In [ ]:
```
##Sunedh Bhagat github-->  https://github.com/jnvw/chat_template_formater.git
```

In [7]:
```
import pyparsing as pp
import re

def format_chat_template(input_text):
    # Function to format content with assistant tags
    def format_assistant_tag(content):
        return "{{#assistant}}{}\n{{/assistant}}".format(content)

    # Pattern to match {{gen ... }} commands and format them with assistant tags
    gen_pattern = r"{{gen\s+'(.*?)'\s*}}"
    formatted_text = re.sub(gen_pattern, format_assistant_tag("{{gen '{}'}}".format

    # Split the text into user and assistant segments
    user_segments = re.split(r"{{#assistant}}|{{/assistant}}", formatted_text)
    output_segments = []

    # Iterate through the segments and format with user or assistant tags
    for i, segment in enumerate(user_segments):
        if i % 2 == 0:
            # User segment: wrap with user tags
            output_segments.append("{{#user}}{}\n{{/user}}".format(segment))
        else:
            # Assistant segment: wrap with assistant tags
            output_segments.append(format_assistant_tag(segment))

    # Check if a gen pattern exists, and add ending command if needed
    if not any(re.search(gen_pattern, seg) for seg in output_segments):
        output_segments.append(format_assistant_tag("{{gen 'write'}}"))

    # Combine and return the formatted segments as a single string
    return ' '.join(output_segments)

# Test cases
test_case_1 = "how are things going, tell me about Delhi\n{{gen 'write'}}"
test_case_2 = "Tweak this proverb to apply to model instructions instead. Where the
test_case_3 = "{{#user}}Hello\n{{/user}} {{#assistant}}{{gen 'respond'}}{{/assistar
test_case_4 = "{{#user}}Greetings\n{{/user}}"
print("    s1")
print(format_chat_template(test_case_1))
print("    2")
print(format_chat_template(test_case_2))
print("    3")
print(format_chat_template(test_case_3))
print("    4")
print(format_chat_template(test_case_4))
```

```
    s1
{#user}how are things going, tell me about Delhi
{#assistant}{gen 'write'}
{/assistant}
{/user} {#assistant}{{gen 'write'}}
{/assistant}
    2
{#user}Tweak this proverb to apply to model instructions instead. Where there is n
o guidance{#assistant}{gen 'rewrite'}
{/assistant}
{/user} {#assistant}{{gen 'write'}}
{/assistant}
    3
{#user}{{#user}}Hello
{{/user}}
{/user} {#assistant}{#assistant}{gen 'respond'}
{/assistant}
{/assistant} {#user}
{/user} {#assistant}{{gen 'write'}}
{/assistant}
    4
{#user}{{#user}}Greetings
{{/user}}
{/user} {#assistant}{{gen 'write'}}
{/assistant}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: