

`/* elice */` Enterprise

# 코드와 원리

RAG 파이프라인의 코드를 해부합니다

10:30 ~ 11:20

# 앞에서 확인한 3가지

---

문서에 질문하면 정확한 답변이 나왔습니다

답변과 함께 출처(파일명 + 페이지)가 표시됐습니다

프롬프트 규칙을 바꾸니 답변 품질이 달라졌습니다

이제 "어떻게" 동작하는지 코드로 봅니다.

# 5단계 - 이번에는 코드로 봅시다

앞에서 질문했을 때, 이 5가지가 순서대로 일어났습니다

Step 1	Step 2	Step 3	Step 4	Step 5
<b>질의 임베딩</b>  "용량이 얼마야?" → 벡터로 변환	<b>벡터DB 검색</b>  비슷한 문서 조각 3개 찾기	<b>프롬프트 조립</b>  "이 문서를 참고해서 답변하세요"  ★ 오늘 핵심 1	<b>LLM 답변</b>  Gemini가 문서 기반 답변 생성	<b>답변 + 출처</b>  답변 + 파일명 + 페이지 반환  ★ 오늘 핵심 2

학습한 Step 1, 2 에 이어서 오늘 Step 3, 4, 5 를 진행합니다.

# Step 3 - 프롬프트 조립

## 3단계 상세

### 벡터DB에서 검색된 문서 조각 3개

#### 조각 1

"RF9000의 총 용량은 868L이며  
냉장 524L, 냉동 344L..."

#### 조각 2

"Triple Cooling System으로  
독립 냉각을 지원하며..."

#### 조각 3

"치수는 912×1825×920mm이고  
중량은 135kg..."

+

### 사용자 질문

"RF9000의 총 용량은 얼마인가요?"



### 조립된 최종 프롬프트

다음 문서를 참고하여 답하세요.

**{context}** ← 문서 3개가 여기에

질문: **{question}** ← 질문이 여기에

# Step 4 - LLM 답변

## 조립된 프롬프트

문서 내용:  
RF9000의 총 용량은  
868L이며 냉장 524L,  
냉동 344L...

질문:  
RF9000의 총 용량은?



## 생성된 답변

RF9000의 총 용량은  
868L입니다.

냉장실 524L,  
냉동실 344L로  
구성되어 있습니다.

**핵심:** LLM이 "자기가 아는 것"이 아니라 "문서에서 찾은 내용"을 근거로 답변합니다

# Step 5 - 답변 + 출처

## result 객체 구조

`result['result']`

→ "RF9000의 총 용량`은 868L입니다..."

`result['source_documents']`

→ 검색에 사용된 문서 조각들

각 조각에 metadata 포함:

`source: "제품사양서.pdf"`

`page: 1`

## 이것이 중요한 이유

답변의 근거를 확인할 수 있다

원본 문서를 바로 찾아갈 수 있다

잘못된 답변을 검증할 수 있다

`return_source_documents=True`  
이 한 줄이 출처 반환을 겁니다

# 핵심 코드 5단계

## 핵심 슬라이드

```
# RAG 5단계를 직접 수행하는 함수
def qa_invoke(query):
```

```
    docs = vectorstore.similarity_search(query, k=3)
    context = "\n\n".join(doc.page_content for doc in docs)
    prompt = f"문서 내용:\n{context}\n질문: {query}\n답변:"
    answer = gemini_client.models.generate_content(
        model=GEMINI_MODEL, contents=prompt).text
    return {"result": answer, "source_documents": docs}
result = qa_invoke("RF9000의 총 용량은?")
```

← Step 1~2: 검색

← Step 3: 조립

← Step 4: LLM

← Step 5: 출처

# 코드 해설 : LLM 호출

## Step 4

```
gemini_client.models.generate_content(model=GEMINI_MODEL, contents=prompt)
```

**model = "gemini-2.0-flash-lite"**

어떤 LLM을 쓸지 선택 - 비용과 성능의 균형

**contents = prompt**

프롬프트 텍스트를 Gemini에 전달

## 쉽게 말하면

"어떤 전문가에게 맡길까?"

Gemini : 빠르고 저렴한 전문가  
프롬프트 규칙으로 정확도를 제어합니다



# 코드 해설 : retriever 파라미터

## Step 2

```
retriever=vectorstore.as_retriever(search_kwargs={"k": 3})
```

### vectorstore.as\_retriever()

Day2에서 만든 ChromaDB를 검색기로 변환.

