

## CHAPTER 03. Runnalbe

### 1. 값을 전달해 주는 - RunnablePassthrough

Runnable이라는 실행 가능한 객체를 통해서 데이터를 받아서 그 받은 데이터를 그냥 전달해 줍니다.

예제 - RunnablePaththrough

```
from dotenv import load_dotenv
import os

load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')

from langchain_openai import ChatOpenAI
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnablePassthrough

llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
)

prompt = PromptTemplate.from_template("{num} 의 10배는?")

# RunnablePassthrough()로 넘겨 받아진 10이 전달되어 다음과 같이 된다. {"num": 10}
# {"num": 10} 딕셔너리가 prompt의 입력으로 들어간다.
prompt와 llm을 연결해서 chain을 만든다.
runnable_chain = {"num": RunnablePassthrough()} | prompt | llm

answer = runnable_chain.invoke(10)

print(answer.content)
```

실행결과

10의 10배는 100입니다.

RunnablePassthrough.assign() : 입력 값으로 들어온 값의 key/value 쌍과 새롭게 할당된 key/value 쌍을 합칩니다.

#### 예제 - RunnablePaththrough

```
from dotenv import load_dotenv
import os

load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')

from langchain_openai import ChatOpenAI
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnablePassthrough

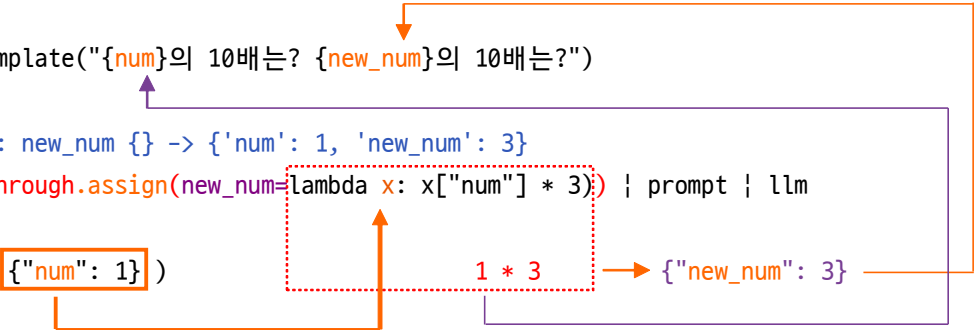
llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
)

prompt = PromptTemplate.from_template("{num}의 10배는? {new_num}의 10배는?")

# 입력 키: num, 할당(assign) 키: new_num {} -> {'num': 1, 'new_num': 3}
runnable_chain = (RunnablePassthrough.assign(new_num=lambda x: x["num"] * 3)) | prompt | llm

answer = runnable_chain.invoke({"num": 1})

print(answer.content)
```



#### 실행결과

1의 10배는 10입니다.

3의 10배는 30입니다.

```
content='1의 10배는 10입니다. \n3의 10배는 30입니다.'
response_metadata={
    'token_usage': {'completion_tokens': 21, 'prompt_tokens': 22, 'total_tokens': 43},
    'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_0f03d4f0ee',
    'finish_reason': 'stop', 'logprobs': None
}
id='run-5780c4fb-09b9-4fbe-89b8-edbbd1fbaae2-0'
usage_metadata={'input_tokens': 22, 'output_tokens': 21, 'total_tokens': 43}
```

## 2. 병렬로 Runnable을 실행하는 RunnableParallel

병렬로 Runnable 객체를 실행해준다.

예제 - 병렬로 Runnable 객체 실행

```
from dotenv import load_dotenv
import os
```

```
load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')
```

```
from langchain_openai import ChatOpenAI
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_core.runnables import RunnableParallel
```

```
llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
```

```
)
chain1 = (
    {"country": RunnablePassthrough()}
    | PromptTemplate.from_template("{country}의 수도는?")
    | llm
```

```
)
chain2 = (
    {"country": RunnablePassthrough()}
    | PromptTemplate.from_template("{country}의 면적은?")
    | llm
```

```
)
combined_chain = RunnableParallel(capital=chain1, area=chain2)
```

```
answer = combined_chain.invoke("대한민국")
```

```
print(answer)
```

