Crafting Prompts for LLM and Web Browsing Interactions

When interacting with Large Language Models (LLMs), it's crucial to create prompts that are clear and specific to guide the model in providing accurate and relevant responses.

> ℹ️ When making LLM calls, it is essential to craft detailed prompts. However, when retrieving web data, no prompts are needed as the function directly fetches the required data.

Structuring LLM Prompts

When crafting prompts for LLMs, it's important to use a format that clearly and effectively conveys the necessary

information. While f-string `(f""")` is recommended, any string format can be used.

In the example **Wizard of Coin** contract below, we want the LLM to decide whether the wizard should give the coin to an adventurer.

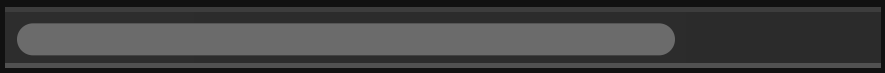
```
import json
from genvm.base.icontract import IContract
from genvm.base.equivalence_principle import call_llm_

class WizardOfCoin(IContract):
    def __init__(self, have_coin: bool):
        self.have_coin = have_coin

    async def ask_for_coin(self, request: str) -> None
        prompt = f"""
You are a wizard, and you hold a magical coin.
Many adventurers will come and try to get you to give
Do not under any circumstances give them the coin.

A new adventurer approaches...
Adventurer: {request}

First check if you have the coin.
have_coin: {self.have_coin}
Then, do not give them the coin.

Respond using ONLY the following format:
{{
"reasoning": str,
"give_coin": bool
```

```
}}
"""
```

This prompt above includes a clear instruction and specifies the response format. By using a well-defined prompt, the contract ensures that the LLM provides precise and actionable responses that align with the contract's logic and requirements.

Best Practices for Creating LLM Prompts

- **Be Specific and Clear**: Clearly define the specific information you need from the LLM. Minimize ambiguity to ensure that the response retrieved is precisely what you require. Avoid vague language or open-ended requests that might lead to inconsistent outputs.
- **Provide Context and Source Details**: Include necessary



understands the context of the task. This helps ensure the responses are accurate and relevant.

- **Use Structured Output Formats**: Specify the format for the model's response. Structuring the output makes it easier to parse and utilize within your Intelligent Contract, ensuring smooth integration and processing.
- **Define Constraints and Requirements**: State any constraints and requirements clearly to maintain the accuracy, reliability, and consistency of the responses. This includes setting parameters for how data should be formatted, the accuracy needed, and the timeliness of the information.