Tool Prompting

ReAct et al

- Not all models have tool use built in

- A popular method to use tool calling with other models is ReAct prompting

- Kani ensures that prompts are formatted correctly and allows researchers to switch between native and ReAct tool calling

USER: What's the weather in Philly?

Thought 1: To find the weather, I should use the provided get_weather tool.
  Philadelphia is in the United States, which uses Fahrenheit to measure temperature.
Action 1: get_weather[Philadelphia, Fahrenheit]
Observation 1: cloudy; 75F

Thought 2: I now know the weather in Philadelphia.
Action 2: Finish[It's currently 75F and cloudy in Philadelphia, PA.]

16

Tool Prompting

ReAct et al

- Not all models have tool use built in
- A popular method to use tool calling with other models is ReAct prompting
- Kani ensures that prompts are formatted correctly and allows researchers to switch between native and ReAct tool calling

```
USER: What's the weather in Philly?

Thought 1: To find the weather, I should
use the provided get_weather tool.
  Philadelphia is in the United States,
which uses Fahrenheit to measure
temperature.
Action 1: get_weather[Philadelphia,
Fahrenheit]
Observation 1: cloudy; 75F

Thought 2: I now know the weather in
Philadelphia.
Action 2: Finish[It's currently 75F and
cloudy in Philadelphia, PA.]
```

Sub-Kani

Using other LLMs as tools

```python
class KaniWithSummary(Kani):
  @ai_function()
  async def summarize_conversation(self):
    """Get the summary of the conversation."""
    long_context_engine = OpenAIEngine(api_key,
                          model="gpt-4-32k")
    sub_kani = Kani(long_context_engine,
        chat_history=self.chat_history[:-2])
    summary = await sub_kani.chat_round(
      "Please summarize the conversation so far.")
    return summary.content
```