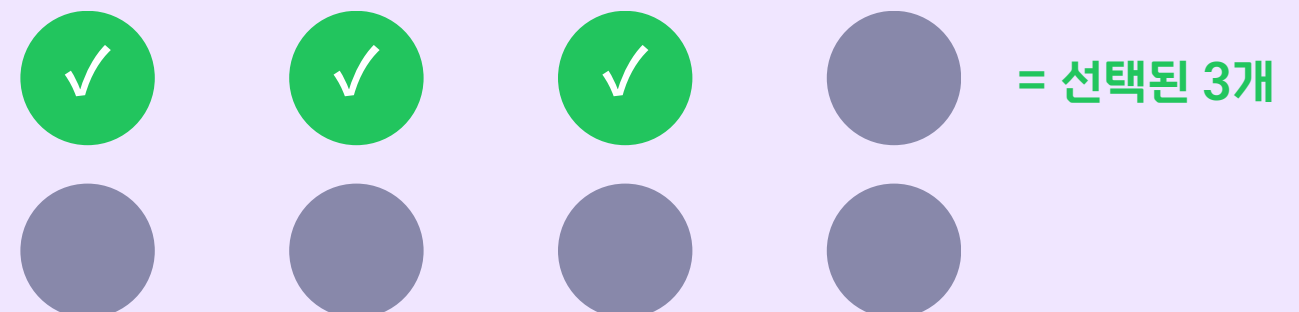
"벡터DB에서 비슷한 문서를 찾아줘"

### search\_kwargs={"k": 3}

상위 3개 문서 조각을 가져옴.  
우리 문서는 7페이지라 3개면 충분.

💡 k=3이면?

벡터DB에 15~20개 조각이 있는데  
그 중 질문과 가장 비슷한 3개만 선택



# 코드 해설 : chain\_type

## Step 3

```
context = "\n\n".join(...)
```

`stuff` = "전부 넣기"

검색된 문서 조각 3개를 프롬프트에 전부 넣는 방식  
우리 문서는 7페이지(사양서 4 + 시험성적서 3)라 전부 넣어도 충분합니다.  
이것만 기억하세요. 나머지 방식은 참고자료에 있습니다.

## 참고: 3가지 방식

`stuff` 전부 넣기

★ 이것만

`map_reduce` 나눠서 요약

`refine` 하나씩 정제

"3가지가 있는데, 우리 문서는 짧으니 `stuff`면 충분합니다. 이것만 기억하세요."

# 코드 해설 : 출처 반환 스위치

## Step 5 담당

```
return_source_documents=True
```

### True (켜기) ✓

답변 + 출처 문서 반환  
→ 파일명, 페이지 번호 확인 가능  
→ 제조업 보고서에 활용 가능

### False (끄기) ✗

답변만 반환  
→ 근거를 확인할 수 없음  
→ "이 숫자 어디서 나왔어?" 답 못함

제조업에서는 항상 True. 근거 없는 답변은 쓸 수 없습니다.

# 커스텀 프롬프트: 왜 필요한가?

블록1에서 기본과 커스텀의 차이를 봤습니다. 왜 달랐을까요?

## 기본 프롬프트 (LangChain 기본값)

Q: "RF9000 소비전력은?"

A: "RF9000의 소비전력은 36W입니다. 이 냉장고는 Triple Cooling System을 채택하고 있으며 에너지효율 1등급..."

장황하게 답변

## 커스텀 프롬프트 (규칙 적용)

Q: "RF9000 소비전력은?"

A: "RF9000의 소비전력은 36W입니다.  
(제품사양서 p.2)"

간결 + 단위 포함

차이를 만든 건 프롬프트에 넣은 "규칙" 3줄입니다.

# PromptTemplate — RAG 버전

2주차에 배운 PromptTemplate과 동일합니다. 변수가 다를 뿐.

```
custom_prompt = PromptTemplate(
    template="""
다음 문서 내용을 참고하여 질문에 답하세요.

문서 내용:
{context}

질문: {question}

답변 규칙:
"""
    input_variables=["context", "question"]
)
```

## {context}

검색된 문서 조각이  
자동으로 들어가는 자리

## {question}

사용자 질문이  
자동으로 들어가는 자리

2주차의 {input}이 RAG에서는 {context}와 {question}으로 나뉩니다

# 프롬프트 규칙 3가지

이 규칙이 답변의 품질을 결정합니다

1

**문서에 있는 정보만 사용하세요**

LLM이 자기 지식으로 답하는 걸 막습니다. → 할루시네이션 방지의 첫 번째 방어선

2

**문서에 없으면 "해당 정보를 찾을 수 없습니다"**

모르면 모른다고 말하게 합니다. → 거짓 답변보다 "모른다"가 안전합니다

3

**수치를 인용할 때는 단위를 반드시 포함하세요**

"868"이 아니라 "868L"로 답하게 합니다. → 제조업에서 단위 없는 숫자는 의미가 없습니다

오후 실습에서 이 규칙을 직접 코드에 넣어봅니다

## BRAINSTORMING

# 우리 업무에 맞는 규칙을 만든다면?

조별로 1개씩 → 오후 실습에서 직접 코드에 넣어봅니다!

### 예 시

- › "안전 관련 수치는 반드시 시험성적서 출처를 명시하세요"
- › "에러코드 설명 시 조치방법도 함께 답하세요"
- › "한국어로만 답변하세요"
- › "표 형식으로 정리해서 답하세요"

# 핵심 정리

---

- 1 RAG 5단계 코드 확인
- 2 코드의 역할 이해
- 3 커스텀 프롬프트로 인한 답변 품질 변화 확인
- 4 규칙 아이디어 생성 (오후 실습 예정)