



Multiple chains

Runnables can easily be used to string together multiple Chains

```
%pip install --upgrade --quiet langchain langchain-openai
```

```
from operator import itemgetter

from langchain.schema import StrOutputParser
from langchain_core.prompts import
ChatPromptTemplate
from langchain_openai import ChatOpenAI

prompt1 =
ChatPromptTemplate.from_template("what is the
city {person} is from?")
prompt2 = ChatPromptTemplate.from_template(
    "what country is the city {city} in?
respond in {language}"
)

model = ChatOpenAI()

chain1 = prompt1 | model | StrOutputParser()
```

```
chain2 = (  
    {"city": chain1, "language":  
itemgetter("language")}  
    | prompt2  
    | model  
    | StrOutputParser()  
)  
  
chain2.invoke({"person": "obama", "language":  
"spanish"})
```

'El país donde se encuentra la ciudad de Honolulu, donde nació Barack Obama, el 44° Presidente de los Estados Unidos, es Estados Unidos. Honolulu se encuentra en la isla de Oahu, en el estado de Hawái.'

```
from langchain_core.runnables import  
RunnablePassthrough  
  
prompt1 = ChatPromptTemplate.from_template(  
    "generate a {attribute} color. Return the  
name of the color and nothing else:"  
)  
prompt2 = ChatPromptTemplate.from_template(  
    "what is a fruit of color: {color}.  
Return the name of the fruit and nothing
```

```
else:"
)
prompt3 = ChatPromptTemplate.from_template(
    "what is a country with a flag that has
the color: {color}. Return the name of the
country and nothing else:"
)
prompt4 = ChatPromptTemplate.from_template(
    "What is the color of {fruit} and the
flag of {country}?"
)

model_parser = model | StrOutputParser()

color_generator = (
    {"attribute": RunnablePassthrough()} |
prompt1 | {"color": model_parser}
)
color_to_fruit = prompt2 | model_parser
color_to_country = prompt3 | model_parser
question_generator = (
    color_generator | {"fruit":
color_to_fruit, "country": color_to_country}
| prompt4
)
```

```
question_generator.invoke("warm")
```

```
ChatPromptValue(messages=[HumanMessage(content='What is the color of strawberry and the flag of China?', additional_kwargs={}, example=False)])
```

```
prompt = question_generator.invoke("warm")  
model.invoke(prompt)
```

```
AIMessage(content='The color of an apple is typically red or green. The flag of China is predominantly red with a large yellow star in the upper left corner and four smaller yellow stars surrounding it.', additional_kwargs={}, example=False)
```

Branching and Merging

You may want the output of one component to be processed by 2 or more other components. [RunnableParallels](#) let you split or fork the chain so multiple components can process the input in parallel. Later, other components can join or merge the results to synthesize a final response. This type of chain creates a computation graph that looks like the following:

```

      Input
      /  \
     /    \
  Branch1 Branch2
     \    /
      \  /
      Combine

```

```

planner = (

ChatPromptTemplate.from_template("Generate an
argument about: {input}")
    | ChatOpenAI()
    | StrOutputParser()
    | {"base_response":
RunnablePassthrough()})
)

arguments_for = (
    ChatPromptTemplate.from_template(
        "List the pros or positive aspects of
{base_response}"
    )
    | ChatOpenAI()
    | StrOutputParser()
)

arguments_against = (
    ChatPromptTemplate.from_template(
        "List the cons or negative aspects of

```

```
{base_response}"
    )
    | ChatOpenAI()
    | StrOutputParser()
)

final_responder = (
    ChatPromptTemplate.from_messages(
        [
            ("ai", "{original_response}"),
            ("human",
"Pros:\n{results_1}\n\nCons:\n{results_2}"),
            ("system", "Generate a final
response given the critique"),
        ]
    )
    | ChatOpenAI()
    | StrOutputParser()
)

chain = (
    planner
    | {
        "results_1": arguments_for,
        "results_2": arguments_against,
        "original_response":
itemgetter("base_response"),
    }
    | final_responder
)
```

```
chain.invoke({"input": "scrum"})
```

'While Scrum has its potential cons and challenges, many organizations have successfully embraced and implemented this project management framework to great effect. The cons mentioned above can be mitigated or overcome with proper training, support, and a commitment to continuous improvement. It is also important to note that not all cons may be applicable to every organization or project.

For example, while Scrum may be complex initially, with proper training and guidance, teams can quickly grasp the concepts and practices. The lack of predictability can be mitigated by implementing techniques such as velocity tracking and release planning. The limited documentation can be addressed by maintaining a balance between lightweight documentation and clear communication among team members. The dependency on team collaboration can be improved through effective communication channels and regular team-building activities.

Scrum can be scaled and adapted to larger projects by using frameworks like Scrum of Scrums or LeSS (Large Scale Scrum). Concerns about speed versus quality can be addressed by

incorporating quality assurance practices, such as continuous integration and automated testing, into the Scrum process. Scope creep can be managed by having a well-defined and prioritized product backlog, and a strong product owner can be developed through training and mentorship.

Resistance to change can be overcome by providing proper education and communication to stakeholders and involving them in the decision-making process. Ultimately, the cons of Scrum can be seen as opportunities for growth and improvement, and with the right mindset and support, they can be effectively managed.

In conclusion, while Scrum may have its challenges and potential cons, the benefits and advantages it offers in terms of collaboration, flexibility, adaptability, transparency, and customer satisfaction make it a widely adopted and successful project management framework. With proper implementation and continuous improvement, organizations can leverage Scrum to drive innovation, efficiency, and project success.'