1. Generate a short product description for a new smartphone, focusing on its top three unique features.
2. Write a brief character introduction for the protagonist of a mystery novel.
3. Create a concise mission statement for a non-profit organization dedicated to environmental conservation.
4. Generate a short description of a futuristic city in a science fiction story.
5. Write a brief dialogue between two friends discussing their favorite hobbies.
6. Create a short description of a magical artifact in a fantasy world.

For each scenario, focus on providing clear instructions and additional context that define the task you want the language model to perform.

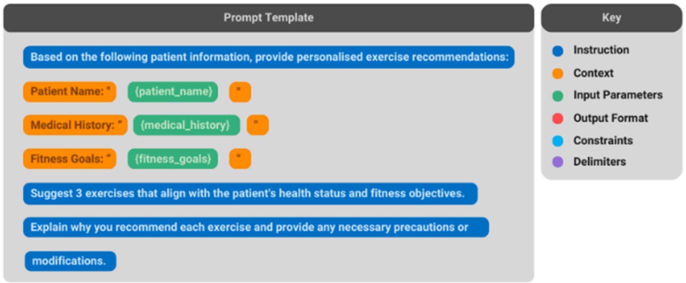
3.3 Input Parameters

Input parameters are dynamic variables in a prompt template that are replaced with specific values before the prompt is sent to the language model for execution. By using placeholders for inputs, you can create flexible and reusable prompt templates that can be customized with different data points as needed. The placeholders act as slots where the actual values will be populated, either manually by the user or programmatically using software libraries like LangChain. This approach streamlines the prompt engineering process, as you don't need to create a new prompt for variations of the input data.

There are two commonly used syntaxes for specifying placeholders in prompts: the f-string syntax from Python `{placeholder}` and the Jinja2 templating syntax `{{placeholder}}`. Both syntaxes are supported by popular Python libraries for large language models, prompt engineering, and retrieval-augmented generation.

For example, the LangChain library, which is widely used for building applications with large language models, supports both f-string and Jinja2 templating syntax for defining prompt templates. Additionally, the Prompt Registry PromptLayer, which will be discussed in the Prompt Management chapter, also supports both syntaxes.

The following diagram illustrates a prompt template for personalized exercise recommendations, highlighting the effective usage of input parameters.



Based on the following patient information, provide personalised exercise recommendations:
Patient Name: " {patient_name} "
Medical History: " {medical_history} "
Fitness Goals: " {fitness_goals} "
Suggest 3 exercises that align with the patient's health status and fitness objectives.
Explain why you recommend each exercise and provide any necessary precautions or modifications.

INPUT PARAMETERS

Licensed to Jeff Kittrell <huskerjeff6224@yahoo.com>

- The prompt template includes three input parameters: `{{patient_name}}`, `{{medical_history}}`, and `{{fitness_goals}}`.
- These parameters act as placeholders for patient-specific information that will be provided when the prompt is used.
- The use of input parameters makes the prompt template flexible and reusable, allowing it to be customized for different patients.

APPROPRIATE NAMING

- The input parameter names are descriptive and clearly indicate the type of information expected.
- `{{patient_name}}` represents the name of the patient.
- `{{medical_history}}` represents the patient's relevant medical history.
- `{{fitness_goals}}` represents the patient's specific fitness objectives.
- The naming convention follows the Jinja2 templating syntax, using double curly braces `{{}}`.

STRATEGIC PLACEMENT

- The input parameters are strategically placed within the prompt template to ensure clarity and coherence.
- `{{patient_name}}` is placed in the context section, providing the patient's identity.
- `{{medical_history}}` and `{{fitness_goals}}` are placed in the input section, providing relevant information for generating personalized recommendations.
- The placement of the input parameters aligns with the logical flow of the prompt and helps guide the user in providing the necessary information.

By effectively using input parameters, the prompt template demonstrates the principles of appropriate naming, and strategic placement. This approach enables the creation of a flexible and reusable prompt for generating personalized exercise recommendations based on patient-specific information.

3.3.1 Practical Example 1: Sentiment Analysis

A company wants to analyse customer feedback to gauge the sentiment towards their products or services. They have a dataset of customer reviews and want to automatically determine the sentiment expressed in each review.

