공 (예, Pinecone)
 - 인프라 독립형 – local 또는 제3자 서비스 (예, FAISS, ChromaDB)
 - 인메모리(in-memory)형 – 가벼운 업무. 메모리 내에서 실행 (Chroma, Redis)

Chroma DB 란?

- 벡터 임베딩을 저장하고 검색하는 데 사용되는 오픈 소스 벡터 스토어
- 임베딩과 메타데이터를 저장하여 LLM에 의해 나중에 사용될 수 있도록 함



Chroma DB 설치 및 사용법

- pip install langchain-chroma
- (PC에 C++가 설치되어 있지 않은 경우)
 - Visual Studio Build Tools 다운로드 → Visual Studio Installer 실행 → Individual Components → MSVC v142 - VS 2019 C++ x64/x86 build tools

```
vector_store = Chroma(  
    collection_name="example_collection",  
    embedding_function=embeddings,  
    persist_directory=".//chroma_langchain_db",  
)
```

벡터 스토어 내에서 사용할 컬렉션 이름

텍스트를 벡터로 변환하는 임베딩 함수

Local disk 저장 경로, 생략시 in-memory 모드 사용

LangChain 의 VectorStore 비교

특징	Chroma	InMemoryVectorStore
데이터 저장 방식	메모리 내 저장 및 영구적 저장 모두 지원. persist_directory를 지정하지 않으면 메모리 내에서 데이터가 저장	메모리 내 저장만 지원하며, 애플리케이션이 종료되면 데이터가 손실됩니다.
종속성	chromadb 및 langchain-chroma 패키지 설치가 필요합니다.	추가 패키지 설치가 불필요 (langchain_core에 포함)
확장성	대규모 데이터셋을 다루거나 데이터의 영구적 저장이 필요한 경우에 적합	테스트 목적이나 소규모 데이터셋을 다루는 경우에 적합
데이터 공유	여러 인스턴스 간에 데이터 공유가 가능	각 인스턴스 별로 데이터가 격리

검색기 (Retrievers)

- LangChain Retriever는 LangChain에서 제공하는 데이터 검색 인터페이스
- LLM이 필요한 정보를 효율적으로 찾을 수 있도록 돕는 역할
- Runnable을 상속받아 동기 및 비동기 실행, 배치 처리를 지원
- VectorStore와 Retriever의 주요 차이점

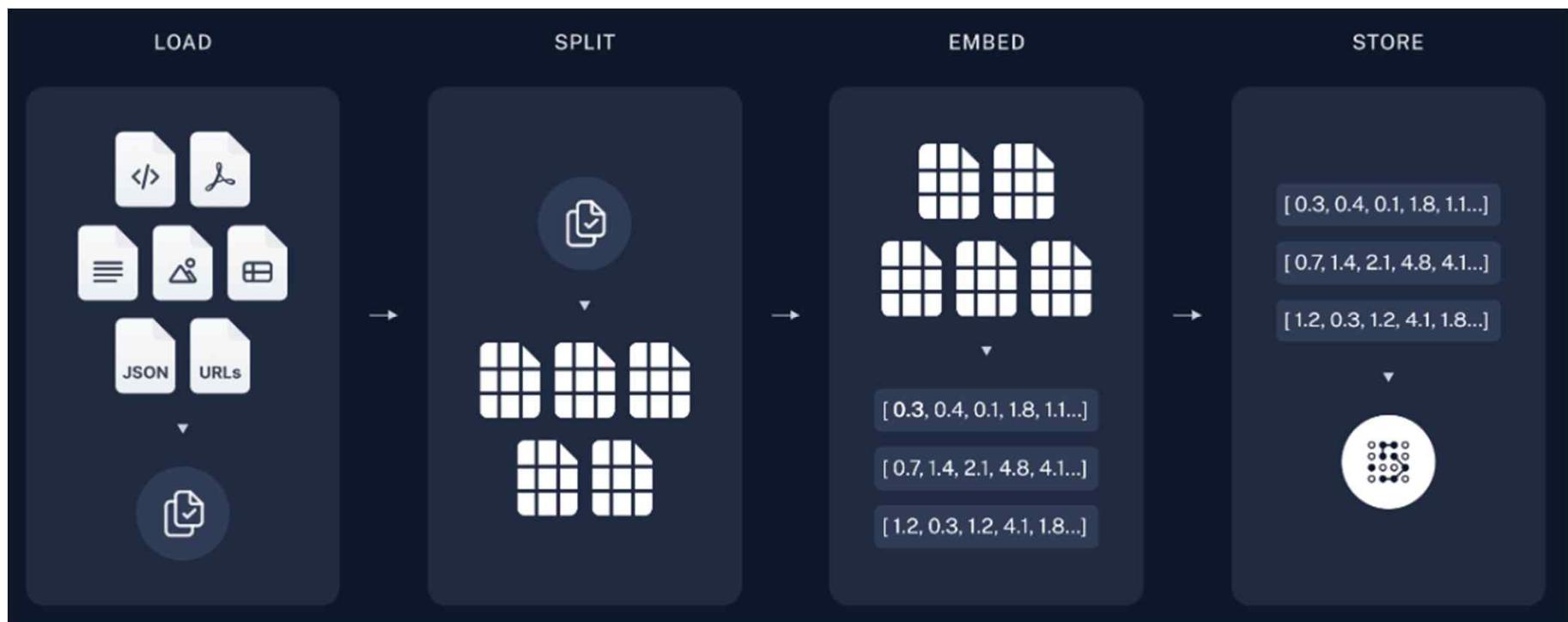
특성	VectorStore	Retriever
주요 역할	문서 저장 및 벡터 임베딩 검색	문서 검색 및 반환
데이터 소스	주로 벡터 데이터 저장소	벡터 스토어 외에도 외부 API와 같은 비-벡터 스토어 데이터 소스와 상호작용
검색 메서드	<code>similarity_search</code> <code>similarity_search_with_score</code> <code>similarity_search_by_vector</code>	<code>as_retriever()</code> <code>get_relevant_documents()</code> <code>search_type="similarity", "mmr"</code>

실습: 050. 시맨틱 검색 엔진 구축

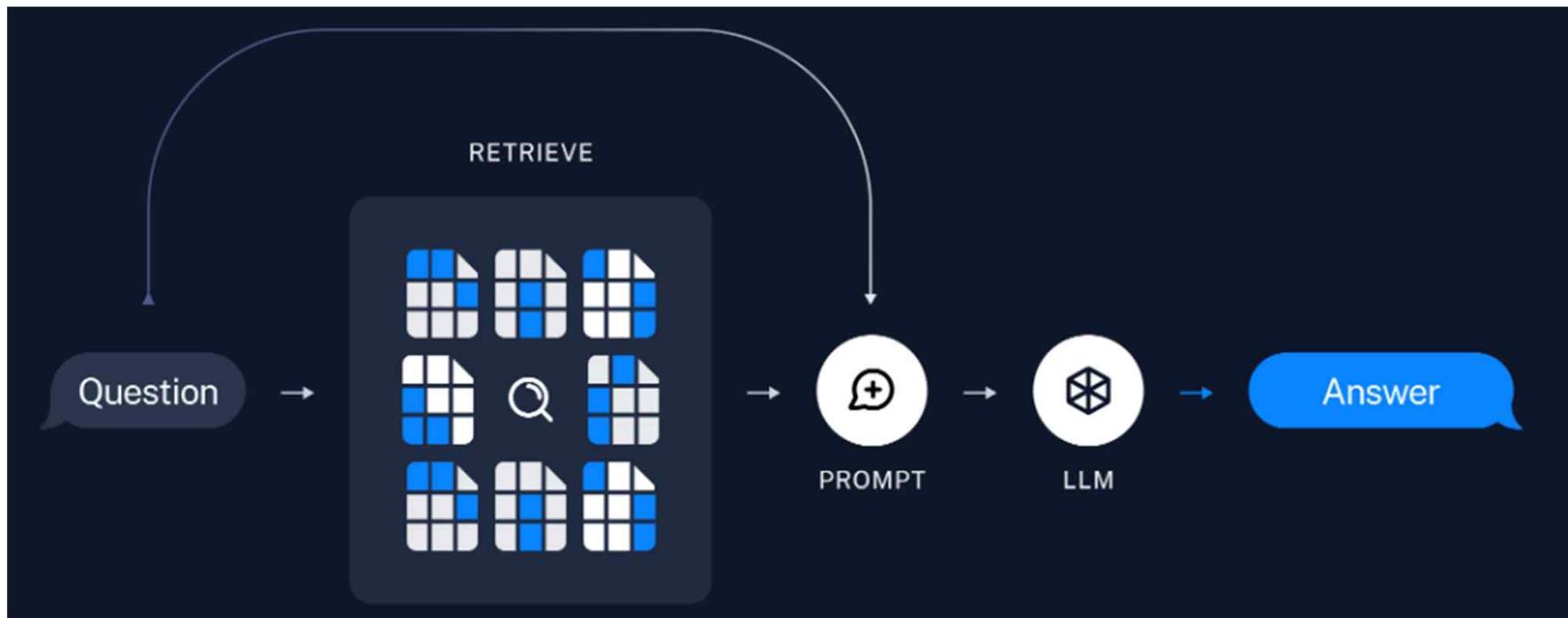
- 문서 및 문서 로더 (Documents and Document Loaders) 이해
- 텍스트 분할 (Splitting)
- 임베딩 (Embeddings)
- 벡터 스토어(Vector Stores) 생성 및 쿼리
- 검색기 (Retrievers)
- similarity_search를 활용한 검색기 작성

Agent를 이용한 RAG 구현

- 인덱싱 단계



- 검색 및 답변 생성 단계

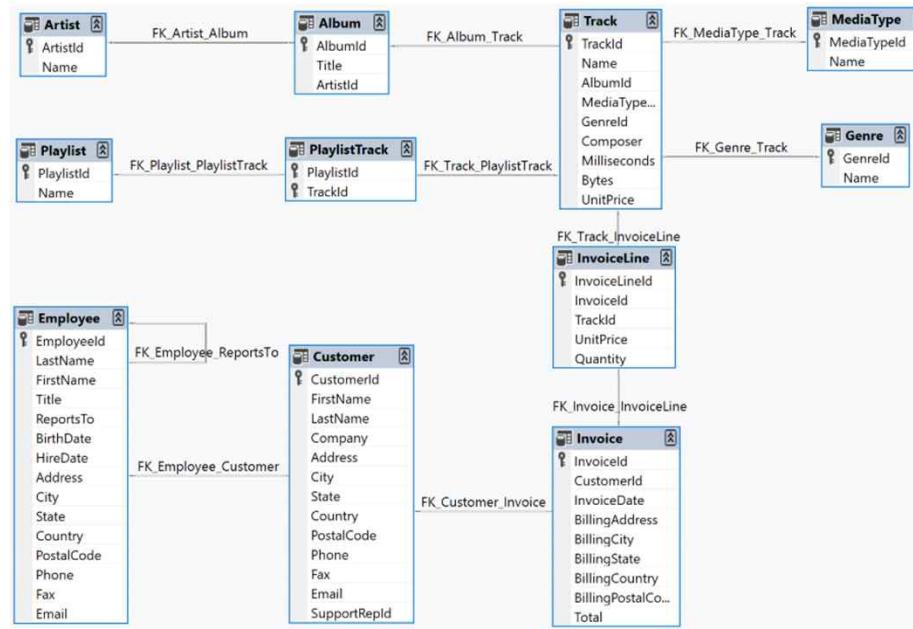


실습: 060. RAG Agent 구축

- RAG 애플리케이션 개념 이해 - 인덱싱, 문서 분할, 문서 저장
- 검색 및 생성(Retrieval and Generation)
- 벡터 스토어 업데이트
- Agent를 이용한 도구 호출
 - LLM 모델은 문맥에 따라 직접 질문에 답변할 수도 있고,
 - 더 많은 정보를 얻기 위해 도구 호출(tool call) 을 생성할 수도 있습니다.
 - Agent는 도구 호출 결과를 문맥에 추가하여 다시 LLM을 호출합니다.

Chinook Database

- 관계형 DB를 학습하고 실습할 수 있도록 제공되는 샘플 DB
- 음악 스토어(Music Store)를 모델링한 데이터베이스



Chinook db 설치

- [Chinook_Sqlite.sql](#) 스크립트를 컴퓨터의 폴더/디렉토리에 저장
- Chinook DB 생성

(langchain) C:\Users\trimu\OneDrive\LangChain_Streamlit_WebService>sqlite3 Chinook.db

- Chinook_Sqlite.sql 스크립트 실행

sqlite> .read Chinook_Sqlite.sql

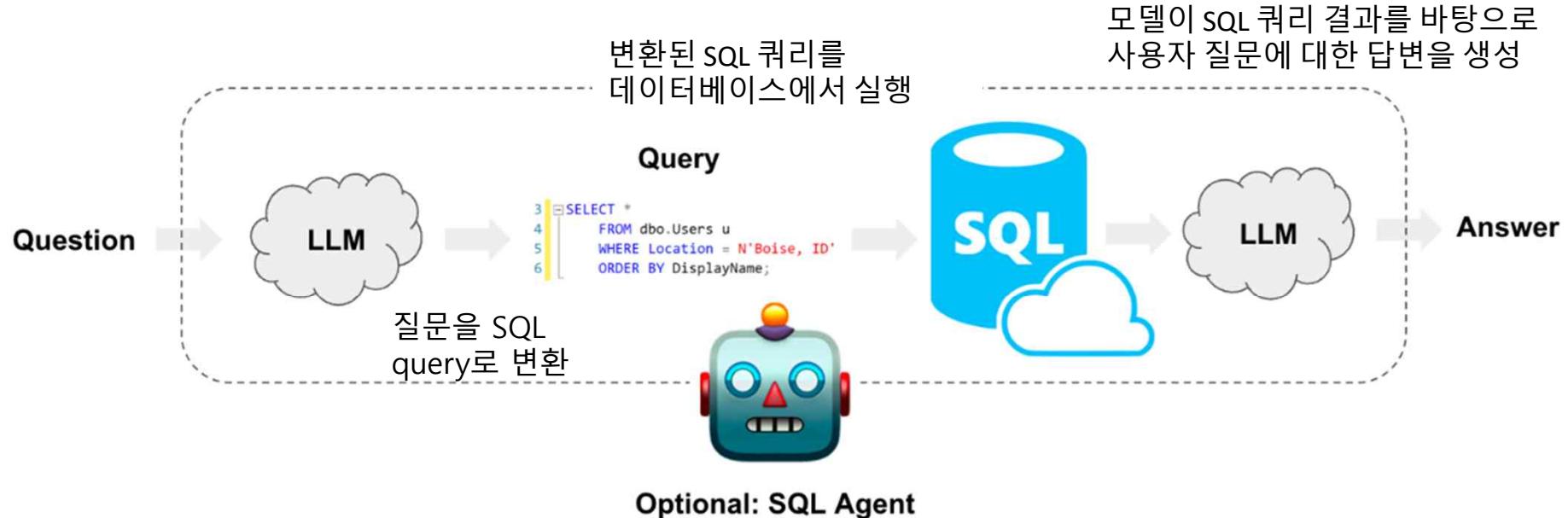
sqlite>.read Chinook_Sqlite.sql

sqlite> SELECT * FROM Artist LIMIT 10;

```
(langchain) C:\Users\trimu\OneDrive\LangChain_Streamlit_WebService>sqlite3 Chinook.db
SQLite version 3.45.3 2024-04-15 13:34:05 (UTF-16 console I/O)
Enter ".help" for usage hints.
sqlite> .read Chinook_Sqlite.sql
sqlite> SELECT * FROM Artist LIMIT 10;
1|AC/DC
2|Accept
3|Aerosmith
4|Alanis Morissette
5|Alice In Chains
6|Antônio Carlos Jobim
7|Apocalyptica
8|Audioslave
9|BackBeat
10|Billy Cobham
```

실습: 070. SQL Agent

- 구조화된 (Structured) Data 처리 - Chinook DB 이용



Supervisor Pattern

- 여러 전문 하위 에이전트(Worker Agents) 를 중앙에서 조율(Supervision) 하는 멀티에이전트 아키텍처
- 각 에이전트가 자신만의 도구와 프롬프트를 가지고 역할 분리
 - Bottom Layer — 실제 API 도구 → create_calendar_event, send_email
 - Middle Layer — 도메인별 전문 에이전트 → calendar_agent, email_agent
 - Top Layer — 전체 조율 담당 Supervisor
- Ex) “다음 주 화요일 오후 2시에 디자인팀 회의 잡고 리마인더 이메일 보내줘.”
→ Supervisor가 schedule_event + manage_email 순차 실행

실습: 080. Supervisor Agent

- LangChain을 활용해 3계층 Supervisor Agent 시스템 구현
- 자연어 요청을 일정 예약 및 이메일 발송으로 자동 변환
- 구현 단계
 - 도구 정의 (Tools) - `create_calendar_event`, `get_available_time_slots`, `send_email`
 - 전문 하위 에이전트 구성
 - `calendar_agent` → 일정 관리
 - `email_agent` → 이메일 작성 및 전송
 - 도구 래핑 (Tool Wrapping)
 - 하위 에이전트를 `schedule_event`, `manage_email`로 감싸 Supervisor 호출 가능
 - Supervisor Agent 생성
 - `create_agent()`로 고수준 요청을 조율

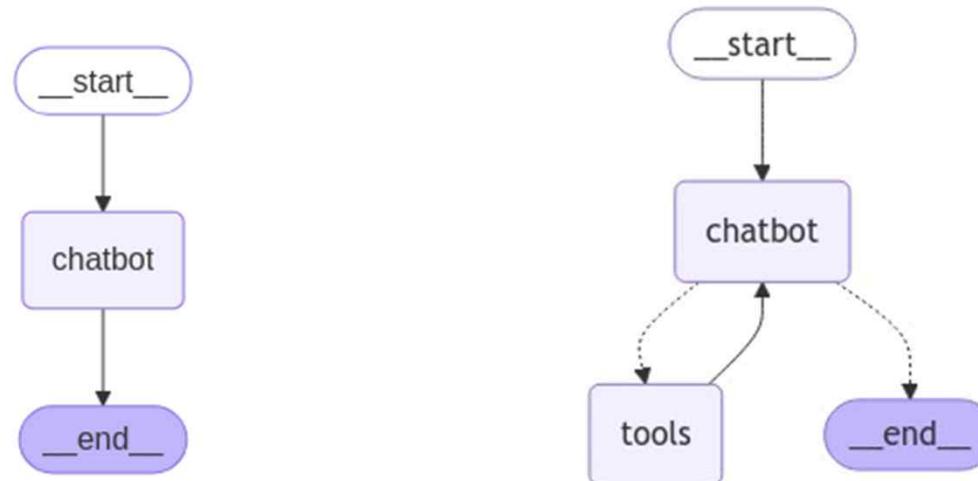
LangGraph

Langgraph 란?

- LLM을 활용해 상태 기반의 다중 에이전트 애플리케이션을 구축
- 그래프 구조를 통해 복잡한 워크플로우를 효율적으로 설계 및 관리
- 상태를 지속적으로 유지해 반복 작업과 복잡한 상태 관리를 지원
- 인간의 개입이 필요한 워크플로우와 메모리 기능을 유연하게 구현
- 주요 특징
 - 상태 지속성
 - 워크플로우 최적화
 - 복잡한 작업 처리
- 활용 사례 - 고객 서비스 챗봇, 복잡한 데이터 분석, 교육용 튜터링 시스템 등

Graph 핵심 요소 – state machine

- State – 애플리케이션의 현재 스냅샷을 나타내는 공유 데이터 구조. TypedDict 나 Pydantic BaseModel로 구성.
- Nodes – Agent 의 logic을 가진 Python 함수. 현재 값을 State 입력으로 받고, 업데이트된 값을 반환
- Edges – 현재 State에 따라 다음에 실행할 Node 를 결정하는 Python 함수. 조건부 분기 또는 고정 전환 가능.



State 구성

- State는 여러 키를 가진 딕셔너리(TypedDict)
- 노드 와 노드 간에 정보를 전달하는 역할
- 각 키는 독립적인 Reducer(병합 규칙)를 가집니다.
- Reducer(add_messages) : 자동으로 list 에 메시지를 추가해 주는 기능
- Reducer를 명시하지 않으면 기본값은 덮어쓰기 override입니다.

```
** 기본 State - no Reducer **
```

```
from typing_extensions import TypedDict

class State(TypedDict):
    foo: int
    bar: list[str]
```

```
** State - Reducer **
```

```
from typing import Annotated
from typing_extensions import TypedDict
from langgraph.graph.message import add_messages

class State(TypedDict):
    foo: int
    bar: Annotated[list[str], add_messages]
```

Node 구성

- 파이썬 함수(동기 또는 비동기)이며, 첫 번째 위치 인수는 state이고(선택적으로) 두 번째 위치 인수는 선택적 구성 가능한 매개변수 (예: thread_id)를 포함하는 "config"

```
def my_node(state: State, config: RunnableConfig):
    print("In node: ", config["configurable"]["user_id"])
    return {"results": f"Hello, {state['input']}!"}
```

- START node – 사용자 입력을 그래프로 전송하는 특수 노드
- END node – 완료를 나타내는 특수 노드

Edge 구성

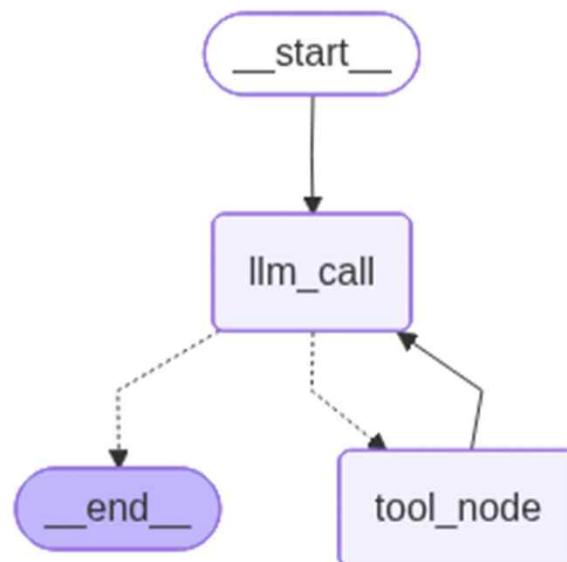
- 노드와 노드 간의 연결
- 일반 에지: 한 노드에서 다음 노드로 직접 이동
- 조건부 에지: 다음에 어느 노드로 이동할지 결정하는 함수를 호출
- 진입점: 사용자 입력이 도착했을 때 가장 먼저 호출할 노드 결정
`graph.add_edge(START, "node_a")`
- 조건부 진입점: 사용자 입력이 도착했을 때 먼저 호출할 노드를 결정하는 함수를 호출
`graph.add_conditional_edges(START, routing_function, {True: "node_b", False: "node_c"})`

LangGraph Checkpointer Memory

- Checkpointer란?
 - LangGraph 실행 상태(state)를 저장·복구하는 메커니즘
 - 각 노드간 실행결과를 추적하기 위한 메모리(대화에 대한 기록과 유사 개념)
- 동작 방식
 - 노드 실행 전/후의 상태(state)를 저장 (messages, 변수 값, edge 위치 등)
 - 실행이 중단되면 마지막 체크포인트에서 재개
 - 체크포인터를 활용하여 특정 시점(Snapshot)으로 되돌리기 기능도 가능
- 활용 시나리오
 - 대화형 에이전트(Chatbot)에서 사용자 맥락 유지
 - 복잡한 워크플로우 실행 중단 후 재개
 - 장기 세션 관리 및 상태 기반 로깅

실습: 501. Graph API - 계산기 에이전트

- LangGraph를 이용해 간단한 계산기 에이전트(calculator agent) 를 구현
- 주요 구성 요소
 - 도구 정의
 - 상태 정의
 - 노드 구성
 - 조건부 흐름 제어
 - 그래프 빌드 및 실행



Functional API 개요

- 일반적인 Python 애플리케이션으로 LangGraph의 핵심 기능인 지속성(persistence), 메모리(memory), Human-in-the-loop, 스트리밍(streaming) 기능을 손쉽게 통합할 수 있도록 설계
- 핵심 구성 요소
 - **@entrypoint**
 - 워크플로우의 시작 지점(entry point)을 표시
 - 내부 로직을 캡슐화하고 실행 흐름을 관리하며, 장기 실행 작업(long-running tasks)과 인터럽트(interrupts) 처리를 담당
 - **@task**
 - 비동기적으로 실행 가능한 작업 단위(API 호출, 데이터 처리 단계 등을 정의)
 - @entrypoint 내부에서 호출되며, Future-like 객체를 반환해 await하거나 동기적으로 결과를 가져올 수 있습니다.
- 컴파일이 필요 없는 Graph API

Functional API vs. Graph API 비교

구분	Functional API	Graph API
접근 방식	<ul style="list-style-type: none">Python 제어 구조(<code>if</code>, <code>for</code>, 함수 호출 등)를 그대로 사용.명령형(Imperative) 스타일.	<ul style="list-style-type: none">노드(Node)와 엣지(Edge)를 사용해 워크플로우를 선언적으로 정의.그래프(Declarative) 스타일.
제어 흐름 정의 (Control Flow)	일반적인 Python 문법으로 로직을 구성할 수 있어 코드가 간결함.	워크플로우를 명시적으로 그래프로 구성해야 함. 복잡한 단계나 병렬 처리를 설계하기에 적합.
단기 메모리	함수 범위 내의 로컬 상태만 유지. 별도의 <code>State</code> 객체나 <code>reducer</code> 정의 불필요.	<code>State</code> 를 명시적으로 정의하고, 상태 업데이트를 위해 <code>reducer</code> 를 작성해야 할 수도 있음.
체크포인트	하나의 entrypoint 실행 전체를 단일 체크포인트로 관리	그래프의 각 superstep(실행 루프)이 끝날 때마다 새 체크포인트 생성
통합 가능성	Graph API와 동일한 런타임(runtime)을 사용하므로, 두 API를 하나의 애플리케이션 내에서 함께 사용 가능.	

실습: 502. Functional API

- 그래프 구성 없이 함수 기반으로 에이전트를 정의
- 코드 흐름을 Python 함수처럼 직관적으로 표현
- 주요 구성 요소
 - 도구 정의
 - LLM 호출 노드 정의
 - 도구 실행 노드 정의
 - 에이전트 집입점 정의

Human-in-the-loop 제어

- AI 에이전트 실행 중간에 사람이 개입할 수 있도록 하는 제어 방식
 - 실행 중간 상태(state)를 일시 중단(pause) → 사용자 입력/승인 대기
- 장점
 - 안전성 확보: 잘못된 도구 호출, 민감 작업(예: 결제/데이터 삭제) 전에 승인 가능
 - 품질 향상: 사람이 중간 결과 검토 후 피드백 제공 → 모델 응답 개선
 - 투명성: 실행 경로와 사람이 개입한 지점 기록 → 감사/로그 관리
- 활용 시나리오
 - 에이전트의 도구 호출: API 요청 전에 사람 승인 받기
 - 콘텐츠 생성: 보고서/코드 자동 작성 후 사람이 검토·수정
 - 안전 민감 작업: 보안 설정 변경, 재무 트랜잭션 등

실습: 505. Human-in-the-loop

- 결제 요청: 사용자가 특정 금액과 목적에 대해 결제 승인을 요청
- 워크플로우 일시정지: interrupt를 통해 관리자의 응답을 기다림
- 관리자 응답: 승인 또는 거부 결정
- 워크플로우 재개: 관리자의 응답에 따라 프로세스 계속 진행

장기 메모리 (Long-term Memory)

- LangGraph의 지속성(persistence) 기능을 사용하여 장기 메모리 구현
- JSON 문서(JSON document) 형태로 스토어(store)에 저장
- 메모리는 다음 두 가지로 구성
 - namespace (네임스페이스): 폴더처럼 데이터를 구분하는 단위
 - key (키): 파일 이름처럼 각 항목을 구분하는 식별자
- 네임스페이스에는 보통 사용자 ID, 조직 ID, 또는 데이터를 체계적으로 구분하기 위한 레이블이 포함

```
store.put(  
    ("users",), # 사용자 데이터를 모아두는 namespace  
    "user_123", # 사용자 ID (key)  
    {  
        "name": "John Smith",  
        "language": "English",  
    } # 사용자 정보  
)
```

실습: 510_Long_Term_Memory

- LangGraph Persistence를 통해 장기 메모리를 다루는 방법 이해
- 임베딩 및 Store 초기화
- 기억 저장
- 유사도 검색
- 도구(Tool)에서 읽기 / 쓰기

MCP

Model Context Protocol

Why MCP ?

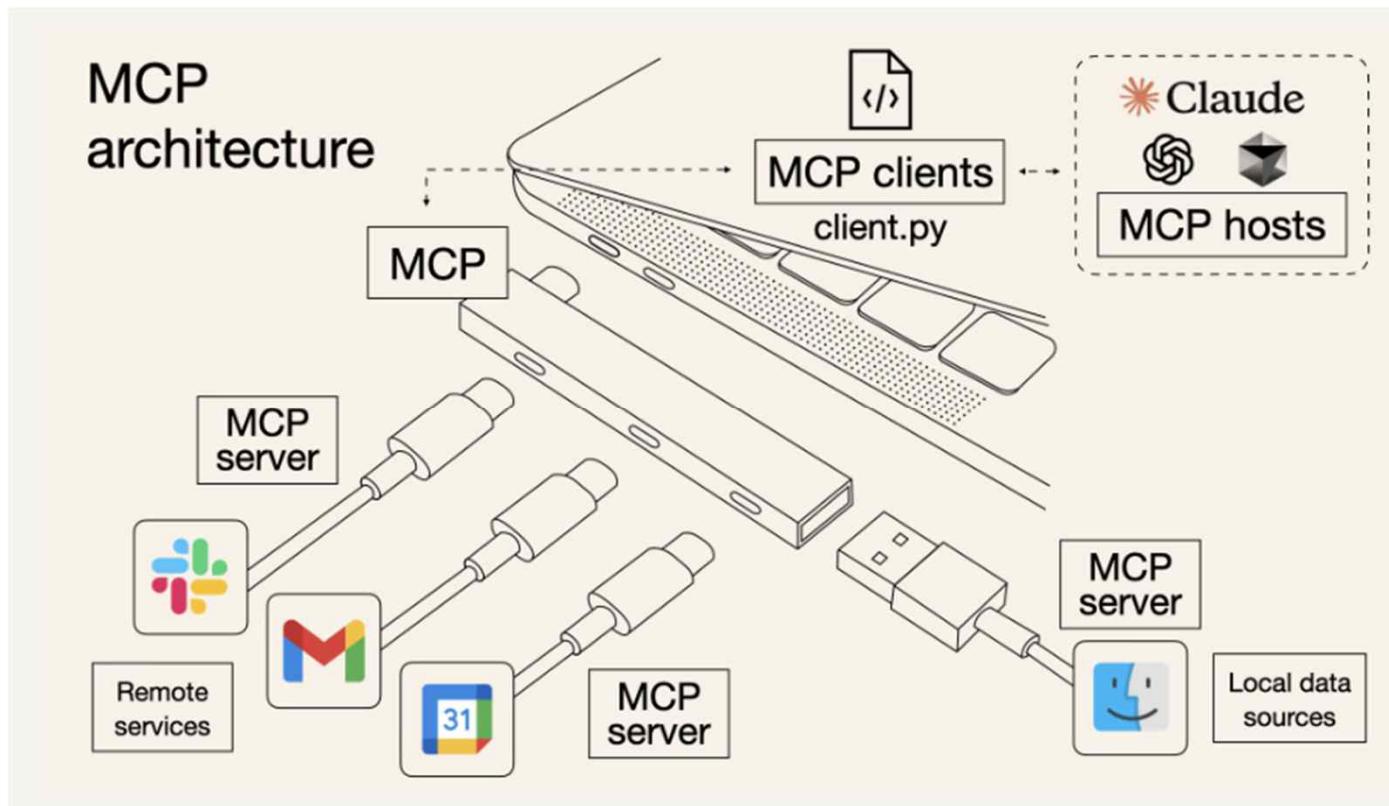
- LangChain 의 Tools 에 해당
(<https://python.langchain.com/docs/integrations/tools/>)
→ LangChain/LangGraph eco-system 내의 개발자용 (Framework 종속)
- MCP → 일반 사용자용 protocol
- Claude 가 표준 규약을 만들고 그 것만 지키면 모든 client 가 사용 가능

MCP protocol

- JSON-RPC 2.0 기반으로, 클라이언트(LLM)와 서버(도구 제공자)가 bidirectional, 상태 유지 방식으로 통신합니다
- MCP는 2024년 11월 Anthropic이 선언한 공통 프로토콜이며, JSON-RPC 2.0 기반의 bidirectional 메시지 통신 구조를 따릅니다
- OpenAI 및 Google DeepMind도 이 MCP 규약을 공식 채택하여, 자사의 에이전트 및 서비스에서 이 표준을 지원하고 있습니다.

MCP architecture

USB-C Type 같은 표준 규약



Cursor,
Claude Desktop 등

MCP 구현 구성 요소

- MCP server / MCP client
- Resources – AI 모델에게 필요한 정보와 컨텍스트를 제공
- Tools – AI 모델이 외부 시스템에서 작업을 수행할 수 있도록 도구 정의
- Prompts – AI 모델이 리소스와 도구를 활용하여 응답을 생성할 수 있도록 재사용 가능한 템플릿 제공

Resources (리소스)

- 역할: LLM(대형 언어 모델)에 데이터를 제공하는 인터페이스
→ REST API에서의 GET 엔드포인트와 유사
- AI 모델에게 필요한 정보와 컨텍스트를 제공 → 읽기 전용(read-only)
- 중대한 연산이나 부작용(side effect)이 없어야 함
- 유저 프로필 조회, 제품 목록 불러오기, 데이터베이스에서 값 읽기 등을 정의하는 안전한 조회용 API

Tools (툴)

- 역할: LLM이 행동을 수행하도록 하는 인터페이스
→ REST API에서의 POST/PUT/DELETE 엔드포인트와 유사
- AI 모델이 외부에서 작업을 수행할 수 있도록 도구 정의
- 부작용(side effect)이 발생할 수 있음
- 데이터 생성/수정, 이메일 발송, 결제 처리, 워크플로우 실행 등

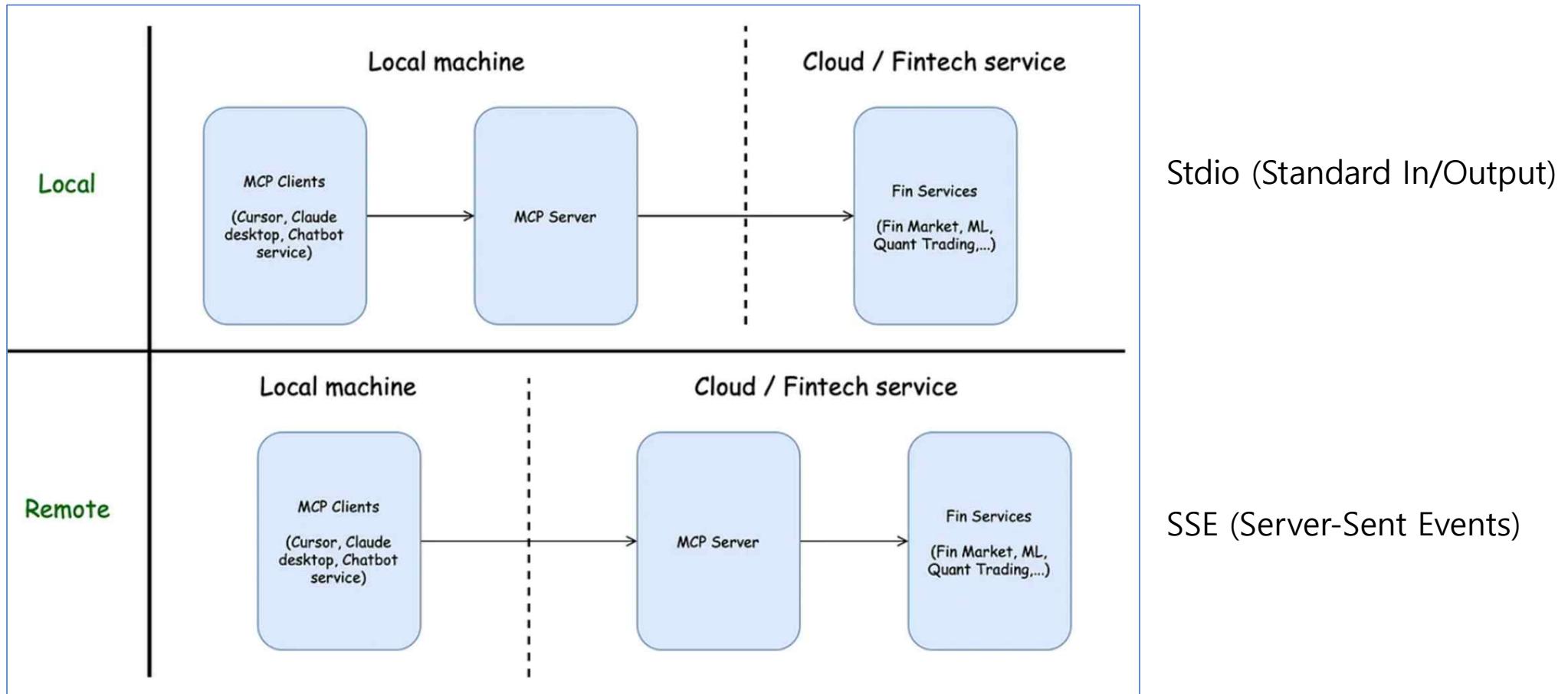
Prompts (프롬프트)

- 역할: LLM이 리소스와 툴을 더 잘 활용하도록 돋는 재사용 가능한 템플릿
→ 가이드 문구, 매크로, 템플릿 등
- 자주 쓰이는 질문/명령을 표준화
- 일관성 있는 요청을 보장
- 예: “사용자 ID를 입력받아 주문 목록을 조회하는 표준 질문 템플릿”
→ 즉, LLM이 요청할 때 실수를 줄이고, 일관된 결과를 내도록 하는 미리 작성된 대화 가이드입니다.

MCP 표준 전송 유형

- Stdio (Standard In/Output)
서버와 클라이언트가 동일한 프로세스 내에서 작동하거나 모델을 local에서 실행할 때 사용
- SSE (Server-Sent Events)
 - 서버와 클라이언트 간 효율적인 단방향 통신(HTTP 연결)을 통해 간단한 실시간 데이터 업데이트를 구현할 때 사용.
 - 모델의 steaming 출력을 실시간으로 받아야 할 때 사용.

Local/Remote MCP



Smithery – MCP Marketplace

The screenshot shows the homepage of the Smithery MCP Marketplace. At the top, there's a navigation bar with links for Docs, Blog, Playground, Deploy Server, and a user profile icon. Below the navigation, a large banner features the text "Your Agent's Gateway to the World" and "Integrate your AI with 7772 skills and extensions built by the community." A search bar contains the placeholder text "automate web browser interactio|". Below the banner, there are two sections: "Featured" and "Popular". The "Featured" section displays four cards with MCP servers: Supabase MCP Server, Memory Tool, ScrapeGraph MCP Server, and Notion. The "Popular" section shows a list of popular MCPs with a "View all" button. The overall design is dark-themed with white and light-colored text.

Docs Blog ✨ Playground Deploy Server

Your Agent's Gateway to the World

Integrate your AI with 7772 skills and extensions built by the community.

automate web browser interactio|

Featured 5 View all →

Supabase MCP Server @supabase-community/supabase-mcp

Connect your Supabase projects to AI assistants like Cursor and Claude. Manage...

Remote 21.17k

Memory Tool @mem0ai/mem0-memory-mcp

Store and retrieve user-specific memories to maintain context and make informed...

Remote 16.51k

ScrapeGraph MCP Server @ScrapeGraphAI/scrapegraph-mcp

ScrapeGraphAI is a LLM-powered scraping API for structured data extraction from any...

Remote 133

Notion @smithery/notion

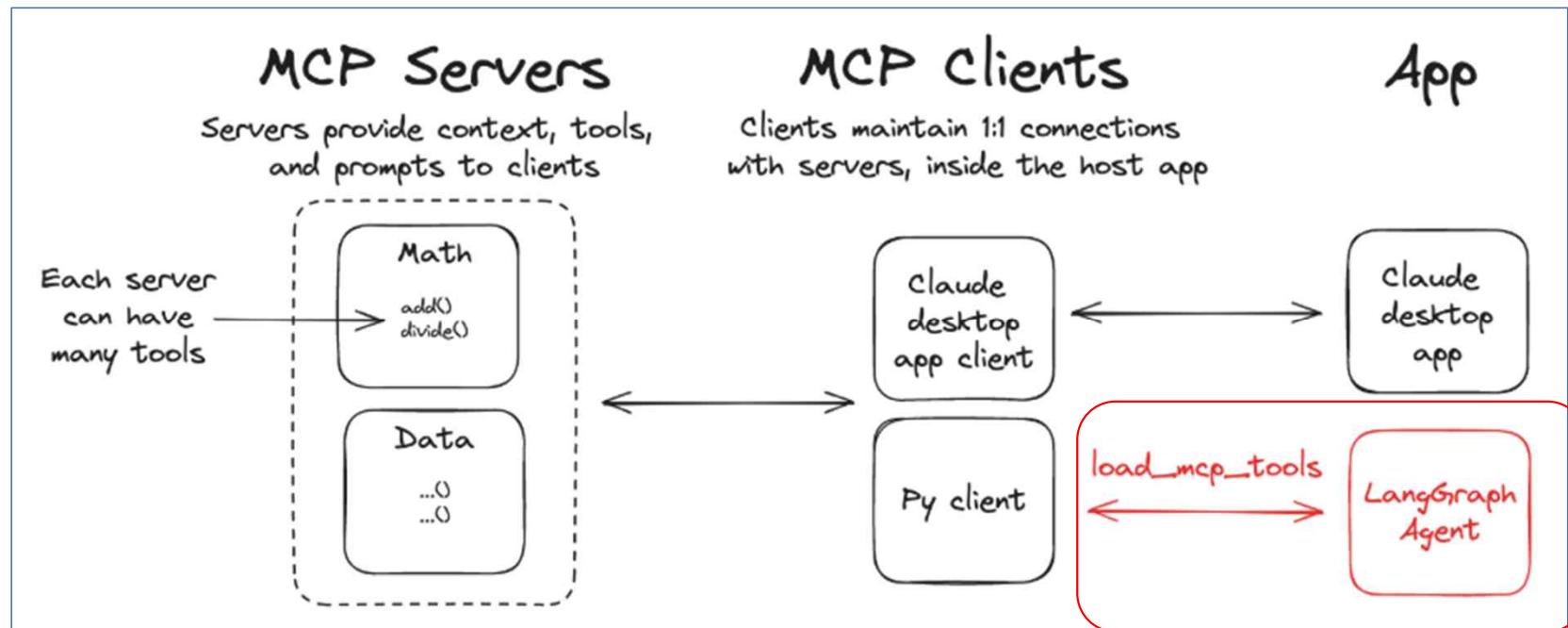
A Notion workspace is a collaborative environment where teams can organize...

Remote

Popular 2963 View all →

LangChain MCP Adapters

<https://github.com/langchain-ai/langchain-mcp-adapters>



- MCP 서버가 제공하는 리소스(Resources), 툴(Tools), 프롬프트(Prompts)를 다른 형식이나 표준에 맞게 변환 → LLM 또는 클라이언트가 이해할 수 있는 형식으로 바꿔 주거나, 반대로 클라이언트의 요청을 MCP 서버가 이해할 수 있도록 변환해 주는 브릿지(bridge) 역할
 - 프로토콜 변환
 - 클라이언트 ↔ MCP 서버 간 통신 형식을 맞춰 줌
 - 예: HTTP → MCP RPC, JSON → 특수 포맷 등
 - 데이터 변환 / 정규화
 - 리소스에서 오는 데이터를 표준 schema에 맞게 정리
 - 툴 실행 시 필요한 입력을 변환하거나 검증
 - 호환성 확보
 - 서로 다른 MCP 서버들이 동일한 방식으로 클라이언트에 노출될 수 있도록 인터페이스 통일

MCP 파이썬 SDK

- SDK (Software Development Toolkit)
 - 개발자가 특정 플랫폼용 앱을 개발할 수 있도록 하는 도구 모음
- 모든 MCP 서버에 연결할 수 있는 MCP 클라이언트를 구축할 수 있음
- 리소스, 프롬프트 및 도구를 노출하는 MCP 서버를 만들 수 있음
- stdio, SSE, Streamable HTTP와 같은 표준 전송을 사용
- 모든 MCP 프로토콜 메시지 및 수명 주기 이벤트를 처리

MCP Python SDK 설치

- <https://github.com/modelcontextprotocol/python-sdk>
- pip install "mcp[cli]"
- pip install langchain-mcp-adapters

MCP Server

```
from mcp.server.fastmcp import FastMCP

# Create an MCP server
mcp = FastMCP("Demo")
```

MCP server 지정

```
# Add an addition tool
@mcp.tool()
def add(a: int, b: int) -> int:
    """Add two numbers"""
    return a + b
```

MCP 도구 지정

```
# Add a dynamic greeting resource
@mcp.resource("greeting://{{name}}")
def get_greeting(name: str) -> str:
    """Get a personalized greeting"""
    return f"Hello, {name}!"
```

MCP가 가지고 있는 data, configuration 등 정의

MCP Client

```
client = MultiServerMCPClient(  
    {  
        # Chinook Database MCP 서버 (stdio 기반 로컬 실행)  
        "chinook": {  
            "transport": "stdio",          # 로컬 subprocess 통신  
            "command": "python",         # MCP 서버 실행 명령어  
            "args": ["./agent_server.py"], # MCP 서버 스크립트 경로  
        }  
    }  
)  
  
async def run():  
    print("===== MCP MultiServer 클라이언트 초기화 중... =====")  
  
    # MCP 서버 연결 및 도구 목록 가져오기  
    tools = await client.get_tools()  
  
    # 메모리 체크포인터 생성 (대화 히스토리 유지용)  
    memory = MemorySaver()  
  
    # LangChain React 에이전트 생성  
    agent = create_agent(  
        model=model,  
        tools=tools,  
        checkpointer=memory  
    )
```

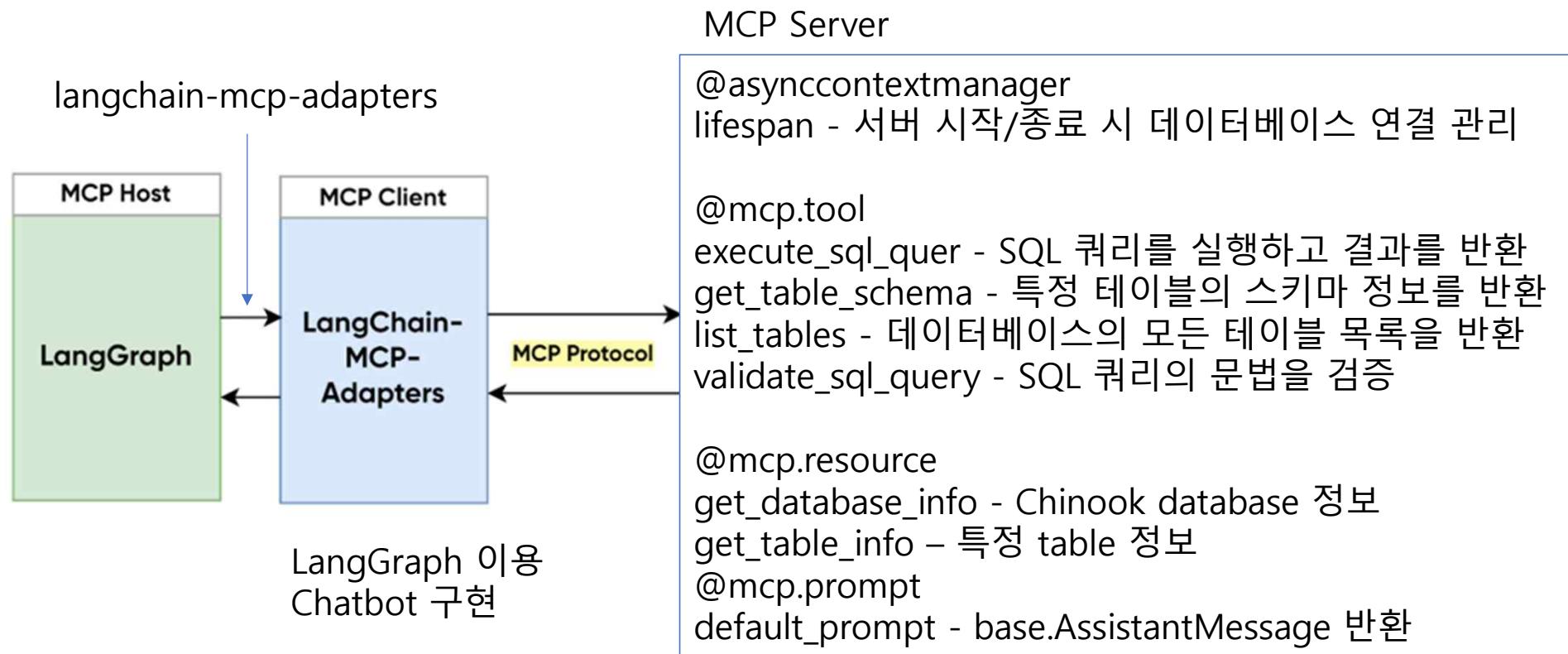
MultiServerMCPClient 설정

MCP 서버 연결

Cursor 설치 (<https://cursor.com/home>)

The screenshot shows the Cursor website's download page. At the top, there is a navigation bar with the Cursor logo, 'Features', 'Enterprise', 'Pricing', 'Resources', 'Sign in', and a 'Download' button. Below the navigation bar, a large text block reads: 'Built to make you extraordinarily productive, Cursor is the best way to code with AI.' A 'Download for Windows' button is located below this text. In the center, there is a screenshot of the Cursor application interface. The interface shows a file tree on the left with 'ML-RESEARCH-NOTEBOOK' containing 'notebooks', 'train_model.py', 'evaluation.py', 'experiments', and 'config.yaml'. The main area displays a code editor with three tabs: 'train_model.py', 'run_experiment.py', and 'config.yaml'. The 'train_model.py' tab contains Python code for PyTorch training. To the right of the code editor, there is a sidebar titled 'PyTorch MNIST Experiments' with a descriptive text box and a 'train_model.py +100-54' button. At the bottom of the sidebar, there is another text box with the text: 'Now let me update the evaluation module to save results and generate a detailed report.'

실습: DB agent 구현 - DB_MCP_Agent



서버 실행

```
(langchain) C:\Users\trimu\OneDrive\LangChain_LangGraph_MCP\LangGraph_MCP_Agent>python agent_server.py
MCP Server is running...
Chinook 데이터베이스 연결 성공
```

Client 실행

```
(langchain) C:\Users\trimu\OneDrive\LangChain_LangGraph_MCP\LangGraph_MCP_Agent>python agent_client.py
=====TESTING CHINOOK DB RESOURCES=====
Database Info Resource: {
    "database_name": "Chinook",
    "dialect": "sqlite",

=====RESOURCES TEST COMPLETE=====

=====대화형 Chinook 데이터베이스 분석 챗봇 시작 (메모리 포함)=====
종료하려면 'quit' 또는 'exit'를 입력하세요.
대화 히스토리가 기억됩니다!

질문을 입력하세요: █
```

- Resource 테스트 질문

- 1. Database Info Resource:

- 데이터베이스 정보를 보여줘"
 - "Chinook 데이터베이스에 대해 알려줘"
 - "데이터베이스 기본 정보를 확인해줘"

- 2. Table Info Resource:

- "Employee 테이블 정보를 보여줘"
 - "Customer 테이블의 구조를 알려줘"
 - "Album 테이블의 스키마를 확인해줘"

- Tools 테스트 질문

- 1. SQL Query Execution Tool:

- "직원이 몇 명인가요?"
 - "고객 수를 알려줘"
 - "앨범이 총 몇 개인가요?"
 - "가장 많이 팔린 곡은 무엇인가요?"
 - "매니저는 몇 명인가요?"

- **2. Table Schema Tool:**
 - "Employee 테이블의 구조를 보여줘"
 - "Customer 테이블의 컬럼 정보를 알려줘"
 - "Track 테이블의 스키마를 확인해줘"
- **3. List Tables Tool:**
 - "데이터베이스에 어떤 테이블들이 있나요?"
 - "사용 가능한 테이블 목록을 보여줘"
 - "모든 테이블 이름을 알려줘"
- **연속 대화 테스트:**
 - 1단계: "직원이 몇 명인가요?"
 - 2단계: "그 중에서 매니저는 몇 명인가요?"
- **컨텍스트 참조 테스트:**
 - 1단계: "고객이 몇 명인가요?"
 - 2단계: "그 고객들 중에서 미국 고객은 몇 명인가요?"
 - 3단계: "앞서 말한 고객들의 주문 건수는?"

실습: Notion MCP 구현하기 - Smithery

- npx로 Notion 공식 MCP 서버(@notionhq/notion-mcp-server) 실행
→ LangGraph ReAct 애이전트에 도구로 연결
- 환경 변수(.env)
 - NOTION_API_KEY → npx 서버에 NOTION_TOKEN으로 주입
 - NOTION_PAGE_ID(선택) → 기본 작업 페이지 안내/프롬프트에 자동 포함
- Node.js 가 미설치된 경우
<https://nodejs.org/ko/download> → Windows 설치프로그램(.msi) 다운로드

Notion API key 생성 - Notion의 개발자 설정(Integrations) 메뉴

The screenshot shows the Notion Developers API Reference page. It features a main heading "Start building with the Notion API" with a sub-subtitle "Connect Notion pages and databases to the tools you use every day, creating powerful workflows." Below this is a "Get started" button. To the right, there's a cartoon illustration of three people wearing hard hats and safety vests, sitting around a table with a hook hanging from the ceiling above them. At the top right of the page is a button labeled "View my integrations".

The screenshot shows the Notion API 통합 (Integrations) settings page. On the left, there's a sidebar with categories like General, Listings, and API 통합. The main area has a heading "API 통합" with a sub-subtitle "개발 정보와 로그인 정보를 생성하고, 검토하고, 편집할 수 있습니다." and a note "API 통합은 API를 사용하여 Notion의 페이지, 데이터베이스, 사용자에 접근합니다. 여기에서 자세히 알아보세요.". A large blue arrow points from the "View my integrations" button on the previous screen to the "New" button on this screen.

The screenshot shows the "New API Integration" creation form. It includes fields for "API 통합 이름" (API Integration Name) set to "MCP Server Test", "관련 워크스페이스" (Related Workspaces) showing "YoungJea Oh의 워크스페이스", "유형" (Type) set to "프라이빗" (Private), and a "Logo" section with a placeholder image for "업로드" (Upload). At the bottom are "취소" (Cancel) and "저장" (Save) buttons. A blue arrow points from the "New" button on the previous screen to this form.

The screenshot shows the MCP Server Test configuration page. It has tabs for "구성" (Configuration), "사용 권한" (Usage Permissions) set to "New", "기본 정보" (Basic Information), and "설정" (Settings). A warning message says "내 API 통합은 아직 어떤 페이지에도 접근할 수 없습니다" (My API integration cannot yet access any pages) and "API 통합을 사용하려면 Access tab에서 페이지와 데이터베이스를 추가하세요" (To use the API integration, add pages and databases in the Access tab). The "API 통합 이름" (API Integration Name) is "MCP Server Test". In the "프라이빗 API 통합 시크릿" (Private API Integration Secret) field, the value "ntn_588632592525RSOP51MCUH11B1C0E81UXVmbyu" is highlighted with a red border. A blue arrow points from the "New API Integration" form on the previous screen to this configuration page.

Notion Page 생성

page-id

- <https://www.notion.so/nobody-project-26cdb4860e7480468071d68f79141934>

nobody 팀의 project 보고서

nobody 팀의 project 보고서

안녕하세요

자바 스프링 프레임워크의 장점

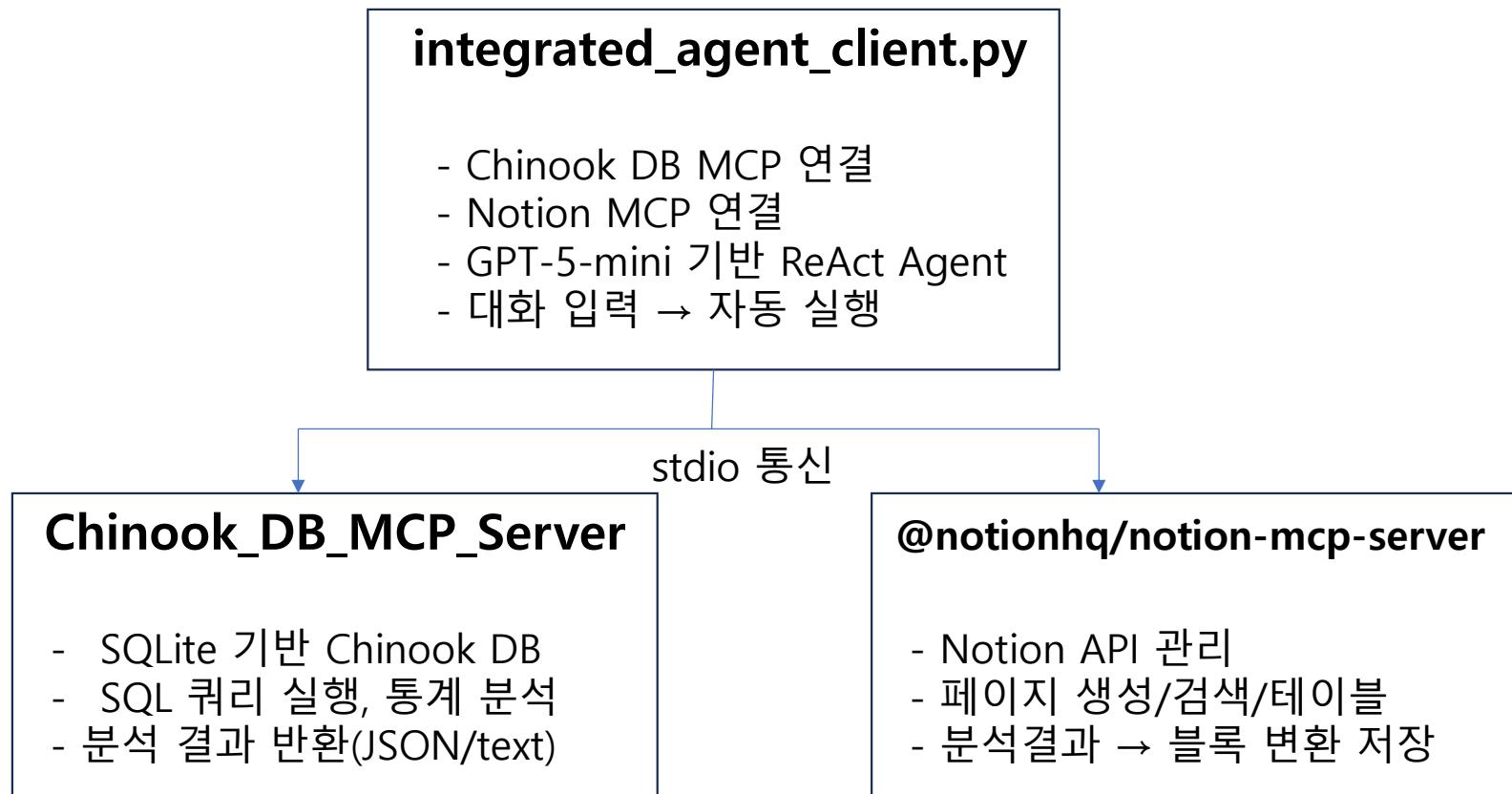
- 의존성 주입(DI): 객체 간의 결합도를 낮춰 유지보수성 향상
- 제어의 역전(IoC): 객체 생성과 생명주기 관리 자동화
- AOP(관점 지향 프로그래밍): 획단 관심사 분리로 코드 중복 제거
- 풍부한 생태계: Spring Boot, Spring Security
- 테스트 지원: 단위 테스트와 통합 테스트를 위한 툴
- 확장성: 마이크로서비스 아키텍처 구축에 적합
- 커뮤니티 지원: 활발한 오픈소스 커뮤니티와 풍부한 문서

활성화된 연결

- M MCP Server Test
생성자: YoungJea Oh
- + 연결 추가하기
- ⚙️ 연결 관리

업데이트와 애널리틱스
버전 기록
알림 받기
연결
Windows 앱에서 열기

실습: Notion + DB MCP 구현하기



- Notion Page 생성 및 API 통합 연결
- LangGraph Agent가 Chinook DB MCP 서버에서 데이터를 분석하고, Notion MCP 서버로 결과를 페이지·테이블 형태로 자동 기록하는 완전한 AI 기반 데이터→문서 자동화 파이프라인 구현
- 질문 예시) PlaylistTrack 의 schema 를 Notion 에 올려줘

Web Interface

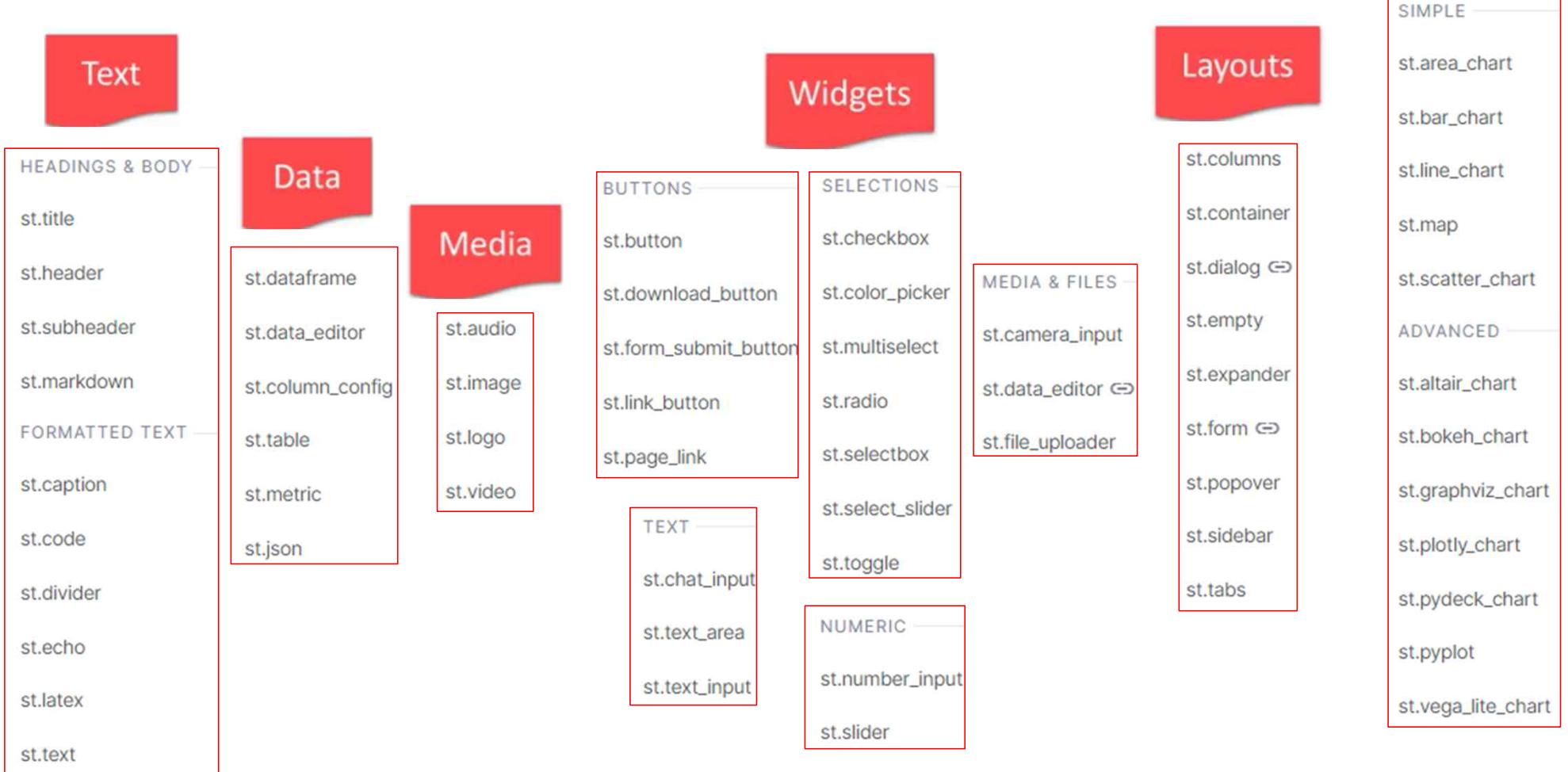
LangChain+Streamlit

What is Streamlit ?

- Machine Learning 과 Data Science를 위한 open-source application framework
- HTML, CSS, JavaScript와 같은 웹 기술을 알 필요 없이 파이썬으로 모든 것 처리
- 시각화 라이브러리와 쉽게 통합되어 다양한 데이터 시각화 가능
- Streamlit의 위젯을 사용하여 대화형 그래프와 대시보드를 만들 수 있다.
- 데이터가 변경될 때마다 실시간으로 애플리케이션을 업데이트
- 간단한 캐싱 메커니즘을 통해 빠른 성능을 유지
- 로컬에서 개발한 애플리케이션을 클라우드에서 손쉽게 배포하고 공유

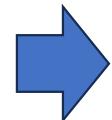
Overview of Streamlit API

Charts



Text Display

```
python app_01_text_display.py > ...
1  import streamlit as st
2
3  def main():
4      st.title("이것은 title입니다.")
5      st.header("이것은 header 입니다.")
6      st.subheader("이것은 subheader 입니다.")
7
8      st.text("Hello World Streamlit")
9      name = "김길동"
10     st.text("{}님, 안녕하세요. 반갑습니다.".format(name))
11
12     st.markdown("이것은 $\\frac{1}{x}$입니다.")
13
14
15 if __name__ == '__main__':
16     main()
```



이것은 **title**입니다.

이것은 **header**입니다.

이것은 **subheader**입니다.

Hello World Streamlit

김길동님, 안녕하세요. 반갑습니다.

이것은 $\frac{1}{x}$ 입니다.

st.write – 만능 function

5

5

```
def main():

    st.text(2+3)
    st.write(2+3)

    st.markdown("## 이것은 title입니다.")
    st.write("## 이것은 title입니다.")
    st.write("이것은  $\frac{1}{x}$ 입니다.")

    st.write(dir(st))
    st.help(st.write)

if __name__ == '__main__':
    main()
```



이것은 title입니다.

이것은 title입니다.

이것은 $\frac{1}{x}$ 입니다.

▶ [0 ~ 100]

▶ [100 ~ 182]

```
main() method streamlit.delta_generator.WriteMixin.write(*args: 'Any',
unsafe_allow_html: 'bool' = False, **kwargs) -> 'None'
```

Write arguments to the app.

This is the Swiss Army knife of Streamlit commands: it does different things depending on what you throw at it. Unlike other Streamlit commands,

Widgets

```
#Button/Radio Button/Checkbox>Select  
st.button('버튼 위젯')  
  
name = "김길동"  
if st.button("Submit"):  
    st.write("이름: {}".format(name))
```

버튼 위젯

Submit

이름: 김길동

```
if st.checkbox("show/hide"):  
    st.text("Showing something")
```

show/hide

Showing something

```
status = st.radio("선택", ("활성화", "비활성화"))  
if status == "활성화":  
    st.success("활성화 되었습니다.")  
else:  
    st.warning("비활성화 되었습니다.")
```

선택

활성화

비활성화

활성화 되었습니다.

```
# Slider  
age = st.slider("나이", 1, 100, 50)  
st.write("나이는 {}입니다.".format(age))
```

나이

1 37 100

나이는 37입니다.

Text Input

```
# Text Input
name = st.text_input("이름을 입력하세요.")
st.write(name)

password = st.text_input("비밀번호를 입력하세요.", type="password")
st.write(password)
```

이름을 입력하세요.

김길동

김길동

비밀번호를 입력하세요.

.....



yeMryTDvb3kxvwR

```

# Text Area
message = st.text_area("메시지를 입력하세요.", height=100)
st.write(message)

# Numbers
number = st.number_input("숫자를 입력하세요.")
st.write(number)

# Date Input
appointment_date = st.date_input("약속 일자를 입력하세요.")
st.write(appointment_date)

```

메시지를 입력하세요.

안녕하세요. 좋은 아침입니다.

안녕하세요. 좋은 아침입니다.

숫자를 입력하세요.

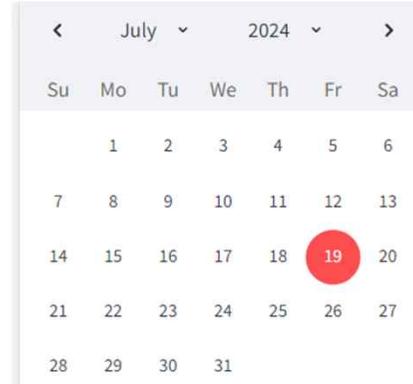
300.00

300.0

약속 일자를 입력하세요.

2024/07/19

2024-07-19



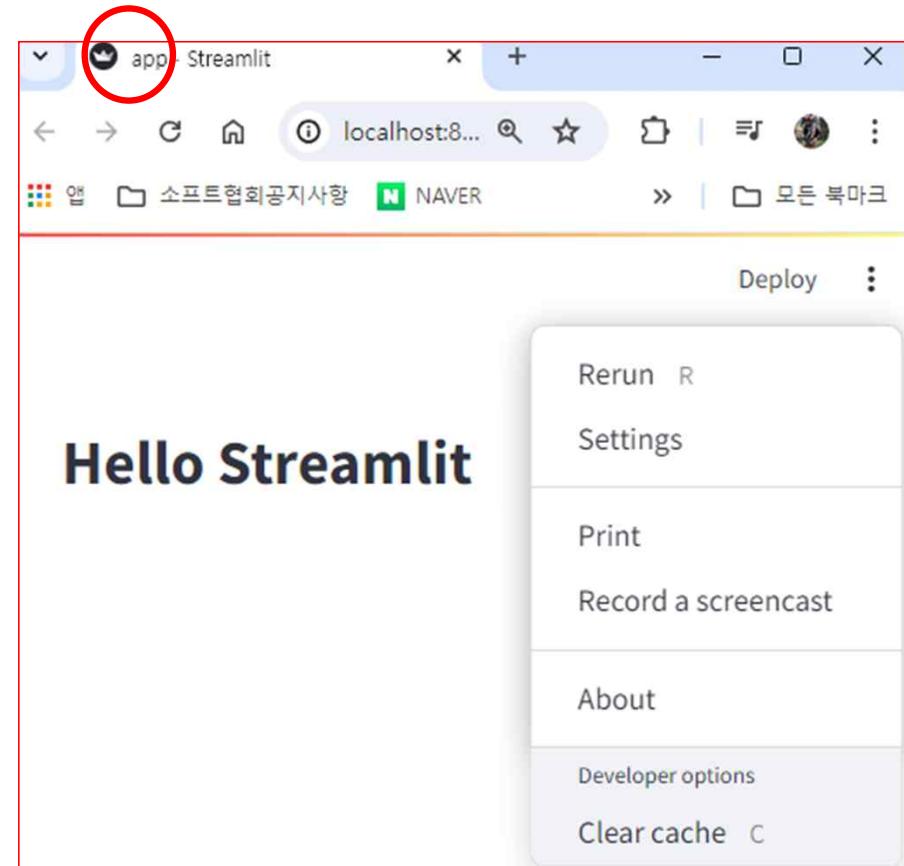
2024/07/19

2024-07-19

Streamlit 실행

- streamlit run app.py

```
app.py > ...
1 import streamlit as st
2
3 def main():
4     st.title("Hello Streamlit")
5
6 if __name__ == '__main__':
7     main()
```



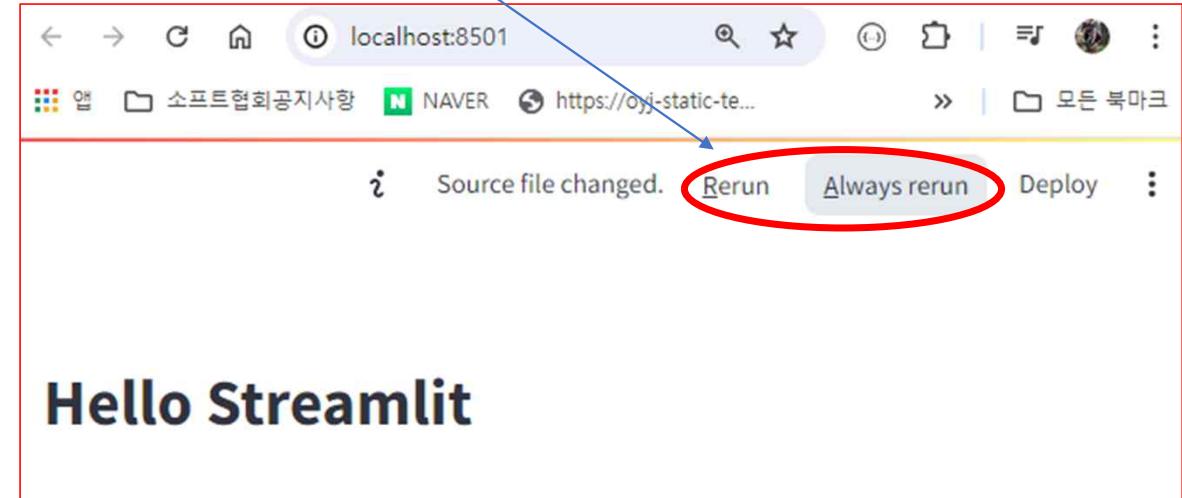
```
(base) C:\Users\trimu\OneDrive\streamlit\LearnStreamlit-Udemy\Module04>streamlit run app.py
```

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.30.1.40:8501

실시간 application update

```
app.py > ...
1  import streamlit as st
2
3  def main():
4      st.title("Hello Streamlit")
5
6  if __name__ == '__main__':
7      main()
```



```
app.py > ...
1  import streamlit as st
2
3  def main():
4      st.title("반가워요 Streamlit")
5
6  if __name__ == '__main__':
7      main()
```

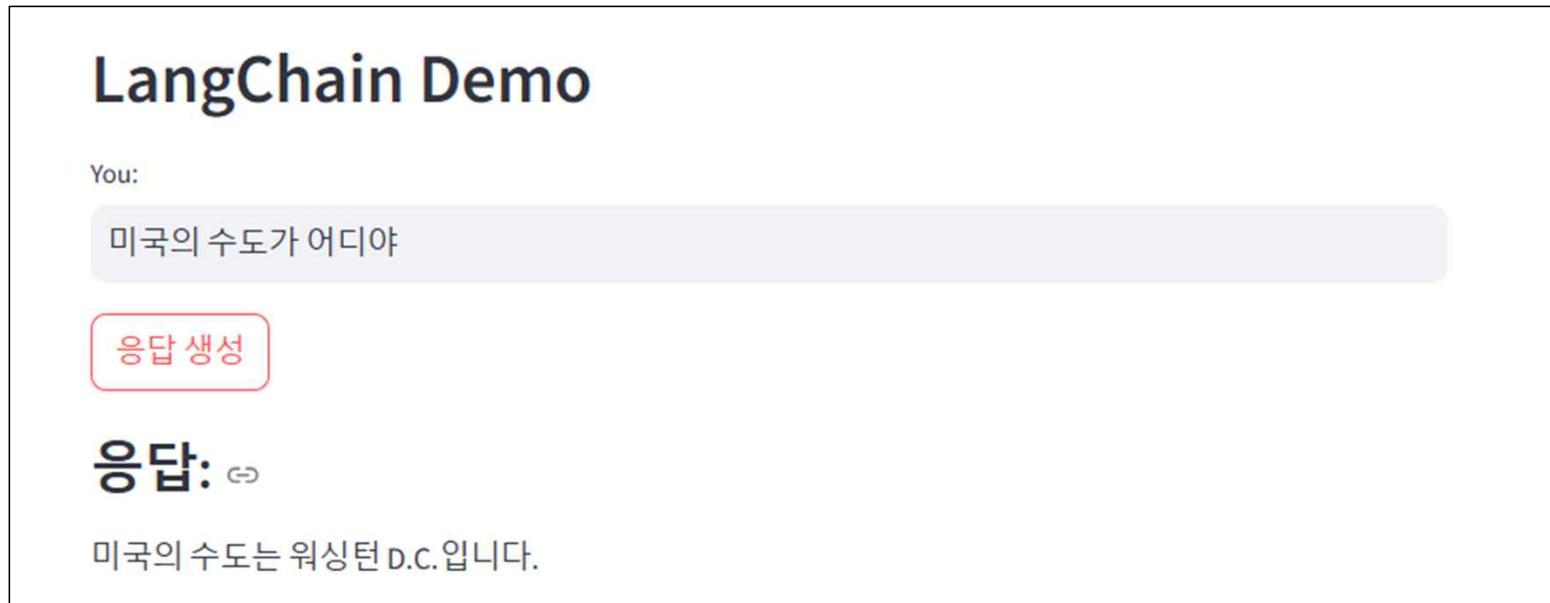
반가워요 Streamlit

App 작성 순서

1. title 추가
2. Sidebar 추가
3. 입력 area widget 추가
4. Button widget 추가
5. OpenAI API 함수 추가
6. Button widget click 과 함수 연결

실습: 010_intro.py

- Streamlit 을 이용한 UI 구성
- LangChain 라이브러리를 사용하여 자연어 처리 수행

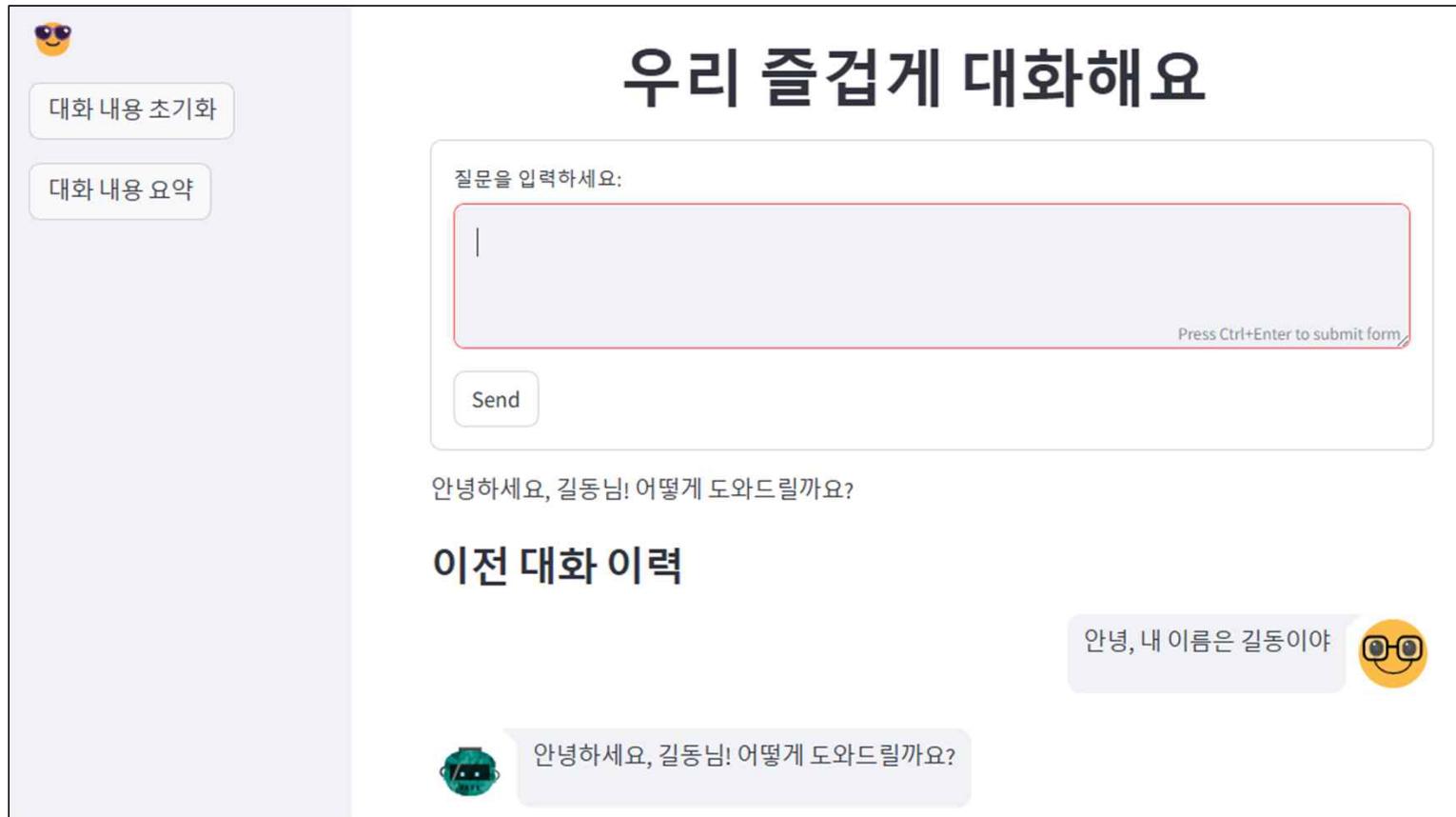


Session state 관리

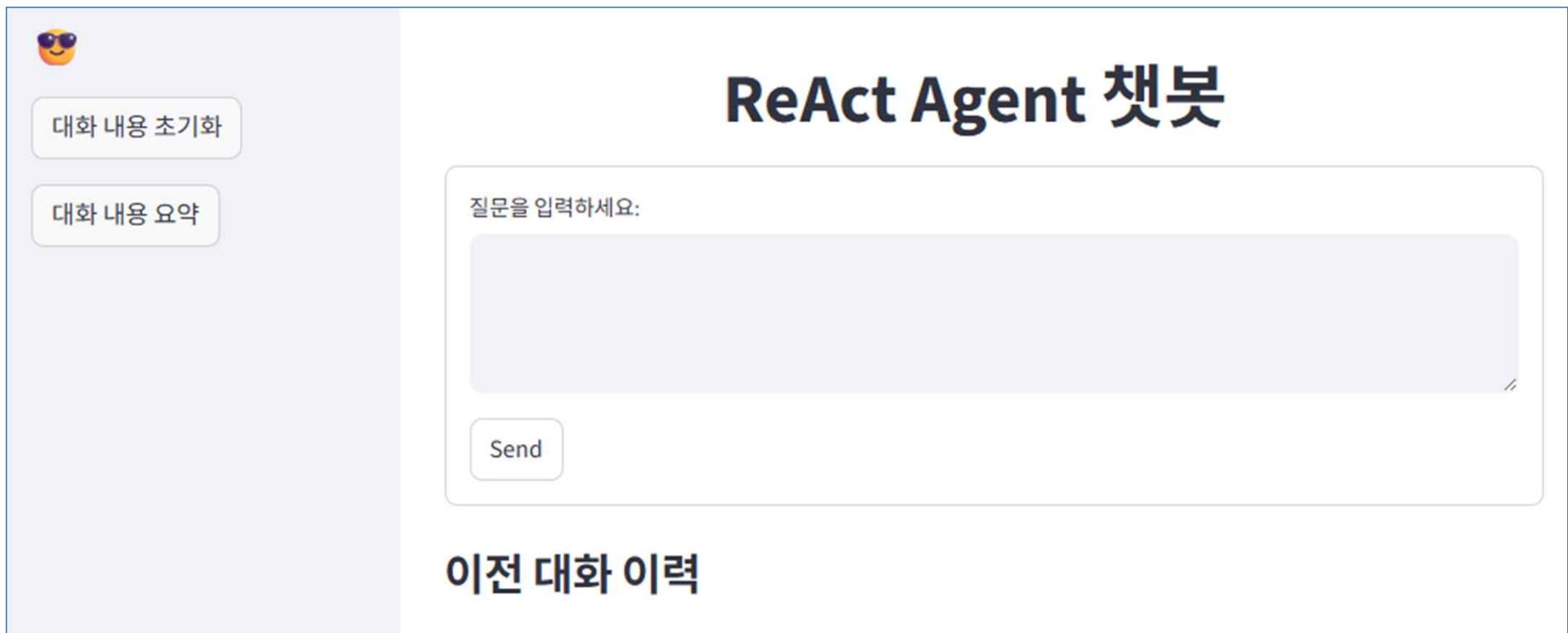
- Streamlit의 st.session_state :
Streamlit에서 세션별로 데이터의 상태를 저장하고 유지하는 기능
- LangGraph의 StateGraph :
LangChain에서 대화 상태와 히스토리를 관리하기 위한 객체

역할	st.session_state (Streamlit)	StateGraph (LangGraph)
메시지 저장	st.text_area()를 통해 입력값을 받아 메시지 리스트를 저장하고 화면에 출력	LangGraph 노드를 이용해 workflow.invoke()로 LLM 호출 후 응답 자동 저장
대화 상태 유지	session_state.messages 리스트를 통해 이전 대화 유지	MemorySaver()를 통해 내부적으로 상태 관리

실습: 050_ChatGPT 만들기 – LangChain 방식



실습: 060_Agent Chatbot 만들기

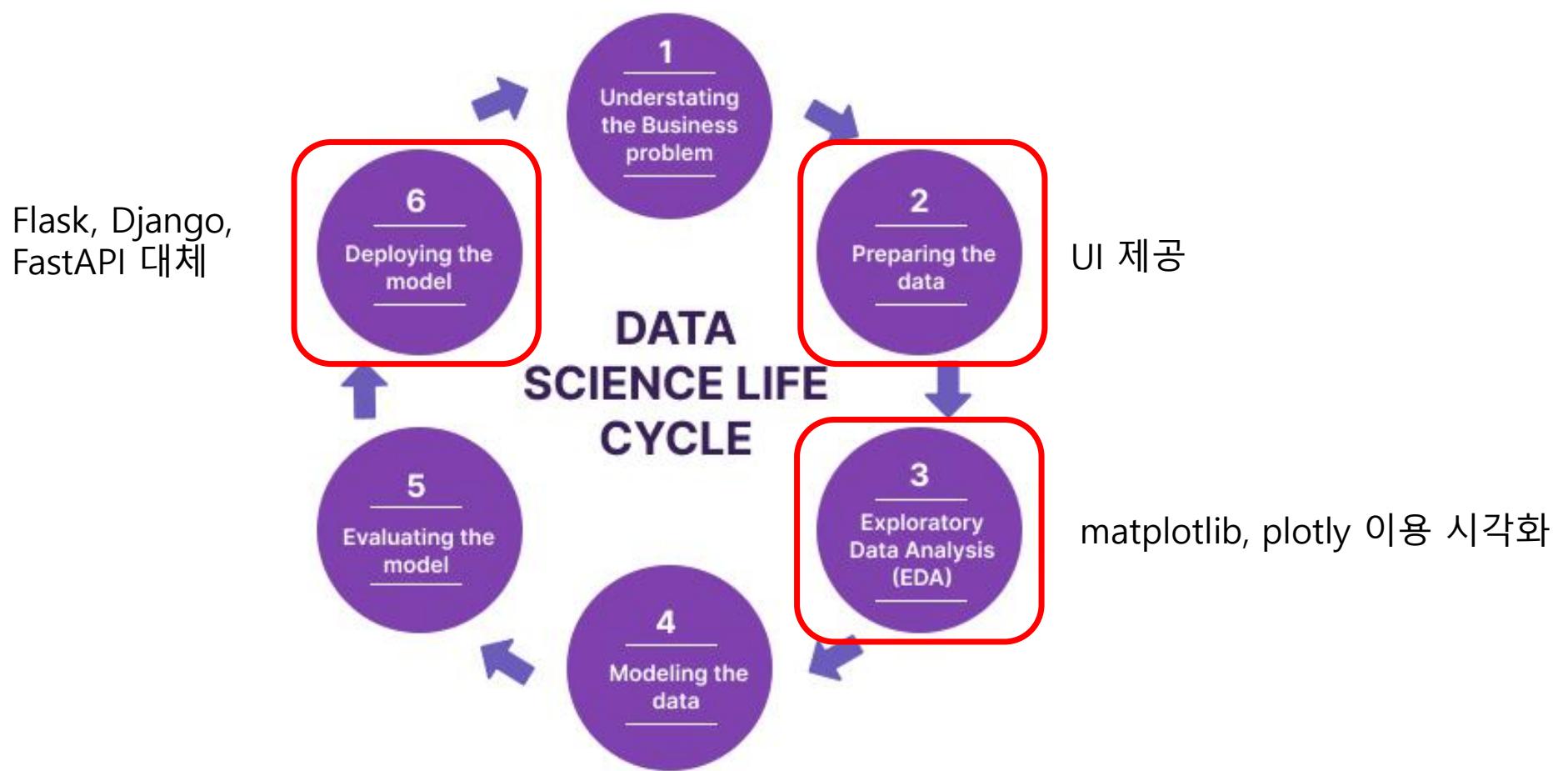


실습: 070_MCP_Agent Chatbot 만들기

The image shows a screenshot of a chatbot application interface. On the left side, there is a vertical sidebar with a dark blue header containing a yellow smiley face icon with sunglasses. Below the header, there are two white buttons with rounded corners: the top one is labeled "대화 내용 초기화" and the bottom one is labeled "대화 내용 요약". The main content area has a light gray background. At the top center, the text "통합 MCP Agent 챗봇" is displayed in a large, bold, dark font. Below this, there is a white input field with a placeholder text "질문을 입력하세요:" and a small "Send" button at the bottom left of the field. At the bottom left of the main area, the text "이전 대화 이력" is visible.

부록: Streamlit 개요

Streamlit 적용 범위



API 카테고리별 설명

- Text (텍스트)
 - 헤딩 및 본문: title, header, subheader, markdown 등을 사용하여 제목과 본문 작성
 - 서식 있는 텍스트: caption, code, text 등을 사용하여 코드를 포함한 다양한 텍스트 형식을 추가
- Data (데이터)
 - 데이터 출력: dataframe, data_editor, table, metric 등을 사용하여 데이터 프레임, 테이블, 메트릭 등을 화면에 표시
 - 구성: column_config를 사용하여 데이터의 컬럼 구성을 설정

- Media (미디어)
 - 이미지 및 비디오 출력: image, video, audio, logo 등을 통해 이미지, 비디오, 오디오 파일을 화면에 삽입
- Widgets (위젯)
 - 버튼 및 링크: button, download_button, form_submit_button, link_button, page_link 등을 통해 사용자 인터랙션을 추가
 - 선택 항목: checkbox, radio, electbox, multiselect 등을 사용하여 다양한 선택 옵션을 제공
 - 텍스트 입력: text_input, text_area, chat_input 을 통해 텍스트 입력
 - 숫자 및 슬라이더: number_input, slider를 통해 숫자를 입력하거나 선택
 - 미디어 및 파일: file_uploader, camera_input을 통해 파일 및 카메라 입력

- Layouts (레이아웃)
 - 레이아웃 구성 요소: columns, container, sidebar, expander, tabs 등을 사용하여 화면 레이아웃을 구성
- Charts (차트)
 - 기본 차트: area_chart, bar_chart, line_chart 등을 통해 기본적인 차트를 빠르게 생성
 - 고급 차트: altair_chart, plotly_chart, bokeh_chart 등 외부 라이브러리와 연동하여 고급 시각화

기타 Streamlit 주요 기능

- st.set_page_config() → 페이지 설정 (제목, 아이콘 등)
- st.sidebar.button() → 사이드바에 버튼 추가
- st.form() → 입력 폼(Form) 생성
- st.text_area() → 사용자 입력 필드
- st.form_submit_button() → 제출 버튼
- st.session_state → 상태(State) 관리 (대화 이력 저장)

Data Display

```
df = pd.read_csv("iris.csv")  
  
# # Dynamic display  
st.dataframe(df)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa

```
# 각 열에서 최대값을 강조  
st.dataframe(df.style.highlight_max(axis=0))
```

	sepal_length	sepal_width	petal_length	petal_width	species
13	4.300000	3.000000	1.100000	0.100000	setosa
14	5.800000	4.000000	1.200000	0.200000	setosa
15	5.700000	4.400000	1.500000	0.400000	setosa
16	5.400000	3.900000	1.300000	0.400000	setosa

```
# 크기 지정  
st.dataframe(df, 300, 100)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5			
1	4.9	3			

```
st.write(df.head())
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa

```
# Static display  
st.table(df)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1000	3.5000	1.4000	0.2000	setosa
1	4.9000	3.0000	1.4000	0.2000	setosa
2	4.7000	3.2000	1.3000	0.2000	setosa
3	4.6000	3.1000	1.5000	0.2000	setosa
4	5.0000	3.6000	1.4000	0.2000	setosa

```
# Select/Multiple Select
languages = ["Python", "Java", "C++", "C#"]
choice = st.selectbox("언어 선택", languages)
st.write("{}을 선택하셨습니다.".format(choice))
```

언어 선택

Python

Python

Java

C++

C#

```
my_languages = st.multiselect("내가 아는 언어들", languages, default="Java")
st.write(my_languages)
```

내가 아는 언어들

Java X C++ X C# X

▼ [

0 : "Java"

1 : "C++"

2 : "C#"

]

감사합니다.