4. 랭체인 기초

In [1]:
```
import os
from dotenv import load_dotenv
load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
```

4-1 랭체인 개요

In [2]:
```
!pip install langchain openai langchain-openai
```

```
Requirement already satisfied: langchain in c:\programdata\anaconda3\lib\site-packages (1.2.3)
Requirement already satisfied: openai in c:\programdata\anaconda3\lib\site-packages (2.15.0)
Requirement already satisfied: langchain-openai in c:\programdata\anaconda3\lib\site-packages (1.1.7)
Requirement already satisfied: langchain-core<2.0.0,>=1.2.1 in c:\programdata\anaconda3\lib\site-packages (from langchain) (1.
2.7)
Requirement already satisfied: langgraph<1.1.0,>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from langchain) (1.0.6)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in c:\programdata\anaconda3\lib\site-packages (from langchain) (2.10.3)
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-core<2.0.
0,>=1.2.1->langchain) (1.33)
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in c:\programdata\anaconda3\lib\site-packages (from langchain-core<2.0.
0,>=1.2.1->langchain) (0.6.2)
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-core<2.
0.0,>=1.2.1->langchain) (24.2)
Requirement already satisfied: pyyaml<7.0.0,>=5.3.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-core<2.0.0,>=
1.2.1->langchain) (6.0.2)
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-co
re<2.0.0,>=1.2.1->langchain) (9.0.0)
Requirement already satisfied: typing-extensions<5.0.0,>=4.7.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-co
re<2.0.0,>=1.2.1->langchain) (4.15.0)
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-core<2.0.
0,>=1.2.1->langchain) (0.13.0)
Requirement already satisfied: jsonpointer>=1.9 in c:\programdata\anaconda3\lib\site-packages (from jsonpatch<2.0.0,>=1.33.0->l
angchain-core<2.0.0,>=1.2.1->langchain) (2.1)
Requirement already satisfied: langgraph-checkpoint<5.0.0,>=2.1.0 in c:\programdata\anaconda3\lib\site-packages (from langgraph
<1.1.0,>=1.0.2->langchain) (4.0.0)
Requirement already satisfied: langgraph-prebuilt<1.1.0,>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from langgraph<
1.1.0,>=1.0.2->langchain) (1.0.6)
Requirement already satisfied: langgraph-sdk<0.4.0,>=0.3.0 in c:\programdata\anaconda3\lib\site-packages (from langgraph<1.1.0,
>=1.0.2->langchain) (0.3.3)
Requirement already satisfied: xxhash>=3.5.0 in c:\programdata\anaconda3\lib\site-packages (from langgraph<1.1.0,>=1.0.2->langc
hain) (3.6.0)
Requirement already satisfied: ormsgpack>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from langgraph-checkpoint<5.0.
0,>=2.1.0->langgraph<1.1.0,>=1.0.2->langchain) (1.12.1)
Requirement already satisfied: httpx>=0.25.2 in c:\programdata\anaconda3\lib\site-packages (from langgraph-sdk<0.4.0,>=0.3.0->l
anggraph<1.1.0,>=1.0.2->langchain) (0.28.1)
Requirement already satisfied: orjson>=3.10.1 in c:\programdata\anaconda3\lib\site-packages (from langgraph-sdk<0.4.0,>=0.3.0->
langgraph<1.1.0,>=1.0.2->langchain) (3.11.5)
Requirement already satisfied: requests-toolbelt>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from langsmith<1.0.0,>=
0.3.45->langchain-core<2.0.0,>=1.2.1->langchain) (1.0.0)
Requirement already satisfied: requests>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from langsmith<1.0.0,>=0.3.45->la
ngchain-core<2.0.0,>=1.2.1->langchain) (2.32.5)
```

```
Requirement already satisfied: zstandard>=0.23.0 in c:\programdata\anaconda3\lib\site-packages (from langsmith<1.0.0,>=0.3.45->
langchain-core<2.0.0,>=1.2.1->langchain) (0.23.0)
Requirement already satisfied: anyio in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=
0.3.0->langgraph<1.1.0,>=1.0.2->langchain) (4.7.0)
Requirement already satisfied: certifi in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.2->langgraph-sdk<0.4.0,>
=0.3.0->langgraph<1.1.0,>=1.0.2->langchain) (2025.10.5)
Requirement already satisfied: httpcore==1.* in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.2->langgraph-sdk<
0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain) (1.0.9)
Requirement already satisfied: idna in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.2->langgraph-sdk<0.4.0,>=0.
3.0->langgraph<1.1.0,>=1.0.2->langchain) (3.7)
Requirement already satisfied: h11>=0.16 in c:\programdata\anaconda3\lib\site-packages (from httpcore==1.*->httpx>=0.25.2->lang
graph-sdk<0.4.0,>=0.3.0->langgraph<1.1.0,>=1.0.2->langchain) (0.16.0)
Requirement already satisfied: annotated-types>=0.6.0 in c:\programdata\anaconda3\lib\site-packages (from pydantic<3.0.0,>=2.7.
4->langchain) (0.6.0)
Requirement already satisfied: pydantic-core==2.27.1 in c:\programdata\anaconda3\lib\site-packages (from pydantic<3.0.0,>=2.7.4
->langchain) (2.27.1)
Requirement already satisfied: distro<2,>=1.7.0 in c:\programdata\anaconda3\lib\site-packages (from openai) (1.9.0)
Requirement already satisfied: jiter<1,>=0.10.0 in c:\programdata\anaconda3\lib\site-packages (from openai) (0.12.0)
Requirement already satisfied: sniffio in c:\programdata\anaconda3\lib\site-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in c:\programdata\anaconda3\lib\site-packages (from openai) (4.67.1)
Requirement already satisfied: tiktoken<1.0.0,>=0.7.0 in c:\programdata\anaconda3\lib\site-packages (from langchain-openai) (0.
12.0)
Requirement already satisfied: regex>=2022.1.18 in c:\programdata\anaconda3\lib\site-packages (from tiktoken<1.0.0,>=0.7.0->lan
gchain-openai) (2024.11.6)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->la
ngsmith<1.0.0,>=0.3.45->langchain-core<2.0.0,>=1.2.1->langchain) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.0.0->langsmit
h<1.0.0,>=0.3.45->langchain-core<2.0.0,>=1.2.1->langchain) (2.3.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm>4->openai) (0.4.6)
```