PROMPT TEMPLATE

Licensed to Jeff Kittrell <huskerjeff6224@yahoo.com>

~~~
```

Determine whether the sentiment is positive, negative, or neutral, and provide a brief
explanation for your conclusion.
Analyze the sentiment of the following customer review: "{review_text}"
```
~~~

PROMPT

~~~
```

Determine whether the sentiment is positive, negative, or neutral, and provide a brief
explanation for your conclusion.
Analyze the sentiment of the following customer review: "I absolutely loved the product!
It exceeded my expectations and solved my problem perfectly. The quality is top-notch,
and the customer support was incredibly helpful. I highly recommend this to anyone
looking for a reliable solution."
```
~~~

GENERATED CONTENT

~~~
```

The sentiment of the given customer review is highly positive. The reviewer expresses
strong satisfaction with the product, using phrases like "absolutely loved," "exceeded
my expectations," and "top-notch quality." They also praise the customer support as
"incredibly helpful" and highly recommend the product to others. The overall tone and
language used in the review indicate a very positive sentiment towards the company and
its offerings.
```
~~~

The prompt template for sentiment analysis uses the **{review_text}** placeholder to dynamically insert the actual customer review into the prompt. By providing a specific review as the user input, the language model analyses the sentiment expressed in that review and generates an output that determines the sentiment (positive, negative, or neutral) along with a brief explanation. The generated output accurately identifies the highly positive sentiment in the given review, highlighting key phrases and overall language that support this conclusion. This demonstrates how the prompt template can be used to automate sentiment analysis on a large dataset of customer reviews.

Licensed to Jeff Kittrell <huskerjeff6224@yahoo.com>

Input parameters are particularly powerful when you need to generate content that requires dynamic information. By utilizing input placeholders, you can create flexible prompt templates that can be easily adapted to various scenarios by replacing the placeholders with different values, like how arguments work in programming functions. This makes inputs an invaluable tool for automating the generation of personalized or customized content efficiently.

In the previous example, we used a single input (customer review) to demonstrate sentiment analysis. Now, let's explore how multiple inputs can be used to generate personalized product recommendations based on customer information.

3.3.2 Practical Example 2: Personalized Product Recommendation

An e-commerce company wants to provide personalized product recommendations to its customers based on their previous purchase history and preferences. They have data on each customer's past orders and want to generate tailored product suggestions.

PROMPT TEMPLATE

~~~
```


Based on the following customer information, provide personalized product
recommendations:
Customer Name: “{{customer_name}}”
Previous Purchases: “{{previous_purchases}}”
Preferences: “{{preferences}}”
Suggest 3 products that align with the customer's interests and explain why you
recommend each product.
```
~~~

PROMPT

~~~
```


Based on the following customer information, provide personalized product
recommendations:
Customer Name: “Emily Johnson”
Previous Purchases: “Wireless Headphones, Fitness Tracker, Yoga Mat”
Preferences: “Technology gadgets, Fitness equipment, Eco-friendly products
Suggest 3 products that align with the customer's interests and explain why you
recommend each product.”
```
~~~

GENERATED CONTENT

Licensed to Jeff Kittrell <huskerjeff6224@yahoo.com>

~~~
```

Based on Emily Johnson's previous purchases and preferences, here are 3 personalized
product recommendations:
1. Smart Water Bottle: Given Emily's interest in technology gadgets and fitness
equipment, a smart water bottle would be a great fit. It can track her daily water
intake, remind her to stay hydrated, and sync with her fitness apps. This product
combines her love for tech and commitment to a healthy lifestyle.
2. Eco-Friendly Resistance Bands: Considering Emily's preference for eco-friendly
products and her previous purchase of a yoga mat, eco-friendly resistance bands would be
a suitable recommendation. These bands are made from sustainable materials and can be
used for a variety of workouts, including strength training and stretching. They are
portable and versatile, making them perfect for home workouts or on-the-go exercise.
3. Noise-Cancelling Earbuds: Since Emily has previously purchased wireless headphones,
she might be interested in upgrading to noise-cancelling earbuds. These earbuds provide
an immersive audio experience by blocking out external noise, making them ideal for
workouts, commuting, or focusing on work. They offer a combination of advanced
technology and convenience, aligning with her interest in tech gadgets.
```
~~~

The prompt template for personalized product recommendations uses placeholders for `**{{customer_name}}**`, `**{{previous_purchases}}**`, and `**{{preferences}}**` to dynamically incorporate customer-specific information into the prompt. By providing Emily Johnson's details as the user input, the language model generates personalized product suggestions based on her past purchases and stated preferences. The generated output recommends three products (Smart Water Bottle, Eco-Friendly Resistance Bands, and Noise-Cancelling Earbuds) and provides a rationale for each recommendation, highlighting how the products align with Emily's interests and needs. This showcases the effectiveness of using input placeholders to generate tailored content for individual customers.

Inputs are dynamic variables in a prompt template that are replaced with specific values before the prompt is sent to the language model. By using placeholders for inputs, you can create flexible and reusable prompt templates that can be customized with different data points as needed. This approach streamlines the prompt engineering process and allows for efficient generation of content across various scenarios. Practicing the creation of prompt templates with input placeholders will help you design more versatile and effective prompts that solve real-world problems.

When using inputs in your prompt templates, consider the following principles for effective input usage:

- Choose descriptive and meaningful names for your input variables. The names should clearly indicate the type of information expected.
- Maintain a consistent input format throughout the prompt. Decide on using either f-strings or Jinja2 templating and stick to that format for all input variables.

Licensed to Jeff Kittrell <huskerjeff6224@yahoo.com>