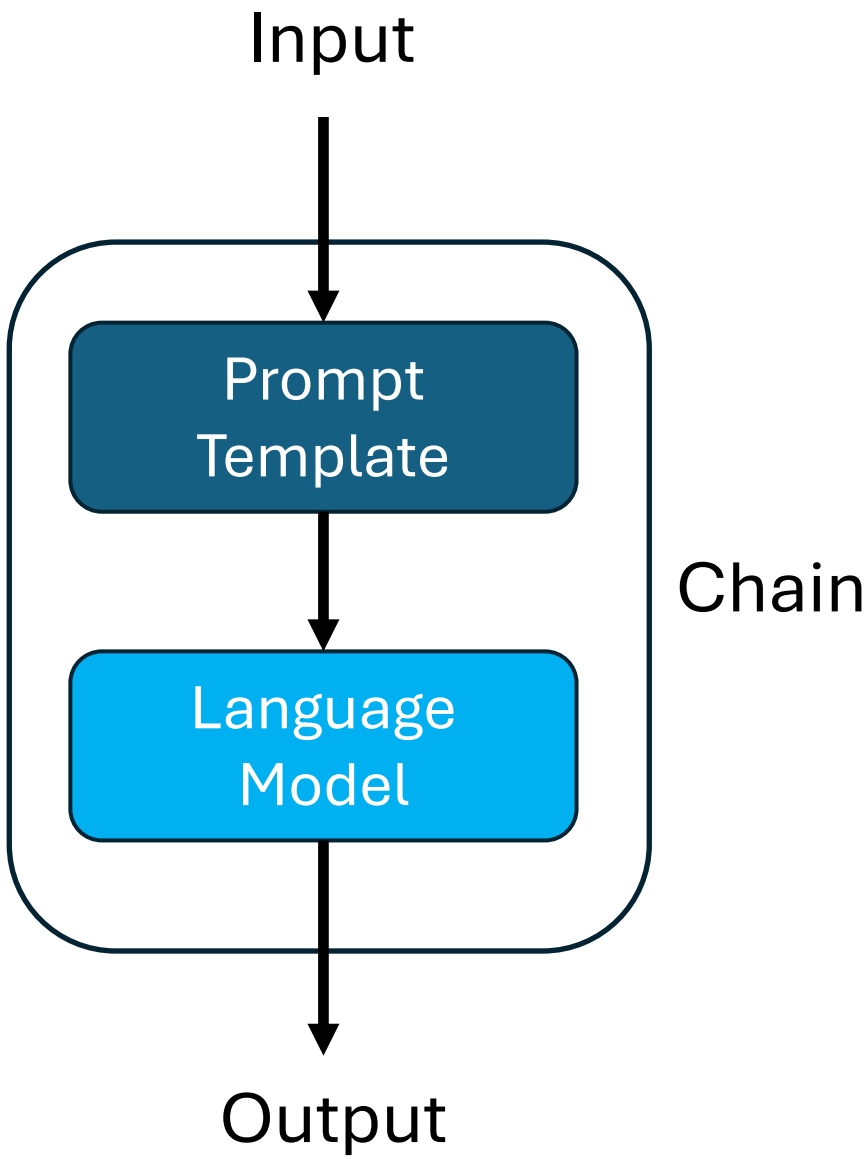
2025

Lesson 10

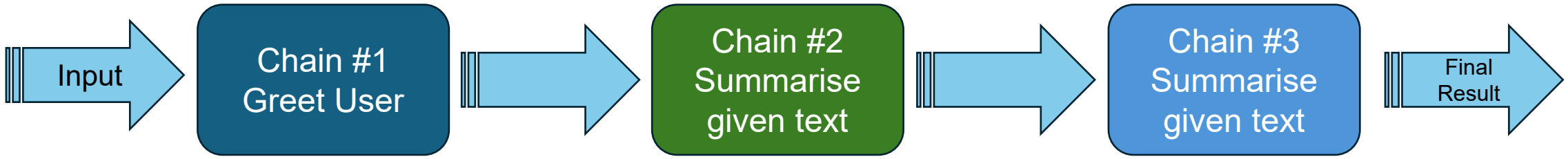
LangChain: Chains & Runnables

Chain

- Python class provided by LangChain.
- Use Chains to make reusable text-generation pipelines.
- Chains can be connected to make a more complex pipeline.
- A chain wraps up a PromptTemplate and an LLM.

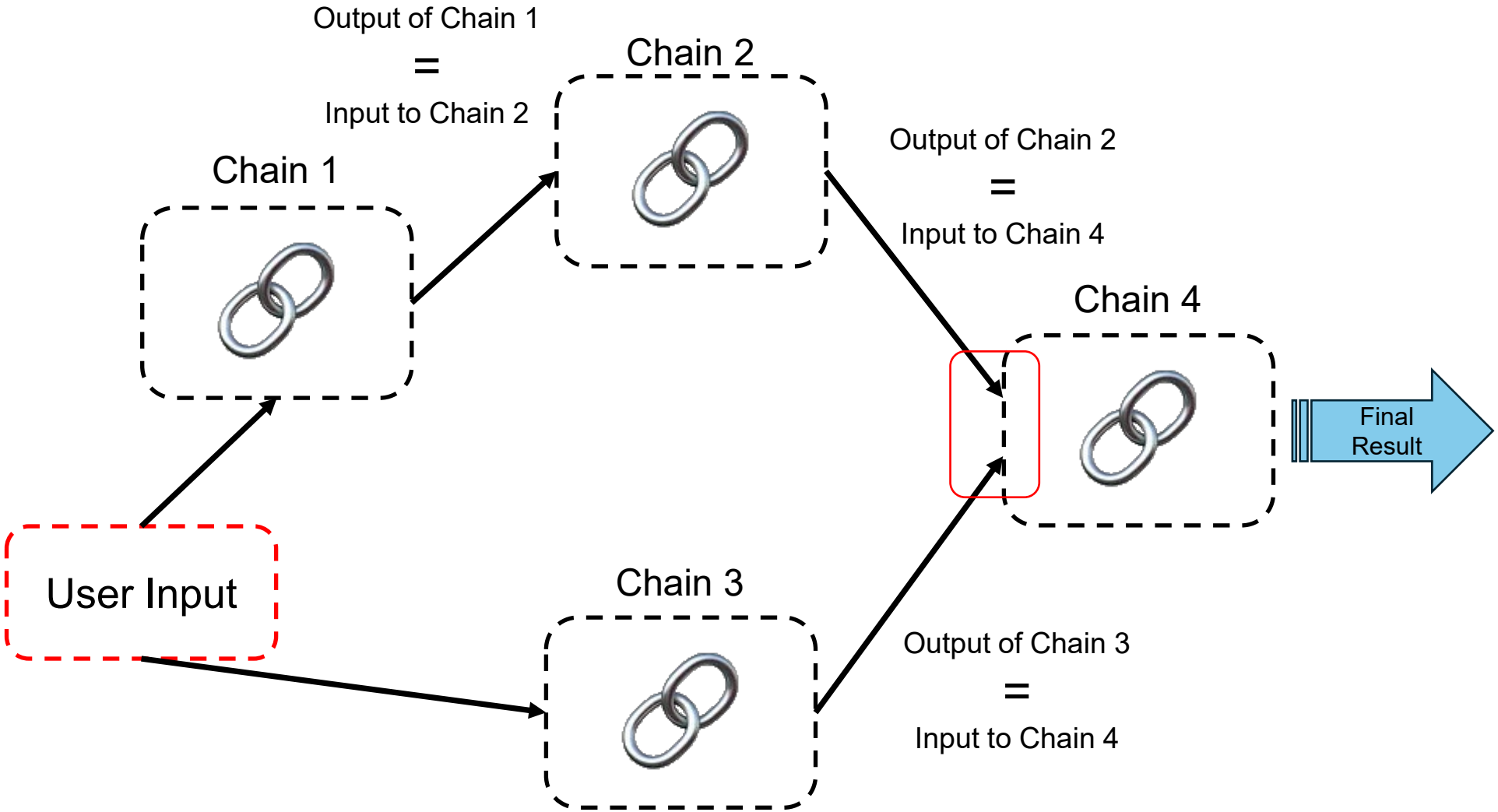


Simple Sequential Chain

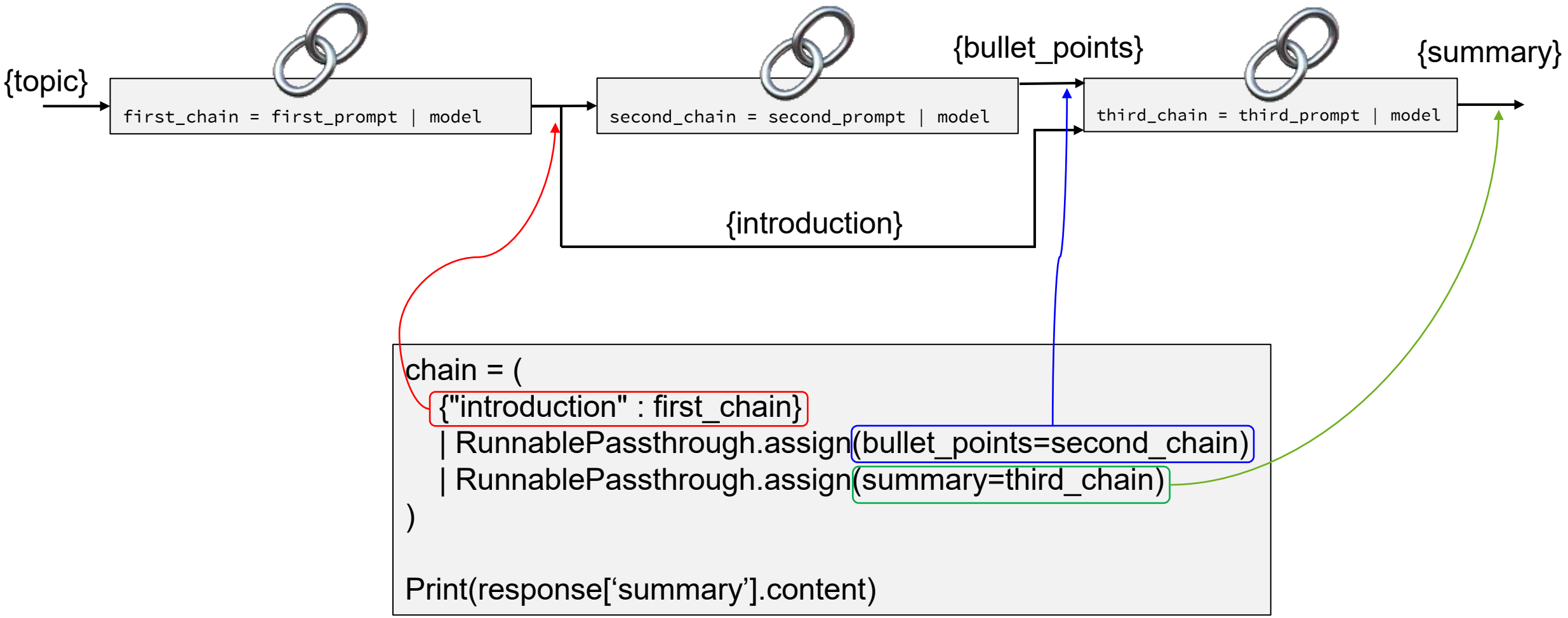


- The output of Chain #1 becomes the input of Chain #2 and the output of Chain #2 becomes the input of Chain #3.
- The output of the first step can flow into the input of the second (if needed), or the steps work independently but sequentially.
- Chains allows us to create complex LLM applications by piecing together components to create a single coherent application.

Sequential Chain

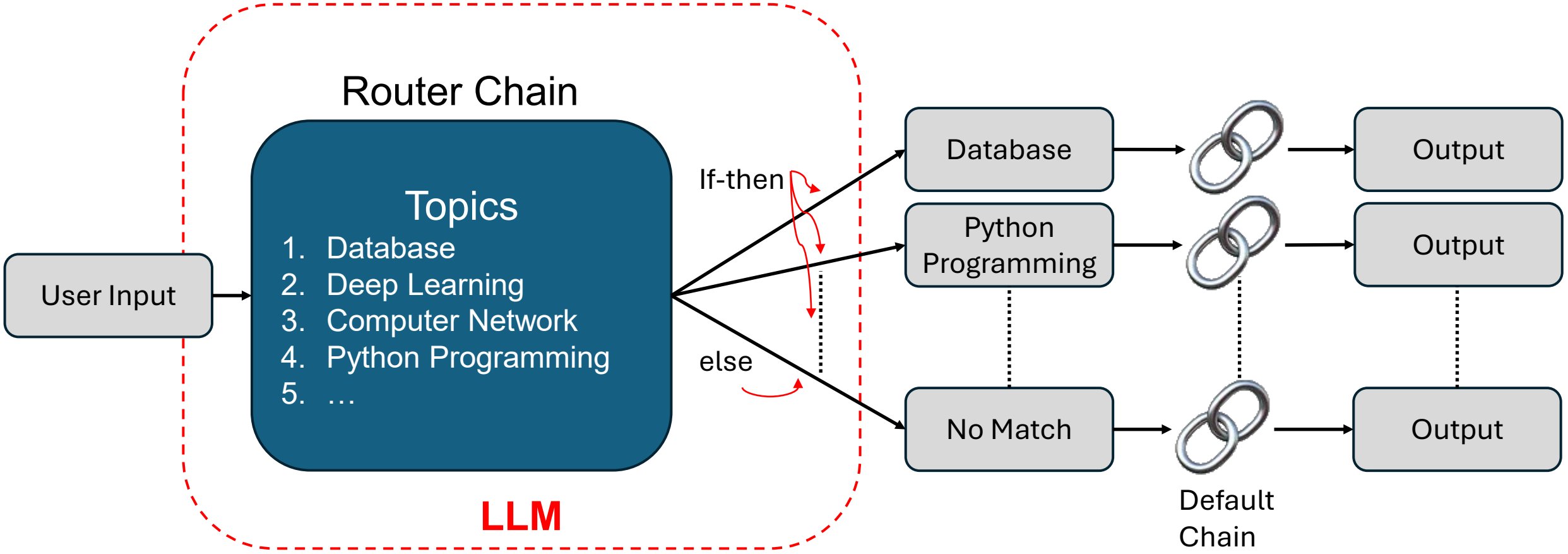


Sequential Chain



Router Chain

- The router chain knows how to route to different chains, depending on the input it receives.



LangChain Interface: Runnables

- Runnable protocol enable the easy creation of custom chains.
- Many `LangChain` components implement the Runnable protocol (chat models, LLMs, output parser, retrievers, prompt templates etc.).
- The components can be invoked, batched, streamed, transformed and composed.
- It is a way of abstracting the idea of taking some input, processing it and producing an output.
- Runnables also have corresponding async methods that should be used with asyncio `await` syntax for concurrency.

Why Runnable?

- Standarisation: It provides a common interface for various components and making it easy to work with different parts in a consistent manner.
- Composability: It is easier to combined and chained together to create complex workflow.
- Flexibility: It allows various types of inputs and outputs.
- Asynchronous support: It can be executed synchronously or asynchronously.

Runnables

RunnablePassthrough

- Passthrough inputs unchanged or with additional keys.

RunnableLambda

- Converts a python callable (function) into a Runnable.

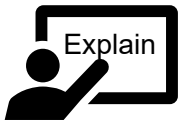
RunnableParallel

- Runs a mapping of Runnables in parallel and returns a mapping of their outputs.

RunnableSequence

- Sequence of Runnables where the output of each is the input of the next.

RunnableParallel




```
chain1 = ChatPromptTemplate.from_template("tell me a joke about {topic}") | model

chain2 = ChatPromptTemplate.from_template("write a short (2 line) poem about {topic}") | model

parallel_chain = RunnableParallel(joke=chain1, short_poem=chain2)

response = parallel_chain.invoke({"topic" : "bears"})
```

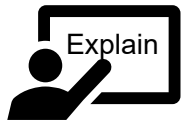
OUTPUT CUT OFF

```
{'joke': AIMessage(content='Why do bears have hairy coats?\n\nBecause the
ta={'token_usage': {'completion_tokens': 14, 'prompt_tokens': 13, 'total_
io_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_tokens': 0}, '
'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_6fc10e10eb', 'finish_
-0', usage_metadata={'input_tokens': 13, 'output_tokens': 14, 'total_toke
ails': {'audio': 0, 'reasoning': 0}}),
 'short_poem': AIMessage(content='In forests deep where shadows play,  \n
e}, response_metadata={'token_usage': {'completion_tokens': 18, 'prompt_t
on_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_predic
0}}, 'model_name': 'gpt-4o-mini-2024-07-18', 'system_fingerprint': 'fp_6f
d-b1ef-826be5ceae77-0', usage_metadata={'input_tokens': 17, 'output_token
0}, 'output_token_details': {'audio': 0, 'reasoning': 0}})}
```

chain 1 & chain 2 are run in ‘parallel’. Each chain takes in “topic” as an input.

RP

RunnablePassthough




```
tweet_chain = RunnableParallel(
    {
        "mood": RunnablePassthrough() | itemgetter("mood"),
        "tweet": tweet_generator
    }
) | tweet_fixer

tweet_chain.invoke(
    {
        "prompt": "Langchain releases LangGraph for building stateful, multi-action applications. https://langchain.com",
        "mood": "sarcastic"
    }
)
```

They are run in parallel.

Next run is “tweet_fixer”

'Oh great, just what we needed—another tool! 🙄 Langchain has dropped LangGraph, your *new best friend* for building stateful, multi-action apps. 🎉💻 Because who doesn’t want to add more complexity to their projects? Let’s all get graphing... or not. 🌐✨ #LangGraph #Langchain #DevLife #Innovation’

Output

Reference

- Runnable Interface
  https://python.langchain.com/docs/concepts/runnables/
- Cheat Sheet
  https://python.langchain.com/docs/how_to/lcel_cheatsheet/

Thank you!