

# The LangChain ecosystem

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



**James Chapman**  
Curriculum Manager, DataCamp

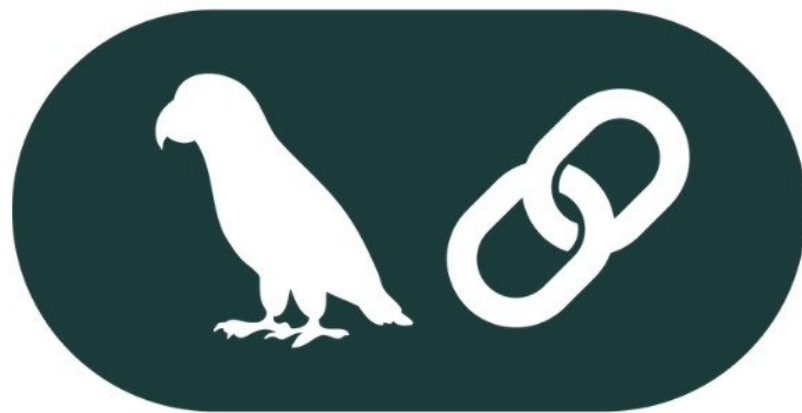
# Meet your instructors...



- Jonathan Bennion, **AI Engineer**
- ML & AI at Facebook, Google, Amazon, Disney, EA
- Created *Logical Fallacy chain* in LangChain
- Contributor to **DeepEval**

# Build LLM Apps with LangChain





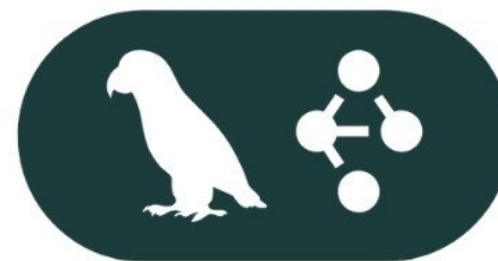
# LangChain



## LangSmith

=

## Deploying Apps



## LangGraph

=

## AI Agents

# LangChain integrations



**ANTHROPIC**

 **OpenAI**

**aws**

 **snowflake**



 **databricks**  **neo4j**

  **MISTRAL AI\_**

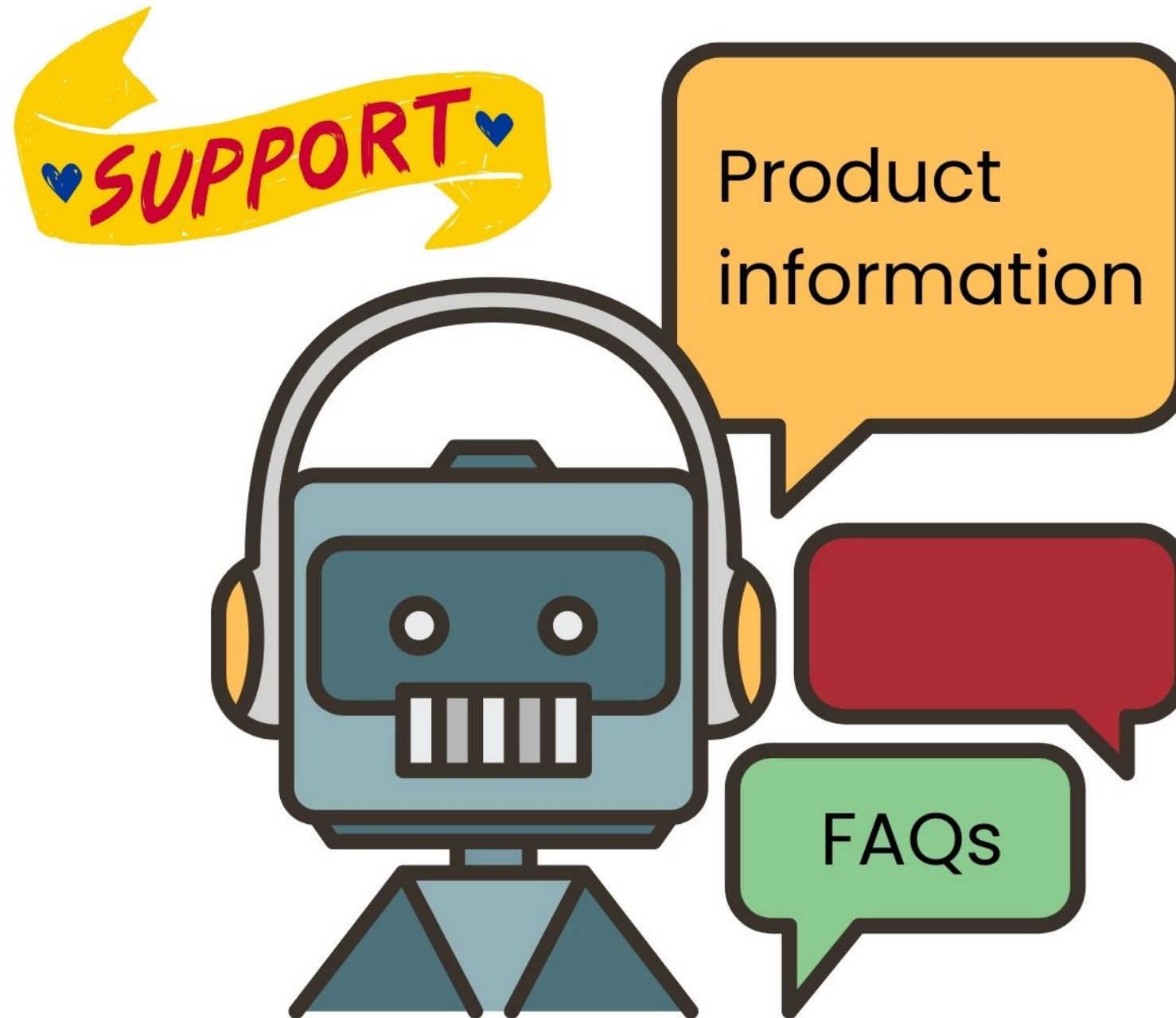
**Hugging Face**

 **cohere**

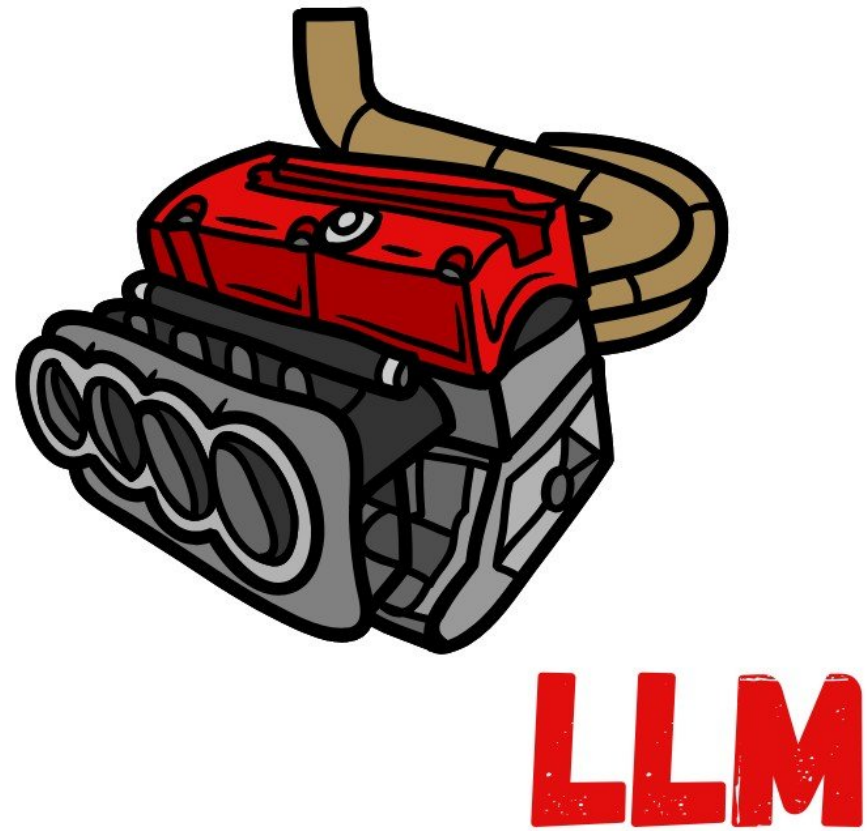
<sup>1</sup> <https://python.langchain.com/docs/integrations/providers/>



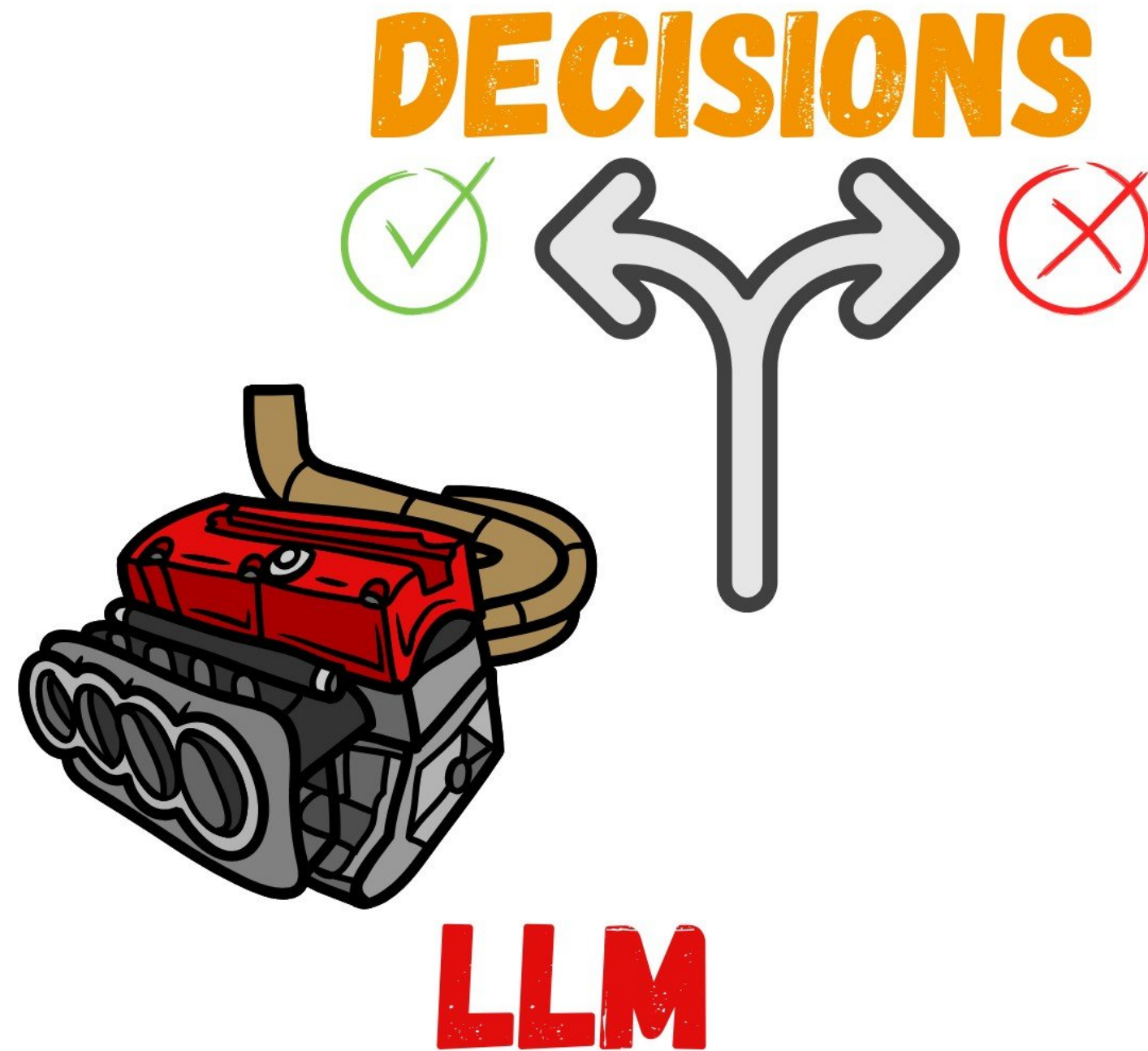
# Building LLM apps the LangChain way...



# Building LLM apps the LangChain way...

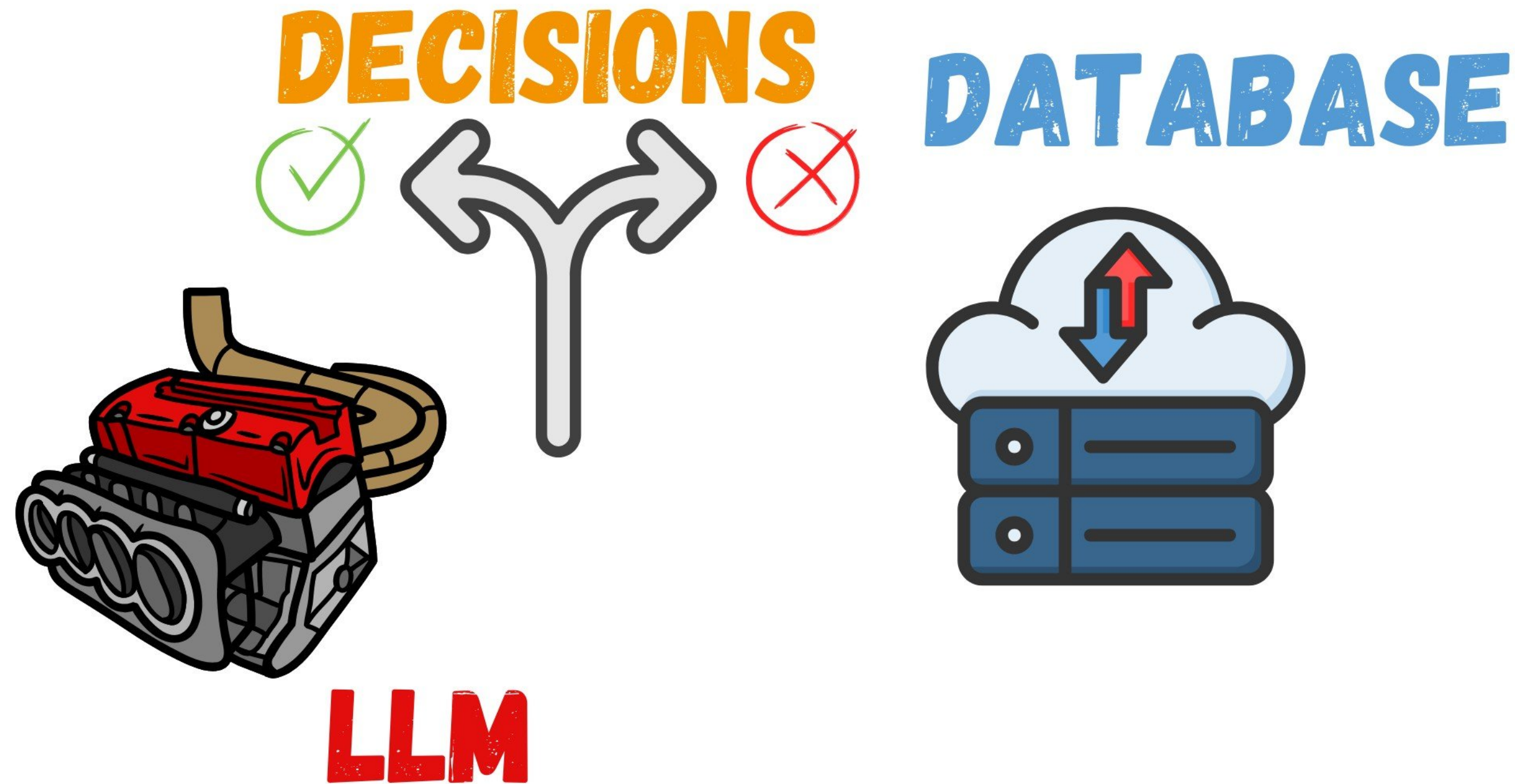


# Building LLM apps the LangChain way...

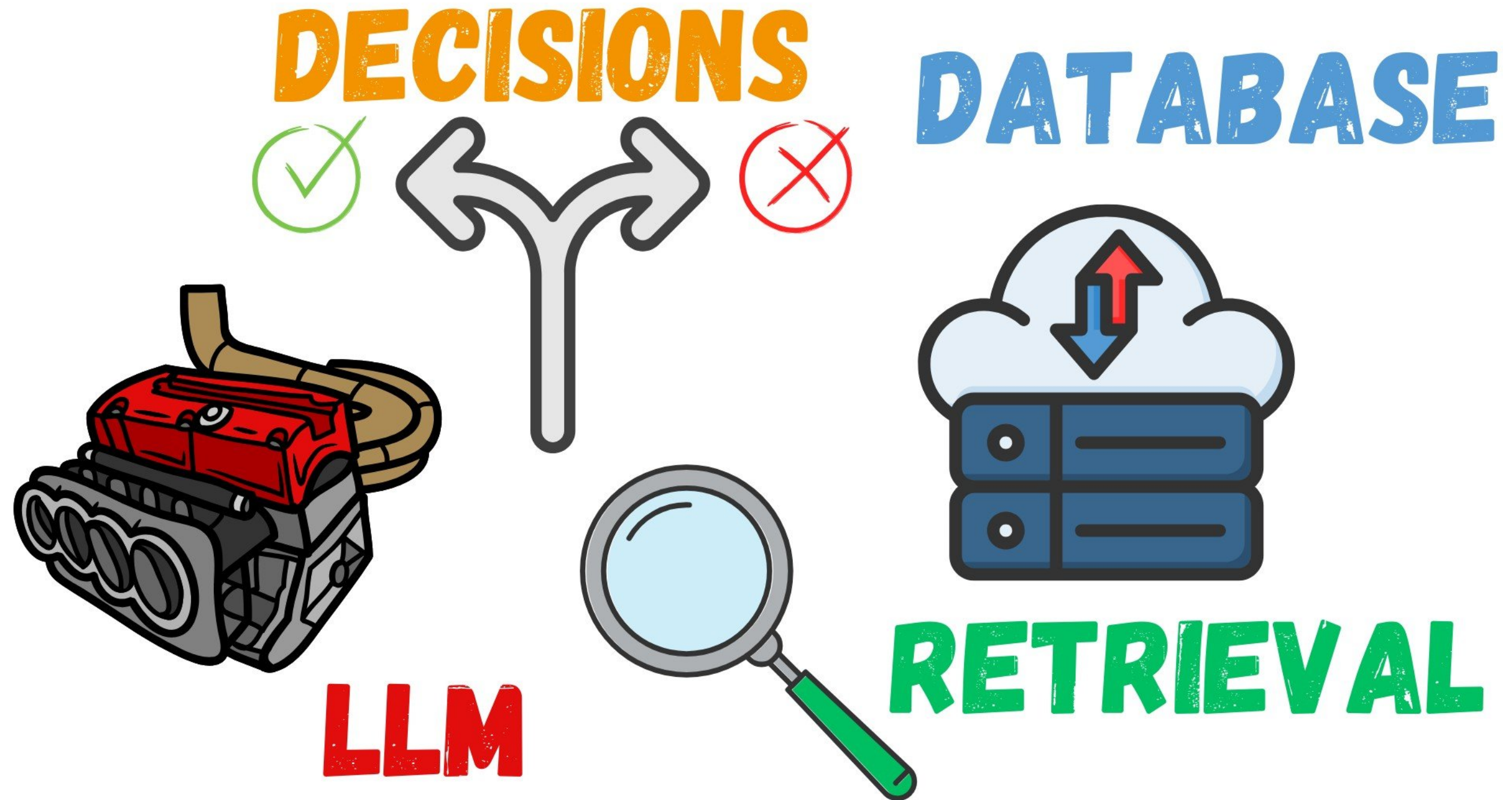




# Building LLM apps the LangChain way...



# Building LLM apps the LangChain way...



# Prompting OpenAI models

```
from langchain_openai import ChatOpenAI

llm = ChatOpenAI(
    model="gpt-4o-mini",
    api_key='...'
)

llm.invoke("What is LangChain?")
```

LangChain is a framework designed for developing applications...

- Additional parameters: `max_completion_tokens` , `temperature` , etc.

<sup>1</sup> <https://platform.openai.com/docs/quickstart>

# □ Prompting Hugging Face models

```
from langchain_huggingface import HuggingFacePipeline

llm = HuggingFacePipeline.from_model_id(
    model_id="meta-llama/Llama-3.2-3B-Instruct",
    task="text-generation",
    pipeline_kwargs={"max_new_tokens": 100}
)

llm.invoke("What is Hugging Face?")
```

Hugging Face is a popular open-source artificial intelligence (AI) library...

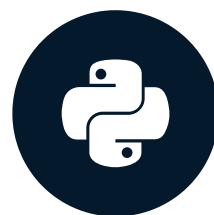
# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



# Prompt templates

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



**James Chapman**

Curriculum Manager, DataCamp

# Prompt templates

- Recipes for defining prompts for LLMs
- Can contain: instructions, examples, and additional context



# Prompt templates

```
from langchain_core.prompts import PromptTemplate

template = "Expain this concept simply and concisely: {concept}"
prompt_template = PromptTemplate.from_template(
    template=template
)

prompt = prompt_template.invoke({"concept": "Prompting LLMs"})
print(prompt)
```

```
text='Expain this concept simply and concisely: Prompting LLMs'
```

```
llm = HuggingFacePipeline.from_model_id(  
    model_id="meta-llama/Llama-3.3-70B-Instruct",  
    task="text-generation"  
)  
llm_chain = prompt_template | llm  
  
concept = "Prompting LLMs"  
print(llm_chain.invoke({"concept": concept}))
```

Prompting LLMs (Large Language Models) refers to the process of giving a model a specific input or question to generate a response.

- LangChain Expression Language (LCEL): `|` (pipe) operator
- **Chain:** connect calls to different components

# Chat models

- Chat roles: `system` , `human` , `ai`

```
from langchain_core.prompts import ChatPromptTemplate

template = ChatPromptTemplate.from_messages(
    [
        ("system", "You are a calculator that responds with math."),
        ("human", "Answer this math question: What is two plus two?"),
        ("ai", "2+2=4"),
        ("human", "Answer this math question: {math}")
    ]
)
```



# Integrating ChatPromptTemplate

```
llm = ChatOpenAI(model="gpt-4o-mini", api_key='<OPENAI_API_TOKEN>')

llm_chain = template | llm
math='What is five times five?'

response = llm_chain.invoke({"math": math})
print(response.content)
```

5x5=25

# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN

# Few-shot prompting

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN



**James Chapman**

Curriculum Manager, DataCamp

# Limitations of standard prompt templates

- `PromptTemplate` + `ChatPromptTemplate`
- Handling small numbers of examples
- Don't scale to larger numbers
- `FewShotPromptTemplate`

```
examples = [  
    {  
        "question": "..."  
        "answer": "..."  
    },  
    ...  
]
```

# Building an example set

```
examples = [  
    {  
        "question": "Does Henry Campbell have any pets?",  
        "answer": "Henry Campbell has a dog called Pluto."  
    },  
    ...  
]
```

```
# Convert pandas DataFrame to list of dicts  
examples = df.to_dict(orient="records")
```



# Formatting the examples

```
from langchain_core.prompts import FewShotPromptTemplate, PromptTemplate

example_prompt = PromptTemplate.from_template("Question: {question}\n{answer}")

prompt = example_prompt.invoke({"question": "What is the capital of Italy?"
                                "answer": "Rome"})

print(prompt.text)
```

```
Question: What is the capital of Italy?
Rome
```

# FewShotPromptTemplate

```
prompt_template = FewShotPromptTemplate(  
    examples=examples,  
    example_prompt=example_prompt,  
    suffix="Question: {input}",  
    input_variables=["input"]  
)
```

- `examples` : the list of dicts
- `example_prompt` : formatted template
- `suffix` : suffix to add to the input
- `input_variables`

# Invoking the few-shot prompt template

```
prompt = prompt_template.invoke({"input": "What is the name of Henry Campbell's dog?"})  
print(prompt.text)
```

```
Question: Does Henry Campbell have any pets?  
Henry Campbell has a dog called Pluto.  
...
```

```
Question: What is the name of Henry Campbell's dog?
```

# Integration with a chain

```
llm = ChatOpenAI(model="gpt-4o-mini", api_key="...")

llm_chain = prompt_template | llm
response = llm_chain.invoke({"input": "What is the name of Henry Campbell's dog?"})
print(response.content)
```

The name of Henry Campbell's dog is Pluto.

# Let's practice!

DEVELOPING LLM APPLICATIONS WITH LANGCHAIN