# 3. Title: Formatting Prompt in Python

| | | | |
|---|---|---|---|
| 📒 | 📒 | 🔧 | 👩‍💻 |
| Not So Typical Intro to LLMs | Prompt Engineering | Formatting Prompt in Python | Hands-on Walkthrough and Tasks |

**Notice the 🔧 Wrench icon for this page**

- The icon appears at the top of this page and also at the navigation bar on the left
- Pages with this icon contain the **key concepts/techniques that will directly help you with the hands-on tasks** (i.e., Notebook for weekly tasks)
- The intention for these pages is to **work as quick references, especially if you need to refer to some help when you are coding**.

This saves you time from opening up the Jupyter Notebook just to look for the techniques we covered.

• However, note that the Notebooks generally have more comprehensive examples and details for the discussed topics.

---

# Why Bother to Format the Prompt?

We have been talking about the importance of prompt engineering in general in 2. Prompt Engineering. In this note, we will cover the important techniques that we will be using to format the prompts in your Python code.

Mastering these basics will make calling LLMs programmatically using Python more efficient and is particularly important for applications that require complex prompts.

✦ **String Formatting (f-strings)**:

- When interacting with a Large Language Model (LLM) programmatically, you often need to construct prompts dynamically based on certain conditions or variables.
- `F-string formatting` in Python allows you to embed expressions or variable(s) inside a string for easier string formatting that also makes it more readable.
- Note that f-string formatting is only available in Python 3.6 and later versions.

Python

```
name = "Alice"
age = 25
print(f"Hello, my name is {name} and I am {age} years old.")
```

---

✦ **Multi-line Strings (Triple double or single quotes)**:

- LLMs often require multi-line prompts for complex tasks or to provide additional context.
- Triple quotes in Python allow you to define multi-line strings easily.
- This is especially useful when the prompt includes several lines of instructions or needs to maintain a specific formatting.

```python
print("""
Hello,
This is a multi-line string.
Each new line is preserved in the output.
""")
```

---

### ✦ Combining String Formatting and Multi-line Strings for Powerful Template

- Both techniques can create dynamic and reusable templates for Large Language Models (LLMs).
- **F-string formatting** allows you to insert any valid Python expression into the string, making the template highly flexible, while **Multi-line strings** allow you to preserve the formatting of the template, making it easier to read and understand.
- By including the template in a function, you can easily reuse it with different values.

```python
def greet(name, age):
    return f"""
    Hello, my name is {name}
    and I am {age} years old.
    """

print(greet("Alice", 25))
```

---

### ✦ Maintainability & Error Minimisation:

- Just like `function` in a Python program, it allows us to reduce duplicative parts of the prompts, allowing for the same template to be applied across different scenarios
- If a change is needed, it can be made in one place, and all instances of the template will reflect that change
- Proper string formatting helps prevent errors.
  - For example, forgetting to close a quote could lead to syntax errors. Using triple quotes for multi-line strings can help avoid such issues.

---

### ✦ Readability and Maintainability:

- Quickly discern the included inputs, which can differ depending on the actual user input
- Using f-strings and triple quotes makes your code more readable and maintainable.
- It allows others (or future you) to understand what your code is doing, which is always a good practice in programming.

### Next: Try out the practical examples in Weekly Tasks - Week 01

• The notebook also contains more **detailed implementations of the prompt formatting**.
• It also include the implementations of **various task-specific prompts**