4-2 Language models

LLMs

In [3]:
```python
from langchain_openai import ChatOpenAI

llm = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)
```

```
result = llm.invoke("자기소개를 해주세요.")
print(result)
```

```
content='안녕하세요! 저는 OpenAI에서 개발한 AI 언어 모델, ChatGPT입니다. 다양한 주제에 대해 대화하고 질문에 답변하며 글쓰기, 번역, 학습 도움 등 여러 가지 작업을 도와드릴 수 있습니다. 궁금한 점이나 도움이 필요하시면 언제든지 말씀해 주세요!' additional_kwargs={'refusal': None} response_metadata={'token_usage': {'completion_tokens': 68, 'prompt_tokens': 14, 'total_tokens': 82, 'completion_tokens_details': {'accepted_prediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 0}}, 'model_provider': 'openai', 'model_name': 'gpt-4.1-mini-2025-04-14', 'system_fingerprint': 'fp_376a7ccef1', 'id': 'chatcmpl-CxljG2vtHioJPXqWDTFsRUARQJJI8', 'service_tier': 'default', 'finish_reason': 'stop', 'logprobs': None} id='lc_run--019bba85-5177-7fe0-8fb2-fc7c1b08aecd-0' tool_calls=[] invalid_tool_calls=[] usage_metadata={'input_tokens': 14, 'output_tokens': 68, 'total_tokens': 82, 'input_token_details': {'audio': 0, 'cache_read': 0}, 'output_token_details': {'audio': 0, 'reasoning': 0}}
```

