

```
= Initialising the LLM
:order: 2
:type: lesson
```

In this lesson, you will:

- * Install the Langchain Python package
- * Initialize an LLM
- * Get a response from the LLM using a **Prompt**

== Choosing your LLM

Langchain supports multiple LLM providers ([link:https://openai.com/](https://openai.com/)[OpenAI^], [link:https://cohere.com/](https://cohere.com/)[Cohere^], [link:https://ollama.ai/](https://ollama.ai/)[Ollama^] and more). The quality and cost vary. LLMs are also trained for different purposes, so it is worth experimenting with models and prompts. Depending on your use case, different LLMs may return better results.

This course includes instructions for using [link:https://openai.com/](https://openai.com/)[OpenAI^], but the same principles apply to all LLMs.

[NOTE]

.Using OpenAI

If you wish to use OpenAI and follow this course's practical activities, you must create an account and set up billing.

== Setup

=== Installing Langchain

To use Langchain, you must install the ``langchain`` package and its dependencies using ``pip``:

[source,sh]

.Install Langchain

```
pip install langchain
```

[TIP]

.Virtual Environment

====

You may find it helpful to create a Python virtual environment using a tool like [link:https://virtualenv.pypa.io/en/latest/](https://virtualenv.pypa.io/en/latest/)[virtualenv^]. Using a virtual environment allows you to install packages without affecting your system Python installation.

====

During the course, you will also be using components from the ``neo4j``, ``langchain-community`` and ``langchainhub`` packages:

[source,sh]

.Install Langchain

```
pip install langchain-community langchainhub neo4j
```

=== Installing OpenAI

You will need to setup OpenAI at

link:<https://platform.openai.com>[platform.openai.com], including:

- * Creating an account
- * Creating an API key
- * Setting up billing

You can install the `openai` and `langchain-openai` Python packages using `pip`:

```
[source,sh]
```

```
.Install OpenAI SDK
```

```
pip install openai langchain-openai
```

== Create a Langchain application

Create a new Python program and copy this code into a new Python file.

```
[source,python]
```

```
----
```

```
from langchain_openai import OpenAI
```

```
llm = OpenAI(openai_api_key="sk-...")
```

```
response = llm.invoke("What is Neo4j?")
```

```
print(response)
```

```
----
```

```
[IMPORTANT]
```

```
.OpenAI API Key
```

Remember to include your OpenAI API key in the `openai_api_key` parameter.

Review the program and note the following:

- * That `OpenAI` is used as the LLM
- * You can pass a question to the `llm.invoke` and get a response

Running the program should produce a response from the LLM similar to:

Neo4j is an open-source, NoSQL graph database management system developed by Neo4j, Inc. It is designed to store and manage data in a graph-like structure using nodes, relationships, and properties. Neo4j is used for a wide variety of use cases including real-time data analytics, fraud detection, recommendation engines, and network and IT operations.

Try modifying the program to ask a different question.

== Prompts

Prompt templates allow you to create reusable instructions or questions. You can use them to create more complex or structured input for the LLM.

Below is an example of a prompt template:

```
[source, python]
----
"""
You are a cockney fruit and vegetable seller.
Your role is to assist your customer with their fruit and vegetable needs.
Respond using cockney rhyming slang.

Tell me about the following fruit: {fruit}
"""
----
```

This prompt template would give context to the LLM and instruct it to respond as a cockney fruit and vegetable seller.

You can define parameters within the template using braces `{}` e.g. `{fruit}`. These parameters will be replaced with values when the prompt is formatted.

Modify your program to use the prompt template:

```
[source,python]
----
from langchain.prompts import PromptTemplate

template = PromptTemplate(template="""
You are a cockney fruit and vegetable seller.
Your role is to assist your customer with their fruit and vegetable needs.
Respond using cockney rhyming slang.

Tell me about the following fruit: {fruit}
""", input_variables=["fruit"])
----
```

Call the LLM, passing the formatted prompt template as the input:

```
[source,python]
----
response = llm.invoke(template.format(fruit="apple"))

print(response)
----
```

You use the `format` method to pass the parameters to the prompt e.g. `fruit="apple"`. The input variables will be validated when the prompt is formatted, and a `KeyError` will be raised if any variables are missing from the input.

The prompt will be formatted as follows:

```
You are a cockney fruit and vegetable seller.
Your role is to assist your customer with their fruit and vegetable
needs
Respond using cockney rhyming slang.
Tell me about the following fruit: apple
```

When running the program, you should see a response similar to:

```
Well, apples is a right corker - they come in all shapes and sizes
from Granny Smiths to Royal Galas. Got 'em right 'ere, two a penny - come
and grab a pick of the barrel!
```

[NOTE]

.Differing Results

If you run the program multiple times, you will notice you get different responses because the LLM is generating the answer each time.

Before moving on to the next lesson, try creating a prompt template and using it to get a response from the LLM.

[TIP]

.Creating PromptTemplates

You can create a prompt from a string by calling the ``PromptTemplate.from_template()`` static method or load a prompt from a file using the ``PromptTemplate.from_file()`` static method.

== Configuring the LLM

When you create the LLM, you can configure it with parameters such as the ``temperature`` and ``model``.

[source,python]

```
----
llm = OpenAI(
    openai_api_key="sk-...",
    model="gpt-3.5-turbo-instruct",
    temperature=0
)
----
```

When selecting a model, it is worth considering the quality of the output and the cost per token. There are several [link:https://platform.openai.com/docs/models/overview](https://platform.openai.com/docs/models/overview) [OpenAI models^] available, each with different characteristics.

As you learned in the first module, all prompts have a ``temperature``. The temperature, between ``0.0`` and ``1.0``, affects the randomness or creativeness of the response.

Note how the code above uses the "gpt-3.5-turbo-instruct" model with a temperature of ``0.0``. Typically, this will produce a good response, as

grounded in fact as possible.

== Check Your Understanding

include::questions/1-prompts.adoc[leveloffset=+1]

[.summary]

== Lesson Summary

In this lesson, you learned how to communicate with an LLM with text and prompt templates.

In the next lesson, you will learn all about Langchain ****Chains****.