## 실행결과

```
{'capital': AIMessage(content='대한민국의 수도는 서울입니다.', response_metadata={'token_usage':  
{'completion_tokens': 8, 'prompt_tokens': 13, 'total_tokens': 21}, 'model_name':  
'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_0f03d4f0ee', 'finish_reason': 'stop',  
'logprobs': None}, id='run-436a1bd6-e1c3-4406-a6f0-e7e4699814d0-0',  
usage_metadata={'input_tokens': 13, 'output_tokens': 8, 'total_tokens': 21}), 'area':  
AIMessage(content='대한민국의 면적은 약 100,210 평방킬로미터(km²)입니다. 이는 한반도의 남쪽 부분에  
해당하며, 북한과 함께 한반도를 구성하고 있습니다.', response_metadata={'token_usage':  
{'completion_tokens': 44, 'prompt_tokens': 14, 'total_tokens': 58}, 'model_name':  
'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_0f03d4f0ee', 'finish_reason': 'stop',  
'logprobs': None}, id='run-4c3ddcda-650e-42e5-bae6-08eee9ffaa5a-0',  
usage_metadata={'input_tokens': 14, 'output_tokens': 44, 'total_tokens': 58}}})
```

### 3. 함수를 실행하는 - RunnableLambda, itemgetter

RunnableLambda는 사용자가 만든 함수를 맵핑 해주는 역할을 하는 Runnable 객체입니다.

함수를 만들고 프롬프트 입력으로 들어가기 전에 함수를 실행시켜 그 결과 값을 프롬프트의 입력으로 넣을 수 있습니다.

- 1) 파이썬으로 작성한 함수를 RunnableLambda로 감싸서 먼저 호출해서 답을 얻습니다.
- 2) 함수를 실행해서 얻은 결과 값을 프롬프트에 있는 어떠한 변수의 입력값으로 넣어줄 수 있습니다.

#### 예제 - RunnableLambda

```
from dotenv import load_dotenv
import os
```

```
load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')
```

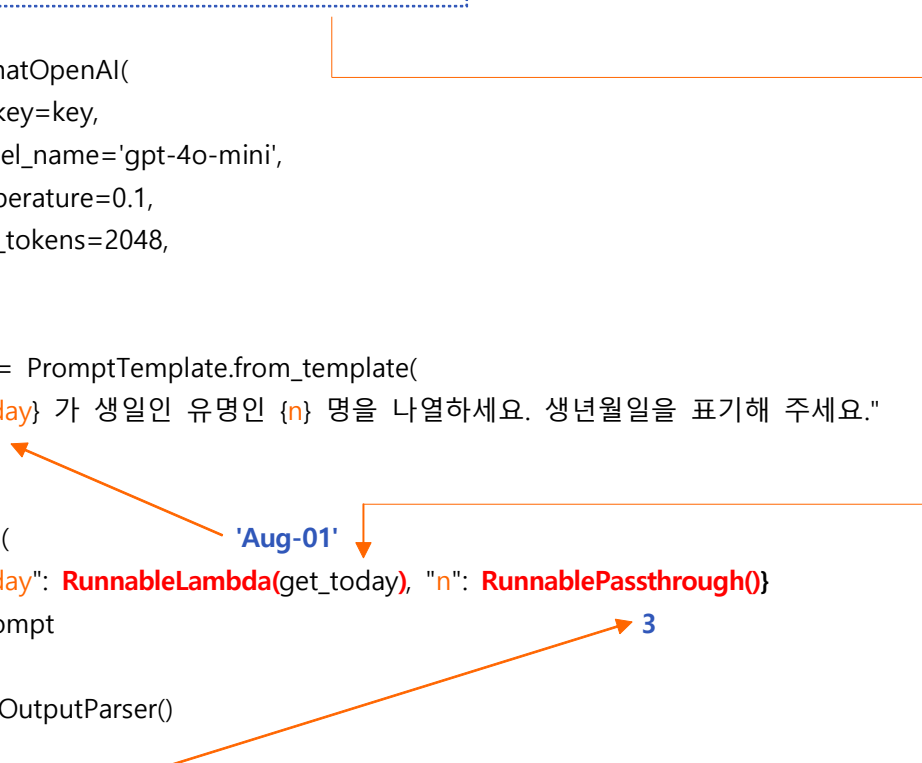
```
from langchain_openai import ChatOpenAI
from langchain_core.runnables import RunnablePassthrough
from langchain_core.runnables import RunnableLambda
from langchain_core.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser
from datetime import datetime
```

```
def get_today(a):
    return datetime.today().strftime("%b-%d")
```

```
llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
)
```

```
prompt = PromptTemplate.from_template(
    "{today} 가 생일인 유명인 {n} 명을 나열하세요. 생년월일을 표기해 주세요."
)
```

```
chain = (
    {"today": RunnableLambda(get_today), "n": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)
print(chain.invoke(3))
```