Chat models

In [4]:
```
from langchain_openai import ChatOpenAI
from langchain_classic.schema import AIMessage, HumanMessage, SystemMessage

chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)

messages = [
    SystemMessage(content="You are a helpful assistant."),
    HumanMessage(content="안녕하세요! 저는 존이라고 합니다!"),
    AIMessage(content="안녕하세요, 존 씨! 어떻게 도와드릴까요?"),
    HumanMessage(content= "제 이름을 아세요?")
]

result = chat.invoke(messages)
print(result.content)
```

```
네, 존 씨라고 하셨어요! 무엇을 도와드릴까요?
```

Callback을 이용한 스트리밍

In [5]:
```
from langchain_classic.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
from langchain_openai import ChatOpenAI
from langchain_classic.schema import HumanMessage

chat = ChatOpenAI(
```

```
    model_name="gpt-4.1-mini",
    temperature=0,
    streaming=True,
    callbacks=[StreamingStdOutCallbackHandler()],
)

messages = [HumanMessage(content="자기소개를 해주세요")]
result = chat.invoke(messages)
```

안녕하세요! 저는 OpenAI에서 개발한 AI 언어 모델, ChatGPT입니다. 다양한 주제에 대해 대화하고 질문에 답변하며 글쓰기, 번역, 학습 도움 등 여러 가지 작업을 도와드릴 수 있습니다. 궁금한 점이나 도움이 필요하시면 언제든지 말씀해 주세요!

4-3 Prompts

Prompt templates

In [6]:
```
from langchain_classic.prompts import PromptTemplate

template = """
다음 요리의 레시피를 생각해 주세요.

요리: {dish}
"""

prompt = PromptTemplate(
   input_variables=["dish"],
   template=template,
)

result = prompt.format(dish="카레")
print(result)
```

다음 요리의 레시피를 생각해 주세요.

요리: 카레

ChatPromptTemplate

In [7]:
```python
from langchain_classic.prompts import (
    ChatPromptTemplate,
    PromptTemplate,
    SystemMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain_classic.schema import HumanMessage, SystemMessage

chat_prompt = ChatPromptTemplate.from_messages([
    SystemMessagePromptTemplate.from_template("당신은 {country} 요리 전문가입니다."),
    HumanMessagePromptTemplate.from_template("다음 요리의 레시피를 생각해 주세요.\n\n요리: {dish}")
])

messages = chat_prompt.format_prompt(country="영국", dish="고기감자조림").to_messages()

print(messages)
```

```
[SystemMessage(content='당신은 영국 요리 전문가입니다.', additional_kwargs={}, response_metadata={}), HumanMessage(content='다음 요리의 레시피를 생각해 주세요.\n\n요리: 고기감자조림', additional_kwargs={}, response_metadata={})]
```

In [8]:
```python
from langchain_openai import ChatOpenAI

chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)
result = chat.invoke(messages)
print(result.content)
```

고기감자조림은 부드러운 감자와 진한 고기 양념이 어우러진 한국식 가정 요리입니다. 영국식 감자 요리와는 다르지만, 풍부한 맛과 푸짐한 식감이 매력적입니다. 아래는 고기감자조림의 기본 레시피입니다.

재료
- 돼지고기 (목살 또는 앞다리살) 300g
- 감자 3~4개 (중간 크기)
- 양파 1개
- 당근 1/2개 (선택 사항)
- 대파 1대
- 다진 마늘 1큰술
- 간장 3큰술
- 설탕 1큰술
- 참기름 1작은술
- 후추 약간
- 물 1컵 (200ml)
- 식용유 1큰술

조리 방법
1. **재료 손질**
   - 돼지고기는 한 입 크기로 썰고, 감자는 껍질을 벗기고 큼직하게 깍둑썰기 합니다.
   - 양파는 채 썰고, 당근도 감자 크기와 비슷하게 썰어줍니다.
   - 대파는 어슷하게 썰어 준비합니다.

2. **고기 볶기**
   - 팬에 식용유를 두르고 중불에서 다진 마늘을 볶아 향을 낸 뒤, 돼지고기를 넣고 겉면이 익을 때까지 볶습니다.

3. **양념 넣기**
   - 간장, 설탕, 후추를 넣고 고기에 잘 배도록 볶아줍니다.

4. **감자와 야채 넣기**
   - 감자, 양파, 당근을 넣고 고기와 잘 섞이도록 볶아줍니다.

5. **조림**
   - 물 1컵을 붓고 뚜껑을 덮은 뒤 중약불에서 20~25분간 감자가 부드러워질 때까지 조립니다.
   - 중간중간 국물이 너무 졸아들면 물을 조금씩 추가해 주세요.

6. **마무리**
   - 감자가 익으면 대파와 참기름을 넣고 한 번 더 섞은 뒤 불을 끕니다.

7. **서빙**
   - 따뜻할 때 밥과 함께 내면 맛있습니다.

필요에 따라 고기 대신 소고기를 사용하거나, 매콤한 맛을 원하면 고춧가루를 약간 넣어도 좋습니다. 영국식 스튜와는 다르게 간장과 설탕으로 단짠단짠한 맛을 내는 점이 특징입니다. 도움이 필요하시면 언제든 말씀해 주세요!

4-4 Output Parsers

PydanticOutputParser

In [9]:
```python
from pydantic import BaseModel, Field

class Recipe(BaseModel):
   ingredients: list[str] = Field(description="ingredients of the dish")
   steps: list[str] = Field(description="steps to make the dish")
```

In [10]:
```python
from langchain_classic.output_parsers import PydanticOutputParser

parser = PydanticOutputParser(pydantic_object=Recipe)
```

In [11]:
```python
format_instructions = parser.get_format_instructions()

print(format_instructions)
```

```
The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "ite
ms": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The object {"properties": {"foo": ["bar", "ba
z"]}} is not well-formatted.

Here is the output schema:
```
{"properties": {"ingredients": {"description": "ingredients of the dish", "items": {"type": "string"}, "title": "Ingredients",
"type": "array"}, "steps": {"description": "steps to make the dish", "items": {"type": "string"}, "title": "Steps", "type": "ar
ray"}}, "required": ["ingredients", "steps"]}
```
```

In [12]:
```python
from langchain_classic.prompts import PromptTemplate

template = """다음 요리의 레시피를 한국어로 생각해 주세요.

{format_instructions}

요리: {dish}
"""

prompt = PromptTemplate(
   template=template,
   input_variables=["dish"],
   partial_variables={"format_instructions": format_instructions}
)
```

In [13]:
```python
formatted_prompt = prompt.format(dish="카레")

print(formatted_prompt)
```

```
다음 요리의 레시피를 한국어로 생각해 주세요.

The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "ite
ms": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The object {"properties": {"foo": ["bar", "ba
z"]}} is not well-formatted.

Here is the output schema:
```
{"properties": {"ingredients": {"description": "ingredients of the dish", "items": {"type": "string"}, "title": "Ingredients",
"type": "array"}, "steps": {"description": "steps to make the dish", "items": {"type": "string"}, "title": "Steps", "type": "ar
ray"}}, "required": ["ingredients", "steps"]}
```

요리: 카레
```

In [14]:
```python
from langchain_openai import ChatOpenAI
from langchain_classic.schema import HumanMessage
```

```
chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)
messages = [HumanMessage(content=formatted_prompt)]
output = chat.invoke(messages)

print(output.content)
```

```json
{
  "ingredients": [
    "닭고기 300g",
    "감자 2개",
    "당근 1개",
    "양파 1개",
    "카레 가루 3큰술",
    "식용유 2큰술",
    "물 500ml",
    "소금 약간",
    "후추 약간"
  ],
  "steps": [
    "닭고기는 한 입 크기로 썰고, 감자와 당근은 껍질을 벗겨 큼직하게 자른다.",
    "양파는 채 썬다.",
    "냄비에 식용유를 두르고 중불에서 양파를 투명해질 때까지 볶는다.",
    "닭고기를 넣고 겉면이 하얗게 익을 때까지 볶는다.",
    "감자와 당근을 넣고 함께 볶는다.",
    "물 500ml를 붓고 끓인다.",
    "끓기 시작하면 중약불로 줄이고 뚜껑을 덮어 15분간 끓인다.",
    "카레 가루를 넣고 잘 저어가며 5분간 더 끓인다.",
    "소금과 후추로 간을 맞춘다.",
    "밥과 함께 따뜻하게 서빙한다."
  ]
}
```

