생성형 AI 기반 RAG 개발자 과정

생성형 AI 기반 RAG 개발자 과정

Session 0 / 33 — OT + 개발환경

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

출처: Day 1 | OT + 환경

4일을 마치면 만들 수 있는 것

Day 1 — LLM 이해와 핵심 이론

- ✅ **Transformer · Attention · Q·K·V** 직관
- ✅ **GPT·RLHF·MoE·추론모델** 진화 이해
- ✅ OpenAI **API · 챗봇** 첫 코드
- ✅ **임베딩·코사인**으로 미니 RAG 직접 구현

Day 2 — LangChain 기반 RAG

- ✅ **LCEL · PromptTemplate** 파이프라인
- ✅ **Loader · Splitter** (TXT/PDF/DOCX)
- ✅ **FAISS · ChromaDB** 인덱스
- ✅ **Retriever + RAG Chain** + 출처·평가

Day 3 — 실제 RAG 웹서비스

- ✅ **Flask** REST 서비스 구조 설계
- ✅ **문서 업로드** (multipart, PDF/TXT/DOCX)
- ✅ **ChromaDB CRUD** — 색인·조회·삭제
- ✅ **질의응답 API + Chat History** 챗봇 완성

Day 4 — Agent 기반 서비스

- ✅ **AI Agent** 개념과 ReAct 루프
- ✅ **Tool 호출** — Wikipedia · Math · Custom
- ✅ **LangGraph** · Anthropic **5대 패턴**
- ✅ **Agentic RAG** + RAG 서비스에 Agent 결합

📦 강의 자료 & 실습 예제 다운로드

🐍 실습 예제 (일자별)

- Day 1~4 **빈칸 실습 템플릿** + 웹 데모 static(HTML/CSS/JS) + README
- 슬라이드의 코드 스니펫을 **직접 따라치며** 완성하는 파일
- 필요한 일자만 받아 압축 해제

📥 **Day 1** · **Day 2** · **Day 3** · **Day 4**

> 통합 교안 PDF·실습 묶음은 **인덱스 페이지의 「📥 강의 자료 다운로드」 카드**에서도 받을 수 있습니다.

📑 강의 슬라이드 (PDF)

- 각 회차는 **뷰어에서 PDF 다운로드** 가능 (인쇄·복습용)

⚙️ 시작하기

1. 해당 일자 압축 해제 — `tar xzf day1.tar.gz`
2. `.env` 에 `OPENAI_API_KEY` 설정
3. Day 폴더에서 실습 진행

대상 학습자

🎯 이 코스가 맞는 대상

- **Python 기본 문법**을 알고 있는 개발자
- 웹 백엔드(Flask/Django) 경험이 있거나 익숙해질 의지가 있는 학습자
- **API 호출 + JSON 처리** 정도는 직접 해 본 적 있는 분
- LLM 을 자신의 서비스/사이드 프로젝트에 결합하고 싶은 분
- LangChain·RAG·Agent 를 들어만 봤지 아직 **실제 코드로 만들어 본 적 없는** 분

🛑 이 코스가 적합하지 않은 대상

- Python 을 처음 배우는 분 (별도 입문 과정 권장)
- **수식 위주 ML 이론**을 원하는 분 (이 코스는 응용 위주)
- 모델 학습/파인튜닝이 주 목적인 분 (별도 코스)

> 🔑 **코스 목표**
>
> "4일 안에 직접 가진 데이터로 **RAG + Agent 데모**를 돌리는 것."

출처: Day 1 | OT + 환경

4일 × 33세션 전체 지도

📅 Day 1 (S0~8) — LLM 이해와 핵심 이론

`OT/환경` · `LLM 발전사` · `Transformer` · `GPT·RLHF·MoE` · `OpenAI API` · `챗봇` · `RAG 개요` · `임베딩·유사도` · `미니프로젝트`

📅 Day 2 (S9~16) — LangChain 기반 RAG

`LangChain/LCEL` · `Prompt` · `Loader` · `Splitter` · `Vector DB·FAISS` · `ChromaDB` · `Retriever` · `RAG Chain`

📅 Day 3 (S17~24) — 실제 RAG 웹서비스

`Flask 구조` · `업로드` · `파싱·임베딩` · `Chroma 조회(R)` · `삭제(D)` · `질의응답 API` · `Chat History` · `최종 프로젝트`

📅 Day 4 (S25~32) — Agent 기반 서비스

`Agent 입문` · `Tool` · `LangGraph` · `Agentic 5패턴` · `Agentic RAG` · `Agent+RAG 결합` · `로컬·배포` · `최종 회고`

> 🎯 **흐름**: 이론(왜) → API·챗봇 → RAG 개념 → LangChain RAG → 웹서비스화(CRUD) → 자율성(Agent)

PART A

PART A
개발환경 셋업

conda 환경 · 의존성 · .env · IDE — 약 50분

Python 3.12 — 본 코스 권장 버전

✅ 3.12 인 이유

- **LangChain 1.x / LangGraph** 의 권장 환경
- **PyTorch · sentence-transformers** 안정 지원
- 성능 개선 (3.11 대비 약 11% 향상)
- 3.13 은 일부 ML 패키지 호환성이 아직 부족

❌ 권장하지 않는 버전

- **3.9 이하** — 일부 LangChain 기능 미지원
- **3.10** — 호환은 되지만 deprecation 경고가 늘어남
- **3.13** — 일부 native 패키지 빌드 실패 가능

🎯 본 코스가 가정하는 환경

항목	값
OS	Windows / macOS / Linux 무관
Python	**3.12.x**
패키지 관리	**conda + pip**
가상환경 이름	`py312_gpt`
IDE	VS Code 또는 Cursor

> 💡 macOS Apple Silicon: 별도 설정 없이 native arm64 로 설치됩니다.

conda 설치 — Miniconda vs Anaconda

🪶 Miniconda (권장)

- 약 **400MB**
- conda + Python 만 포함
- 필요한 패키지만 추가 설치 → 디스크 절약
- 다운로드: docs.anaconda.com/miniconda

🐘 Anaconda

- 약 **3GB** — pandas/numpy/scikit-learn 등 250+ 패키지 포함
- 초보자에게는 편리하지만, 본 코스에서는 사용하지 않는 패키지가 더 많음
- 다운로드: anaconda.com/download

💻 OS 별 설치 (Miniconda 기준)

```
# macOS (Apple Silicon)
brew install --cask miniconda

# Linux
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh

# Windows
# 위 URL 에서 .exe 다운로드 → 더블클릭 설치
# "Add Miniconda3 to PATH" 체크 권장 (개발자라면 OK)
```

`py312_gpt` 환경 만들기

명령어 세 줄이면 끝

```
# 1. 환경 생성 — Python 3.12
conda create -n py312_gpt python=3.12 -y

# 2. 활성화
conda activate py312_gpt

# 3. 확인
python --version          # → Python 3.12.x
which python              # → ~/miniconda3/envs/py312_gpt/bin/python
```

자주 만나는 셸 이슈

```
# conda activate 가 동작하지 않을 때 (bash/zsh)
conda init bash           # 또는 zsh
exec $SHELL               # 셸 재시작 없이 반영

# Windows PowerShell
conda init powershell
# 그런 다음 PowerShell 재시작
```

> 💡 **환경 이름 규칙**: 본 코스의 모든 코드/README 가 `py312_gpt` 를 가정합니다. 다른 이름을 쓰면 안내 명령의 환경명을 그에 맞게 바꿔 쓰세요.

핵심 의존성 한 번에

설치 — pip install

```
conda activate py312_gpt

# Day 1 ~ Day 3 전체 의존성
pip install \
    openai==1.* \
    anthropic==0.40.* \
    python-dotenv \
    flask requests tiktoken \
    langchain==1.* \
    langchain-openai \
    langchain-anthropic \
    langchain-community \
    langchain-text-splitters \
    langchain-chroma \
    faiss-cpu chromadb pypdf \
    sentence-transformers \
    langgraph \
    wikipedia duckduckgo-search
```

검증 — 모두 import 되는지

```
python -c "
import openai, anthropic, langchain, langgraph
import faiss, chromadb, tiktoken, pypdf
print('✅ 모든 핵심 패키지 import 성공')
print('openai:', openai.__version__)
print('langchain:', langchain.__version__)
"
```

> ⚠️ Windows + `faiss-cpu` 실패 시 `pip install --no-cache-dir faiss-cpu` 또는 `conda install -c pytorch faiss-cpu` 시도.

본 코스 실습 코드

```
git clone https://github.com/lovehyun/tutorial-genai.git
```

📁 주요 폴더 매핑

폴더	다루는 세션
`1.openai/1.intro/`	Day1 S4 — REST / SDK old / SDK new
`1.openai/2~5.chatbot*/` · `8.streaming/`	Day1 S5 — 챗봇 (history · sqlite · session · SSE)
`2.langchain/7.RAG/1.basics/`	Day1 S7·8 — 임베딩·유사도·미니 RAG
`2.langchain/1.llm_models~4.chaining/`	Day2 S9·10 — LangChain · LCEL · Prompt
`2.langchain/7.RAG/2.loaders/`	Day2 S11·12 — Loader · Splitter
`2.langchain/7.RAG/3.vectorstore/`	Day2 S13·14 — FAISS · ChromaDB
`2.langchain/7.RAG/4.rag_chain/`	Day2 S15·16 — Retriever · RAG Chain
`2.langchain/7.RAG/8.web_app/`	**Day3 S17~24 — Flask RAG 웹서비스 (CRUD)**
`2.langchain/7.RAG/5.conversational/`	Day3 S23 — Chat History RAG
`2.langchain/8.agents/`	Day4 S25·26·28 — Agent · Tool · 패턴
`2.langchain/9.langgraph/`	Day4 S27 — LangGraph
`2.langchain/7.RAG/6.agentic/` · `7.local_model/`	Day4 S29·31 — Agentic RAG · 로컬

API Key 는 `.env` 에 — 코드에 하드코딩하지 말 것

📄 `.env` 파일 (저장소 루트)

```
# OpenAI
OPENAI_API_KEY=sk-proj-...

# Anthropic (Day 3 에서 사용)
ANTHROPIC_API_KEY=sk-ant-...

# Google Gemini (옵션)
GOOGLE_API_KEY=...

# LangSmith (옵션 — Agent 추적)
LANGCHAIN_TRACING_V2=true
LANGCHAIN_API_KEY=...
```

🔒 `.gitignore` 에 반드시

```
.env
.env.*
*.key
```

🐍 Python 에서 불러오기

```
# pip install python-dotenv
from dotenv import load_dotenv
import os

# 같은 폴더의 .env
load_dotenv()

# 상위 폴더의 .env (1.openai/1.intro 에서 루트 .env 참조)
load_dotenv(dotenv_path='../.env')

api_key = os.getenv('OPENAI_API_KEY')
```

> 💡 본 코스 코드는 대부분 **루트** `.env` 를 참조합니다 (`load_dotenv(dotenv_path='../.env')`).

API Key — 깃(git)에 올리면 5분 안에 도용

#	원칙	위반 시
1	`.env` 에 저장, `.gitignore` 에 추가	GitHub 봇이 5분 내 키 수집 → 무단 사용 → 청구서 폭탄
2	**결제 한도(spend limit)** 를 무조건 설정	도용 발생해도 손해 최소화
3	**Usage 알림** 이메일 활성화	비정상 사용 즉시 감지
4	노출 의심되면 **즉시 폐기 후 재발급**	시간 끌수록 손해 커짐
5	다른 사람과 **절대 공유 금지** — 팀이면 키 별도 발급	누가 썼는지 추적 불가

🚨 사고 사례

- OpenAI: 한 학생이 API 키를 코드에 하드코딩한 채 `git push` → 2시간 만에 봇이 발견 → 약 600달러 누적
- AWS: 동일 패턴으로 EC2 마이닝 → 며칠 만에 수만 달러

PART B

PART B
IDE 선택

VS Code · Cursor · Claude Code — 약 10분

VS Code vs Cursor vs Claude Code

항목	VS Code	Cursor	Claude Code
가격	무료	월 $20 (Pro)	API 종량제
AI 모델	GitHub Copilot (별도)	GPT-4o · Sonnet · 자체	Claude (Opus/Sonnet/Haiku)
Python 지원	확장으로 우수	확장으로 우수 (VS Code 기반)	CLI 기반, 모든 에디터 무관
본 코스 적합도	⭐⭐⭐ — 기본기	⭐⭐⭐⭐ — AI 페어 코딩	⭐⭐⭐⭐ — 터미널 워크플로우
추천 대상	처음 IDE 고르는 분	AI 보조에 익숙해지고 싶은 분	터미널 작업이 손에 익은 분

본 코스 강의 시연 환경

- **VS Code + Python 확장** 으로 시연 (가장 보편적)
- 다른 IDE 가 익숙하면 그걸로 진행해도 됩니다

본 코스에서 도움 되는 5가지

확장	역할
Python (Microsoft)	인터프리터, 디버거, 린팅 — 필수
Pylance	타입 추론, 자동완성 — Python 확장 설치 시 자동
Jupyter	`.ipynb` 노트북, 인라인 결과 — 회차 4 부터 권장
Ruff	빠른 린터·포매터 (PEP8 자동 정리)
dotenv	`.env` 파일 syntax highlight

설치 — 한 번에

```
code --install-extension ms-python.python
code --install-extension ms-toolsai.jupyter
code --install-extension charliermarsh.ruff
code --install-extension mikestead.dotenv
```

인터프리터 선택

```
Ctrl + Shift + P
  → Python: Select Interpreter
  → py312_gpt
```

> 💡 **확인 방법**: VS Code 좌하단에 `Python 3.12.x ('py312_gpt')` 가 표시되어야 OK.

PART C

PART C
첫 실행

Hello, LLM — 약 10분

코드 4줄로 첫 호출

hello_llm.py

```
from openai import OpenAI
client = OpenAI()  # OPENAI_API_KEY 는 환경변수에서 자동 인식

resp = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": "한 줄로 자기소개해줘"}],
)
print(resp.choices[0].message.content)
```

실행

```
conda activate py312_gpt
cd tutorial-genai/1.openai/1.intro
python 3.sdk_new.py
```

예상 출력 (예시)

```
저는 OpenAI 가 만든 인공지능 비서로,
다양한 질문에 답변하고 글쓰기·요약·
분석·코드 작성 같은 작업을 도와드립니다.
```

잘 안 될 때 — 디버그 체크리스트

에러 메시지	원인	해결
`ModuleNotFoundError: No module named 'openai'`	conda 환경 활성화 안 함	`conda activate py312_gpt`
`openai.AuthenticationError: 401`	API Key 누락 또는 만료	`.env` 확인, 플랫폼에서 키 재발급
`openai.RateLimitError: 429`	초당 호출 한도 초과	`time.sleep(1)` 추가, 결제수단 등록 확인
`openai.InsufficientQuotaError`	무료 크레딧 소진	결제수단 등록 + spend limit 설정
`ssl.SSLError`	회사망/프록시 차단	핫스팟 사용 또는 IT 부서 문의

가장 흔한 시나리오 (90%)

```
# 1. 어디서 실행 중인지 확인
which python              # → py312_gpt 가 아니면 활성화 누락

# 2. .env 가 같은 디렉토리에 있는지
ls -la .env

# 3. .env 에 키가 실제로 있는지 (값은 마스킹)
grep "^OPENAI" .env
```

오늘 정리 7가지

1. ✅ Python **3.12** + **conda** 환경 `py312_gpt` 생성
2. ✅ 핵심 의존성 (`openai`, `langchain`, `chromadb` 등) 설치
3. ✅ 소스 코드 클론 — `tutorial-genai/`
4. ✅ `.env` 에 OPENAI_API_KEY 저장 + `.gitignore` 등록
5. ✅ 결제 한도 (Soft $5, Hard $20) 설정
6. ✅ VS Code 인터프리터 선택 → `py312_gpt`
7. ✅ 첫 호출 — `Hello, LLM` 응답 확인

시작 전 — 점검 4가지

- [ ] **conda activate py312_gpt** 가 정상 동작
- [ ] **OPENAI_API_KEY** 가 `.env` 에 저장됨, Hard limit 설정됨
- [ ] `python 3.sdk_new.py` 실행해 출력 확인
- [ ] (옵션) **ANTHROPIC_API_KEY** 도 같이 발급 (Agent 실습용)

🎯 최종 점검

```
cd tutorial-genai
conda activate py312_gpt
python -c "
from openai import OpenAI
r = OpenAI().chat.completions.create(
    model='gpt-4o-mini',
    messages=[{'role':'user','content':'준비 완료?'}],
    max_tokens=20
)
print('✅', r.choices[0].message.content)
"
```

> 위 한 줄이 응답 출력되면 **개발환경 준비 완료**.
> 안 되면 슬라이드 18(트러블슈팅) 참고 또는 강사에게 화면 공유 요청.