## 실행결과

---

다음은 8월 1일에 생일인 유명인 3명입니다:

1. \*\*버지니아 울프 (Virginia Woolf)\*\* - 1882년 1월 25일
2. \*\*프레드릭 더글라스 (Frederick Douglass)\*\* - 1818년 2월 14일
3. \*\*미셸 오바마 (Michelle Obama)\*\* - 1964년 1월 17일

이 중에서 생일이 8월 1일인 유명인은 없습니다. 8월 1일에 태어난 유명인으로는 다음과 같은 인물이 있습니다:

1. \*\*루이스 암스트롱 (Louis Armstrong)\*\* - 1901년 8월 4일
2. \*\*미래의 유명인 (Future Celebrity)\*\* - 2000년 8월 1일

이 외에도 8월 1일에 태어난 유명인으로는 \*\*제리 하퍼 (Jerry Heller)\*\* , \*\*스티븐 스필버그 (Steven Spielberg)\*\* 등이 있습니다.

---

## 예제 -

```
from dotenv import load_dotenv
import os

load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')

from langchain_openai import ChatOpenAI
from langchain_core.runnables import RunnablePassthrough
from langchain_core.runnables import RunnableLambda
from operator import itemgetter
from langchain_core.prompts import PromptTemplate
from langchain_core.output_parsers import StrOutputParser
from datetime import datetime

def get_today(a):
    print(f'입력받은 값: {a}')
    return datetime.today().strftime("%b-%d")

llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
)

prompt = PromptTemplate.from_template(
    "{today} 가 생일인 유명인 {n} 명을 나열하세요. 생년월일을 표기해 주세요."
)

chain = (
    {"today": RunnableLambda(get_today), "n": itemgetter("n")}
    | prompt
    | llm
    | StrOutputParser()
)

# 사용자의 입력이 딕셔너리로 전달되는 경우
print( chain.invoke( {"n": 3} ) )
```



## 실행결과

입력받은 값: {'n': 3}

다음은 8월 2일에 태어난 유명인 3명입니다:

1. \*\*제프리 제임스 (Jeffrey James)\*\* - 1980년 8월 2일
2. \*\*메간 폭스 (Megan Fox)\*\* - 1986년 8월 2일
3. \*\*올리비아 뉴턴-존 (Olivia Newton-John)\*\* - 1948년 8월 2일

이 외에도 8월 2일에 태어난 많은 유명인들이 있습니다.

예제 -

```
from dotenv import load_dotenv
import os
```

```
load_dotenv(verbose=True)
key = os.getenv('OPENAI_API_KEY')
```

```
from langchain_openai import ChatOpenAI
from langchain_core.runnables import RunnableLambda
from langchain_core.prompts import ChatPromptTemplate
from operator import itemgetter
```

```
llm = ChatOpenAI(
    api_key=key,
    model_name='gpt-4o-mini',
    temperature=0.1,
    max_tokens=2048,
)
```

# 문장의 길이를 반환하는 함수입니다. 매개 변수가 1개

```
def length_function(text):
    return len(text) # len("hello") → 3
```

# 두 문장의 길이를 곱한 값을 반환하는 함수입니다. 매개 변수가 2개

```
def _multiple_length_function(text1, text2):
    return len(text1) * len(text2) # len("hello") * len("world") → 25
```

# 두 문장의 길이를 곱한 값을 반환하는 함수입니다. 매개 변수 1개

```
def multiple_length_function(_dict): # {"text1": "hello", "text2": "world"}
    return _multiple_length_function(_dict["text1"], _dict["text2"])
```

"hello" "world"

```
prompt = ChatPromptTemplate.from_template("{a} + {b}는 무엇인가요?")
# → 3 + 25는 무엇인가요?
```

```
chain = (
    {
        "a": itemgetter("word1") | RunnableLambda(length_function), # 결과 "a": 3
        "b": {"text1": itemgetter("word1"), "text2": itemgetter("word2")} # 결과 "b": 25
        | RunnableLambda(multiple_length_function), [{"text1": "hello", "text2": "world"}]
    }
    | prompt
    | llm
)
```

```
answer = chain.invoke({ "word1": "hello", "word2": "world" })
```

```
print(answer.content)
```



## 실행결과

---

'5 + 25는 30입니다.'

---

- 출처 : 랭체인LangChain 노트 (<https://wikidocs.net/233343>)