In [15]:
```
recipe = parser.parse(output.content)
print(type(recipe))
print(recipe)
```

```
<class '__main__.Recipe'>
ingredients=['닭고기 300g', '감자 2개', '당근 1개', '양파 1개', '카레 가루 3큰술', '식용유 2큰술', '물 500ml', '소금 약간', '후추 약간'] steps=['닭고기는 한 입 크기로 썰고, 감자와 당근은 껍질을 벗겨 큼직하게 자른다.', '양파는 채 썬다.', '냄비에 식용유를 두르고 중불에서 양파를 투명해질 때까지 볶는다.', '닭고기를 넣고 겉면이 하얗게 익을 때까지 볶는다.', '감자와 당근을 넣고 함께 볶는다.', '물 500ml를 붓고 끓인다.', '끓기 시작하면 중약불로 줄이고 뚜껑을 덮어 15분간 끓인다.', '카레 가루를 넣고 잘 저어가며 5분간 더 끓인다.', '소금과 후추로 간을 맞춘다.', '밥과 함께 따뜻하게 서빙한다.']
```

4-5 Chains

LLMChain—PromptTemplate, Language model, OutputParser를 연결

In [16]:

```python
from langchain_openai import ChatOpenAI
from langchain_classic.output_parsers import PydanticOutputParser
from langchain_classic.prompts import PromptTemplate
from pydantic import BaseModel, Field

class Recipe(BaseModel):
    ingredients: list[str] = Field(description="ingredients of the dish")
    steps: list[str] = Field(description="steps to make the dish")

output_parser = PydanticOutputParser(pydantic_object=Recipe)

template = """다음 요리의 레시피를 한국어로 생각해 주세요.

{format_instructions}

요리: {dish}
"""

prompt = PromptTemplate(
    template=template,
    input_variables=["dish"],
    partial_variables={"format_instructions": output_parser.get_format_instructions()}
)

chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)
```

In [17]:
```python
from langchain_classic.chains import LLMChain

chain = LLMChain(prompt=prompt, llm=chat, output_parser=output_parser)

recipe = chain.invoke("카레")

print(type(recipe))
print(recipe)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21632\2630578017.py:3: LangChainDeprecationWarning: The class `LLMChain` was deprecated in LangChain 0.1.17 and will be removed in 1.0. Use `RunnableSequence, e.g., `prompt | llm`` instead.
  chain = LLMChain(prompt=prompt, llm=chat, output_parser=output_parser)
<class 'dict'>
{'dish': '카레', 'text': Recipe(ingredients=['닭고기 300g', '감자 2개', '당근 1개', '양파 1개', '카레 가루 3큰술', '식용유 2큰술', '물 500ml', '소금 약간', '후추 약간'], steps=['닭고기는 한 입 크기로 썰고, 감자와 당근은 껍질을 벗겨 큼직하게 자른다.', '양파는 채 썬다.', '냄비에 식용유를 두르고 중불에서 양파를 볶아 투명해질 때까지 익힌다.', '닭고기를 넣고 겉면이 하얗게 변할 때까지 볶는다.', '감자와 당근을 넣고 함께 볶는다.', '물 500ml를 붓고 끓기 시작하면 중약불로 줄여 재료가 부드러워질 때까지 약 15분간 끓인다.', '카레 가루를 넣고 잘 저어가며 5분 정도 더 끓인다.', '소금과 후추로 간을 맞춘 후 불을 끄고 잠시 둔다.', '밥과 함께 접시에 담아 낸다.'])}
```

SimpleSequentialChain — Chain과 Chain을 연결

In [18]:
```python
chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)

cot_template = """다음 질문에 답하세요.

질문: {question}

단계별로 생각해 봅시다.
"""

cot_prompt = PromptTemplate(
    input_variables=["question"],
    template=cot_template,
)

cot_chain = LLMChain(llm=chat, prompt=cot_prompt)
```

In [19]:
```
summarize_template = """다음 문장을 결론만 간단히 요약하세요.

{input}
"""
summarize_prompt = PromptTemplate(
   input_variables=["input"],
   template=summarize_template,
)

summarize_chain = LLMChain(llm=chat, prompt=summarize_prompt)
```

In [20]:
```
from langchain_classic.chains import SimpleSequentialChain

cot_summarize_chain = SimpleSequentialChain(chains=[cot_chain, summarize_chain])

result = cot_summarize_chain.invoke(
   """저는 시장에 가서 사과 10개를 샀습니다.
   이웃에게 2개, 수리공에게 2개를 주었습니다.
   그런 다음에 사과 5개를 더 사서 1개를 먹었습니다.
   남은 개수는 몇 개인가요?"""
)
print(result["output"])
```

```
남은 사과는 10개입니다.
```

위 입력은 Chain-of-Thought Prompting | Prompt Engineering Guide에서 인용했습니다.

4-6 Memory

ConversationBufferMemory

In [21]:
```
from langchain_classic.chains import ConversationChain
from langchain_openai import ChatOpenAI
from langchain_classic.memory import ConversationBufferMemory

chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)
conversation = ConversationChain(
```

```
    llm=chat,
    memory=ConversationBufferMemory()
)

while True:
    user_message = input("You: ")

    if user_message == "끝":
        print("(대화 종료)")
        break

    ai_message = conversation.invoke(input=user_message)["response"]
    print(f"AI: {ai_message}")
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_21632\427066300.py:8: LangChainDeprecationWarning: Please see the migration guide a
t: https://python.langchain.com/docs/versions/migrating_memory/
  memory=ConversationBufferMemory()
C:\Users\User\AppData\Local\Temp\ipykernel_21632\427066300.py:6: LangChainDeprecationWarning: The class `ConversationChain` was
deprecated in LangChain 0.2.7 and will be removed in 1.0. Use `langchain_core.runnables.history.RunnableWithMessageHistory` ins
tead.
  conversation = ConversationChain(
AI: 안녕하세요, 존님! 만나서 반갑습니다. 저는 여러분과 대화하고 도움을 드리기 위해 여기 있는 AI입니다. 오늘은 어떤 이야기를 나누
고 싶으신가요? 혹시 관심 있는 주제나 궁금한 점이 있으신가요?
AI: 네, 존님이라고 하셨으니 존님이라는 이름을 알고 있습니다! 앞으로도 편하게 존님이라고 부를게요. 혹시 이름에 얽힌 이야기나 의미
가 있으신가요? 함께 이야기 나누면 좋을 것 같아요.
(대화 종료)
```

(칼럼) Chat models에서 Memory를 사용할 때 주의할 점

In [22]:

```
from langchain_openai import ChatOpenAI
from langchain_classic.schema import AIMessage, HumanMessage, SystemMessage

chat = ChatOpenAI(model_name="gpt-4.1-mini", temperature=0)

messages = [
    SystemMessage(content="You are a helpful assistant."),
    HumanMessage(content="안녕하세요. 저는 존이라고 합니다!"),
    AIMessage(content="안녕하세요, 존 님! 어떻게 도와드릴까요?"),
    HumanMessage(content= "제 이름을 아세요?")
]
```

```
result = chat.invoke(messages)
print(result.content)
```

```
네, 존 님이라고 하셨어요! 무엇을 도와드릴까요?
```

In [ ]: