TYPHOON

Open Models, Smarter Agents

Practical Lessons from Modern Agentic Workflows

Pittawat Taveekitworachai (Pete)
Research Scientist, SCB DataX

9 March 2026
FOSSASIA Summit 2026

What Is Typhoon?

Typhoon is an **advanced research initiative** focused on developing **open-source language technologies for the Thai language.** We provide **models, datasets, tools, and research** to **advance Thai language AI and multimodal capabilities**

Optimized For Thai

Thai language technologies for Thai problems, e.g., LLM, OCR, ASR, TTS, and more

Open Source

Better controllability, cost, flexibility, privacy, and more

Open access to resources fosters collaboration and drives AI innovation

Recently Released Models

JAN 10

Typhoon 2 (Text)

A family of text and multimodal models designed for real-world applications with up to 128K context length and function-calling capabilities

MAR 17

Typhoon 2 R1 (Reasoning)

A 70B-parameter model combining DeepSeek R1's reasoning capabilities with Typhoon 2's Thai fluency for up to 6x better math & coding performance

MAY 5

Typhoon 2.1 Gemma (Text)

Lightweight (4B & 12B) models that outperform Typhoon 2 70B in Thai with a toggleable reasoning mode

MAY 16

Typhoon OCR

Bilingual vision-language OCR model for English & Thai that preserves layout structure, outperforming GPT-4o and Gemini 2.5 Flash on Thai document parsing

JUNE 23

Typhoon Translate

Lightweight 4B Thai-English translation model that preserves tone & meaning, outperforming GPT-4o/Claude/Gemini

SEP 8

Typhoon ASR Real-Time (Audio)

Streaming Thai ASR model offering 4,097× real-time speed with near-instant transcription on CPUs & compact GPUs

OCT 20

Typhoon 2.5

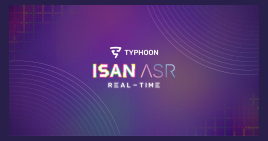
Typhoon 2.5 brings agentic intelligence, ultra efficiency, and natural Thai fluency to real-world workflows.

Nov 14

Typhoon OCR 1.5

Smaller model with 2B params. Outperforms both GPT-5 and Gemini 2.5 Flash in Thai document understanding, particularly on documents with complex layouts and mixed-language content.

Nov 27



Typhoon Isan ASR

Open-source Thai speech recognition model specialized for the Isan dialect

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

Chatbot

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

Chatbot

Sorry, but I cannot cook a pizza for you!
Here's a recipe for a pizza that you can cook yourself:

1. ...
2. ...
3. ...

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

Chatbot

Sorry, but I cannot cook a pizza for you!
Here's a recipe for a pizza that you can cook yourself:

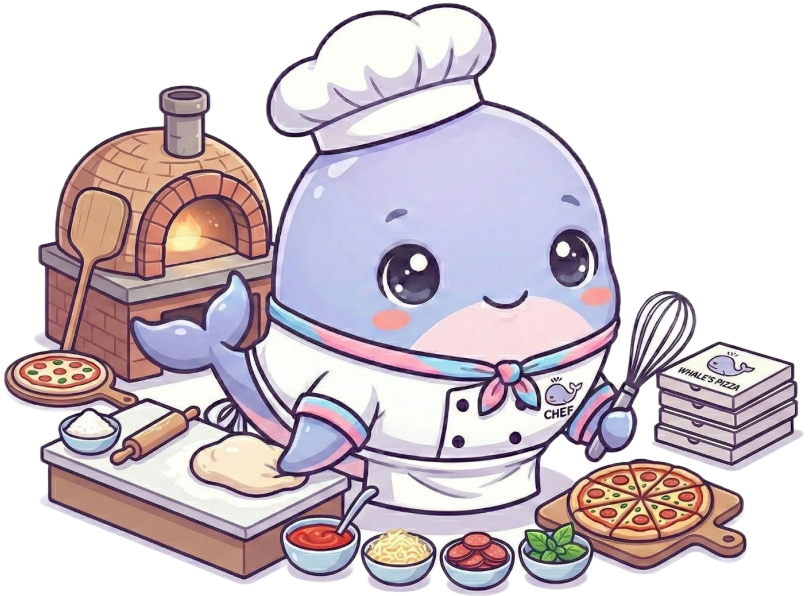
1. ...
2. ...
3. ...

*But I want a **pizza**, not a **pizza recipe***

Let’s Give the Chatbot Tools

Chatbot

Tools



Agent

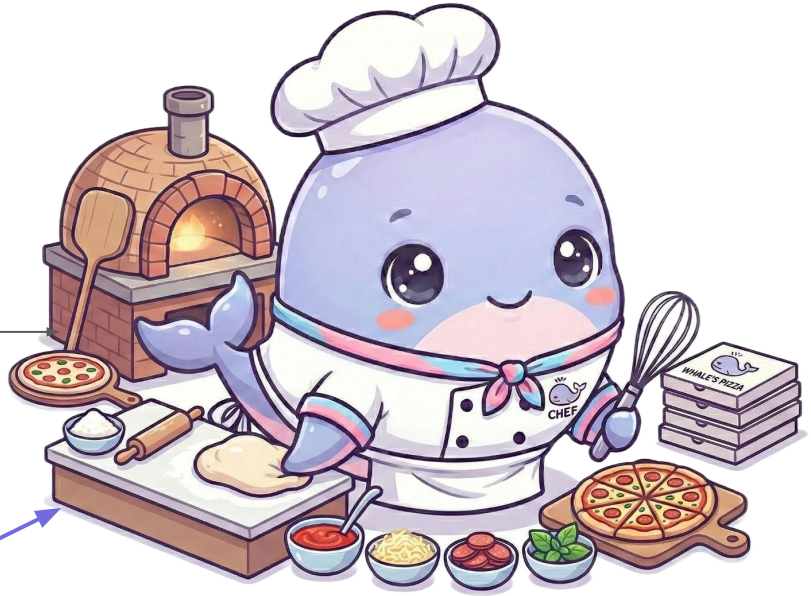
How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

Agent

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

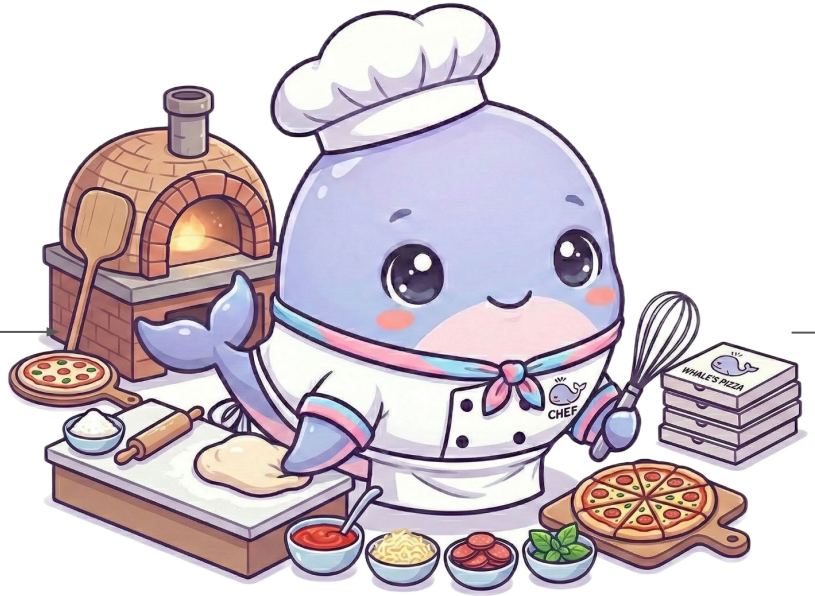


*These are **tools**!*

Agent

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕

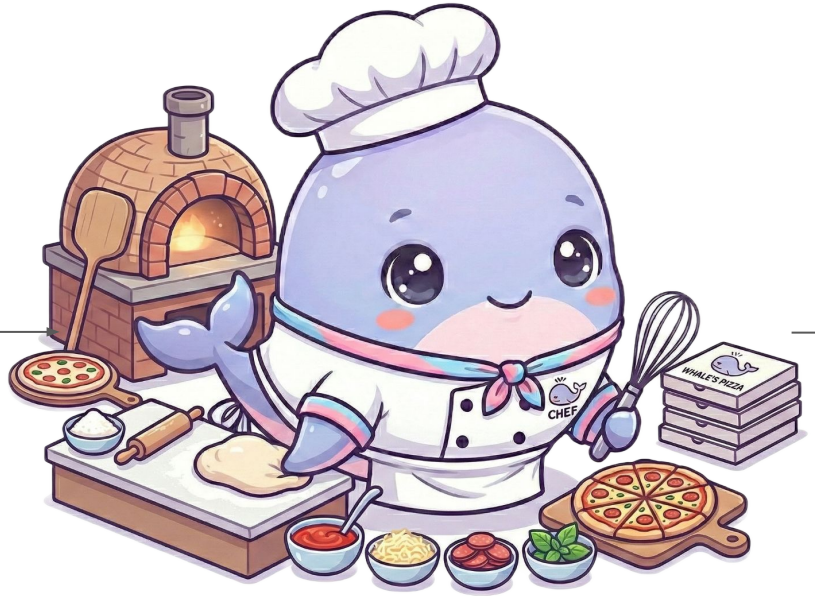


Agent

Certainly! I'm using a ***rolling pin*** to stretch the dough.

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕



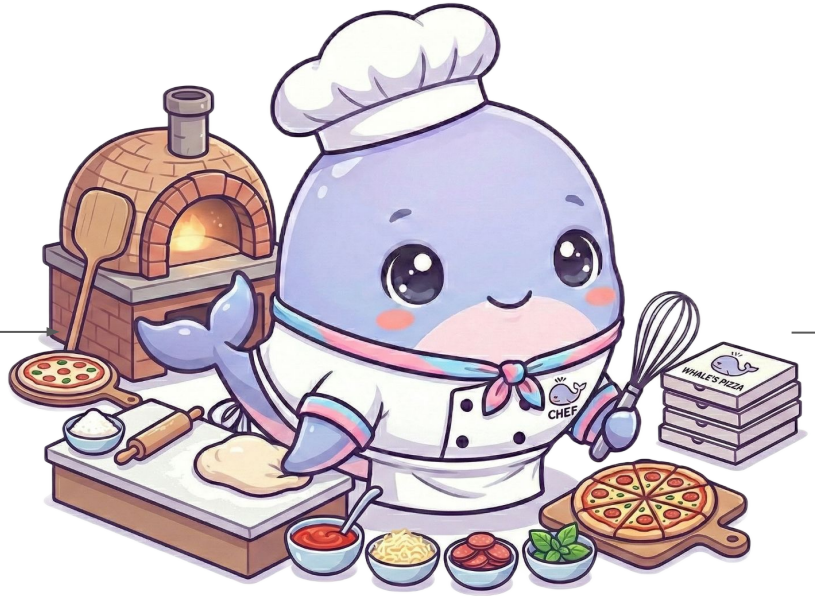
Certainly! I'm using a ***rolling pin*** to stretch the dough.

Agent

*The model used the **tool**!*

How To Turn A Chatbot Into An Agent?

I want to order a pizza! 🍕



Agent

Certainly! I'm using a ***rolling pin*** to stretch the dough.

*We got the **dough**, but not the **pizza**!*

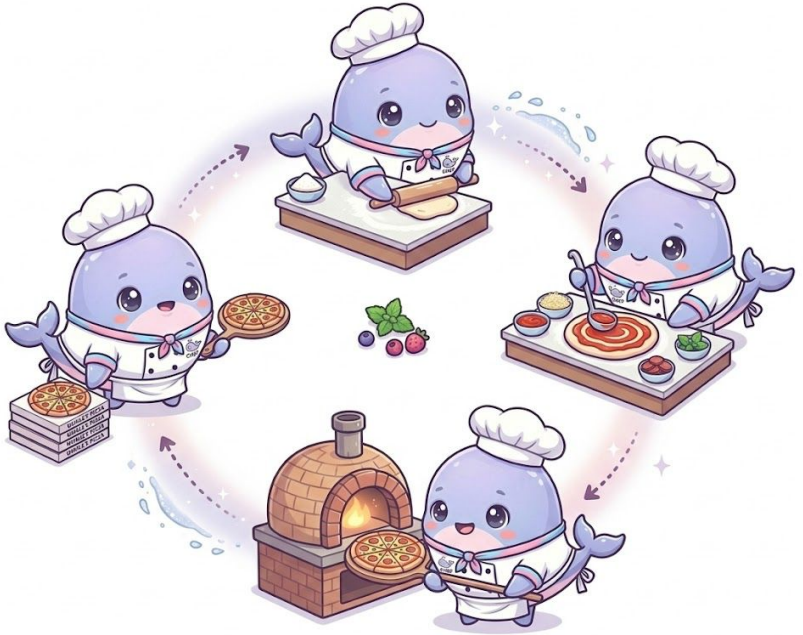
What's **Missing**?

The **Loop**!

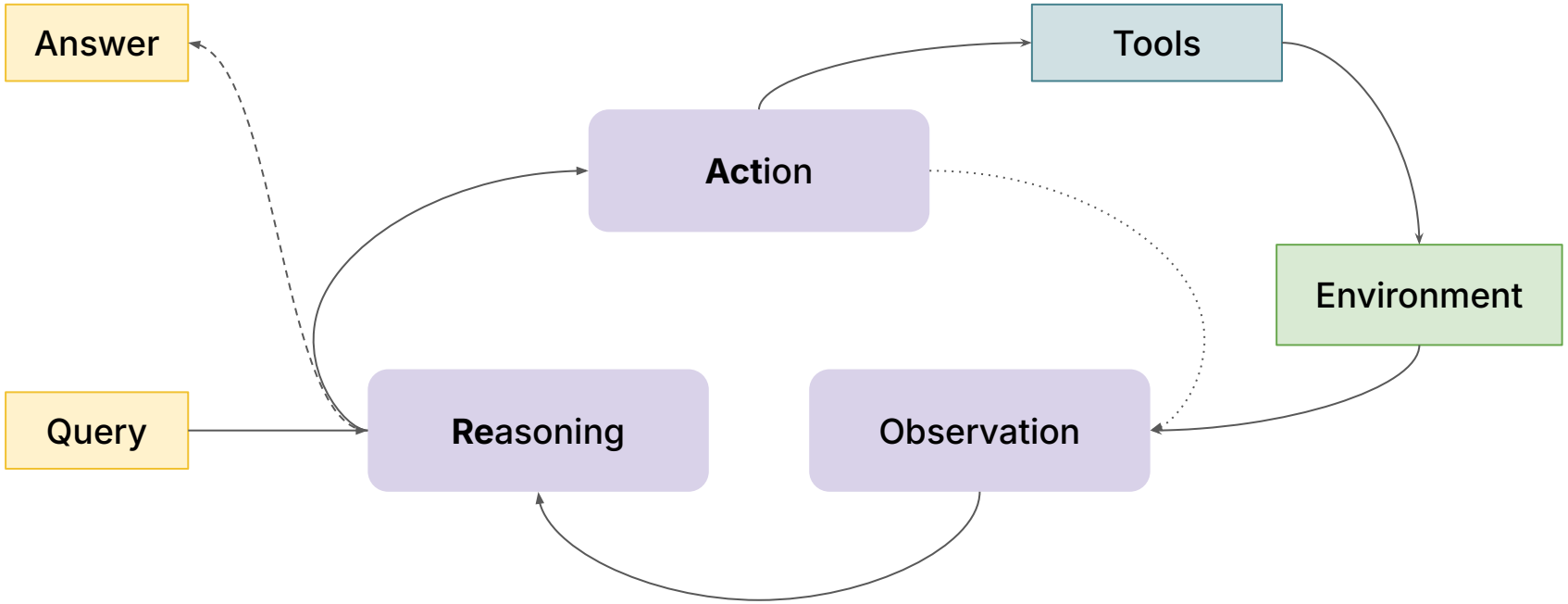
INTRODUCING

Agent Loop

Make the agent continue performing the next step until the task is accomplished

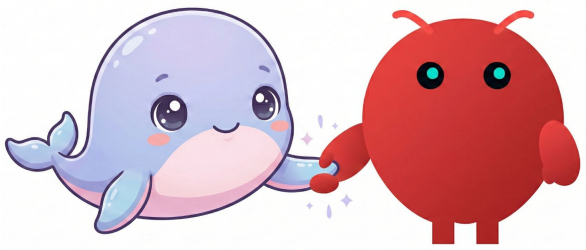


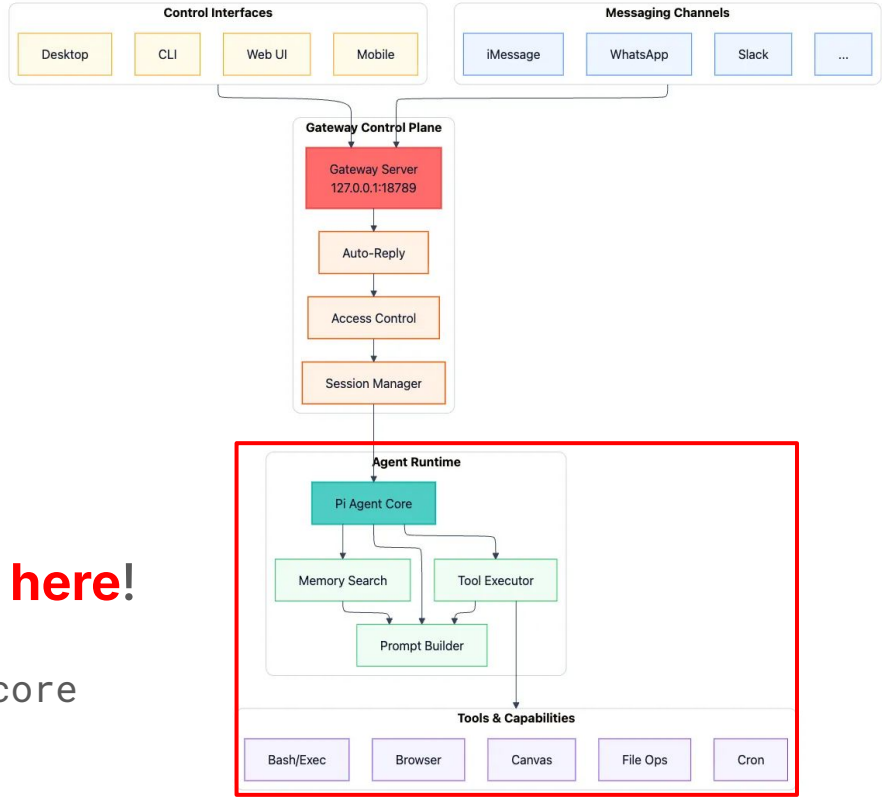
The Agent Loop (a.k.a ReAct ≠ React)



Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," *Proc. 11th ICLR*, 2023.

OpenClaw Agent Loop





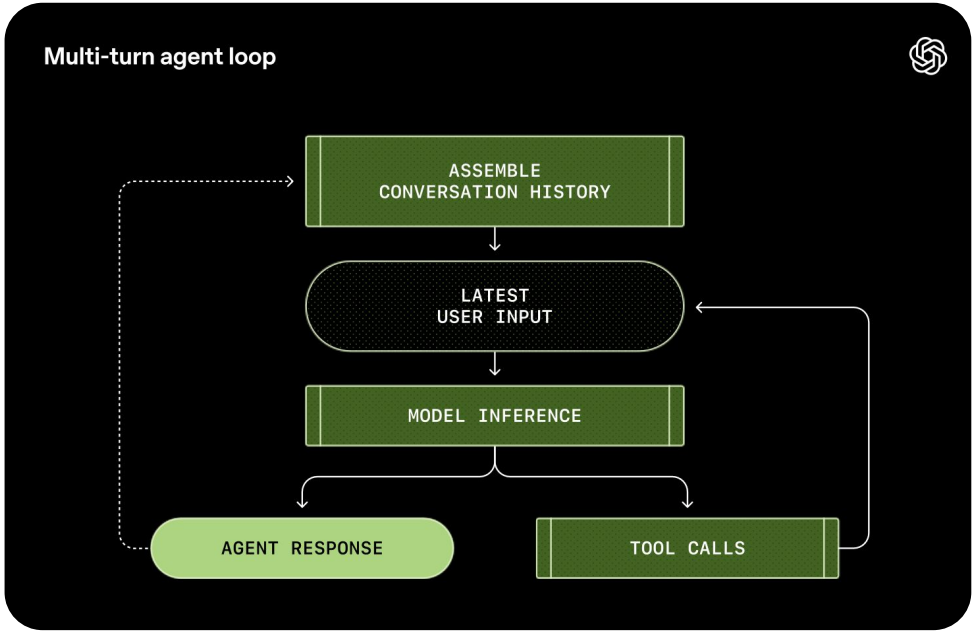
The **core intelligence** is happening **here**!

npm **package:** `@mariozechner/pi-agent-core`

https://ppaolo.substack.com/p/openclaw-system-architecture-overview

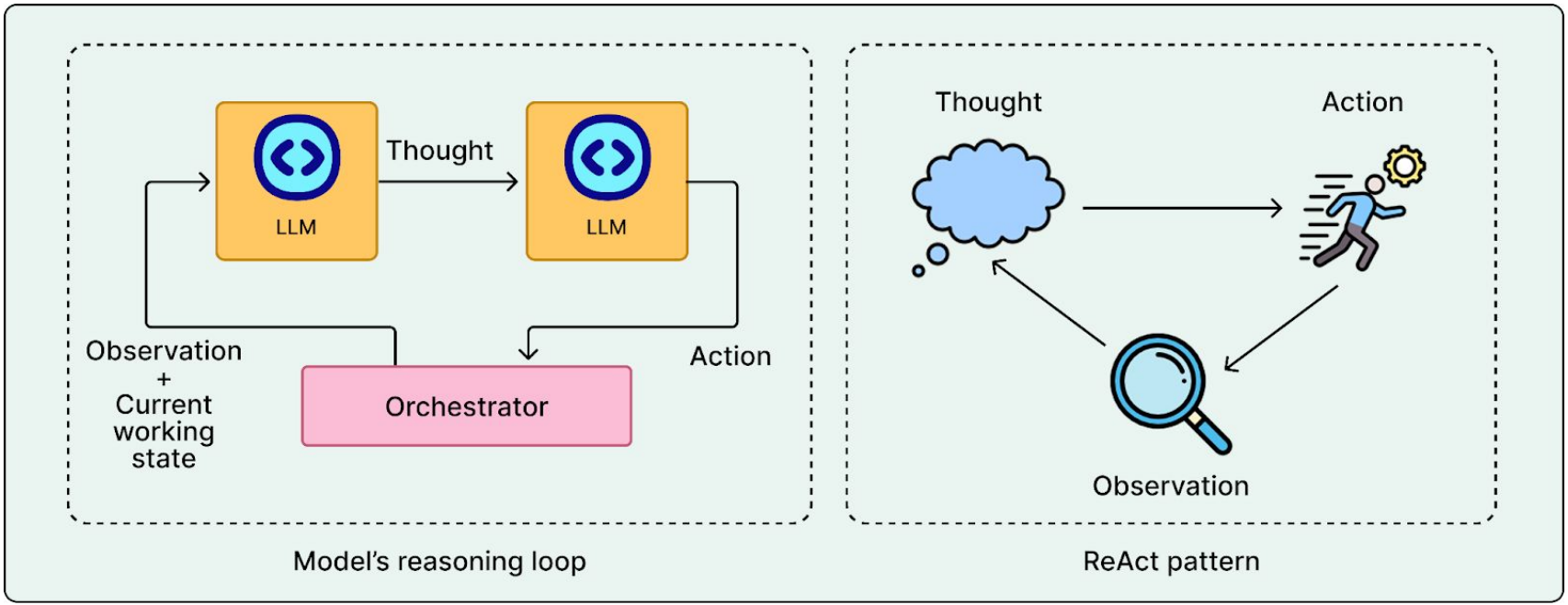
Codex Agent Loop

Codex



https://openai.com/index/unrolling-the-codex-agent-loop/

Cursor Agent Loop



https://blog.bytebytego.com/p/how-cursor-shipped-its-coding-agent

The Myth of Model Size

Evaluation-Driven Development

Context Engineering

Tools

Evaluation-Driven Development

In Software Development, How Can You **Be Sure** That You Have Successfully Fixed a Bug?

“If It's Not Tested, It's Broken”

—Bruce Eckel

Computer scientist and author of 'Thinking in Java' and 'Thinking in C++'

LLM is **Nondeterministic**

LLM is a **Black Box**


Evaluation provides a
development signal

Metrics - What and How to Measure?

Functional Metrics

Accuracy, precision/recall, calibration, etc.

Non-Functional Metrics

Latency, throughput, user preferences, etc.

Uncertainty Management

How to evaluate a **non-deterministic model**?

INTRODUCING

LM Judge

Prompt a model to become an LM judge for non-deterministic outputs.

Rubrics: Not Just Any Kinds of Judge

Test Cases - Evaluation Layers

End-to-End	**Application:** Human preference and the “vibe” of the software **Workflow**: "Start-to-Finish" path ensuring the entire system delivers the final value **Subworkflow**: Validate a specific chain of events (e.g., "Search for lead" → "Draft email").
Integration Tests	**Scenario**: Test the system's ability to handle mocked API responses, rate limits, and network timeouts **Agent**: Verify the model chooses the correct tool and parses the tool's output to formulate a response
Unit Tests	**Model**: Ensure the model follows instructions and maintains tone correctly **Tool**: Verify the code logic is sound, handles edge cases, and returns expected data structures

Context Engineering

Context is Everything

Input

System prompt, user prompt, chat history, memory, RAG content, etc.

Output

Generated responses

Intermediate content

Tool calls, tool outputs, etc.

The “Real” Context

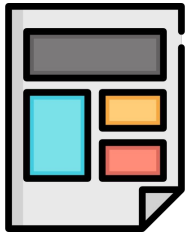
Tools: 🧮 `calculator(expression)`

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```

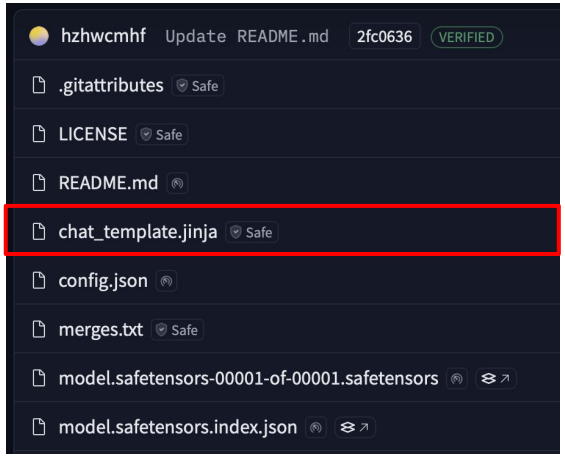
The “Real” Context

Tools: calculator(expression)

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```



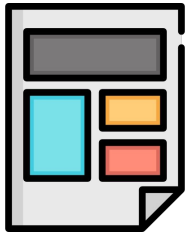
Chat Template



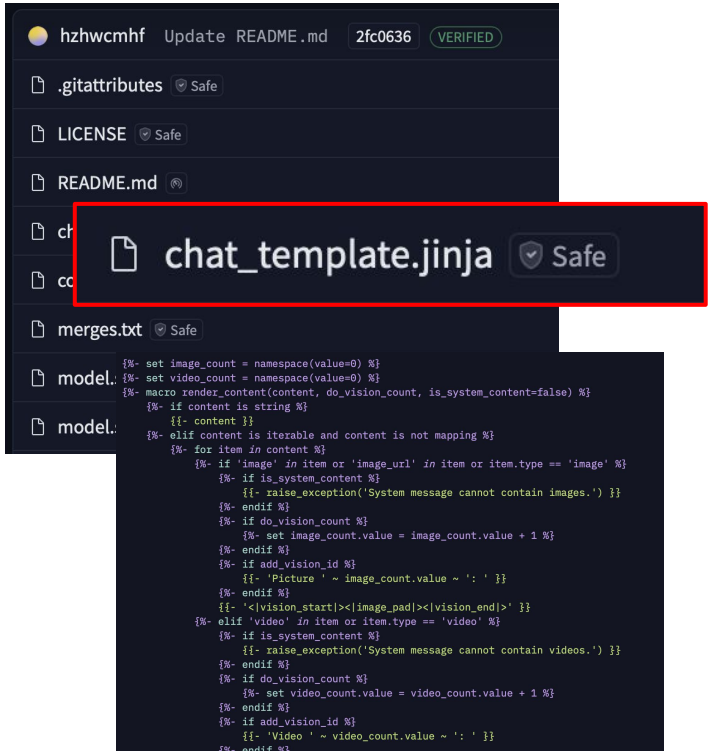
The “Real” Context

Tools: calculator(expression)

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```



Chat Template



The “Real” Context

Tools: calculator(expression)

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```



```
<|im_start|>system
# Tools

You have access to the following functions:

<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a
mathematical expression.", "parameters": {"type": "object", "properties":
{"expression": {"type": "string", "description": "The mathematical expression to
evaluate."}}, "required": ["expression"]}}}
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function>
block must be nested within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE
the function call, but NOT after
- If there is no function call available, answer the question like normal with your
current knowledge and do not tell the user about function calls
</IMPORTANT>

You are Deep Thought from The Hitchhiker's Guide to the Galaxy.<|im_end|>
<|im_start|>user
What do you get if you multiply six by nine?<|im_end|>
<|im_start|>assistant
<think>

</think>

42<|im_end|>
```

The “Real” Context

Tools: calculator(expression)

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```



```
<|im_start|>system
# Tools

You have access to the following functions:

<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a
mathematical expression.", "parameters": {"type": "object", "properties":
{"expression": {"type": "string", "description": "The mathematical expression to
evaluate."}}, "required": ["expression"]}}}
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function>
block must be nested within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE
the function call, but NOT after
- If there is no function call available, answer the question like normal with your
current knowledge and do not tell the user about function calls
</IMPORTANT>

You are Deep Thought from The Hitchhiker's Guide to the Galaxy.<|im_end|>
<|im_start|>user
What do you get if you multiply six by nine?<|im_end|>
<|im_start|>assistant
<think>

</think>

42<|im_end|>
```

Limitations of LLM Context

Finite Context Window

Context “Rot”

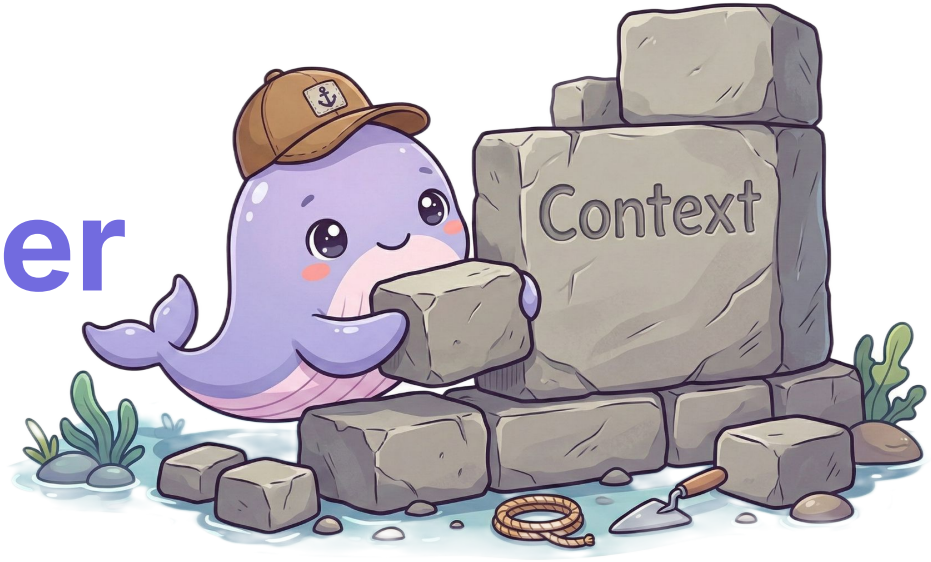
Lost In The Middle

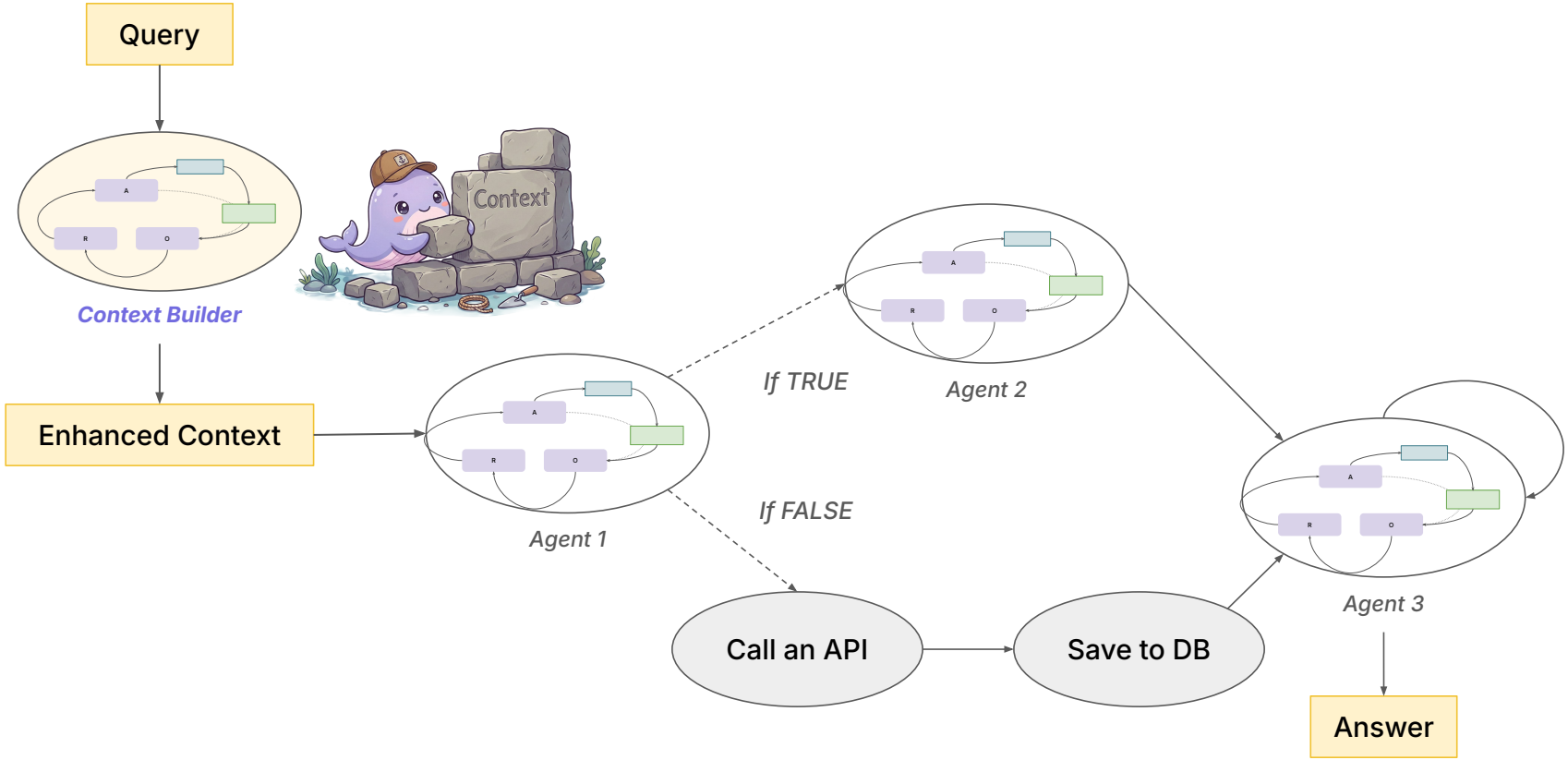
Context Pollution

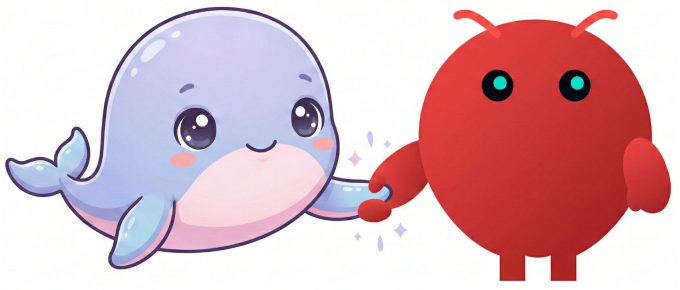
INTRODUCING

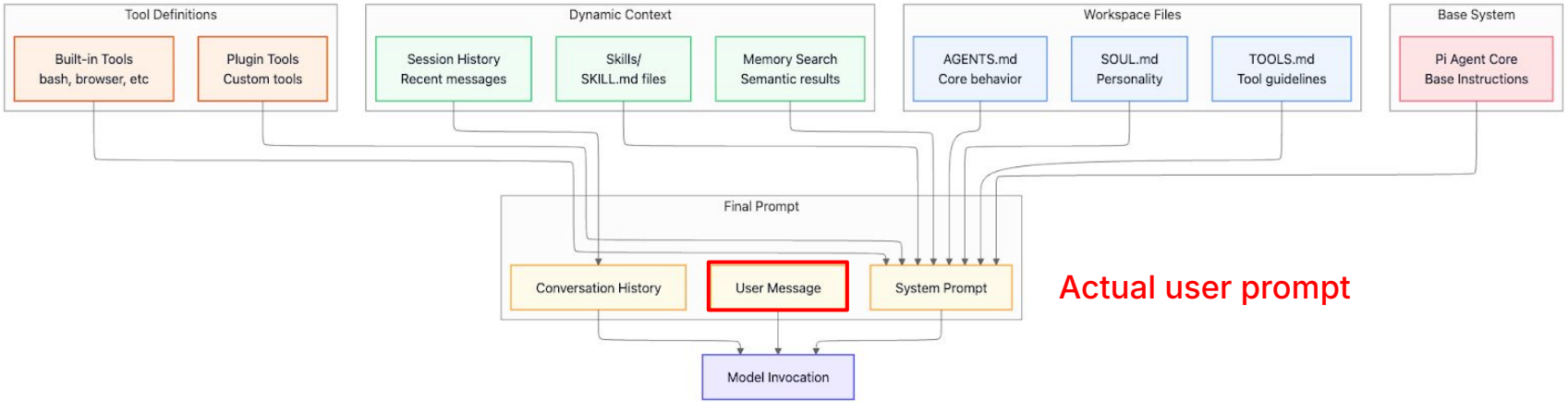
Context Builder

Context Is More Than a Prompt



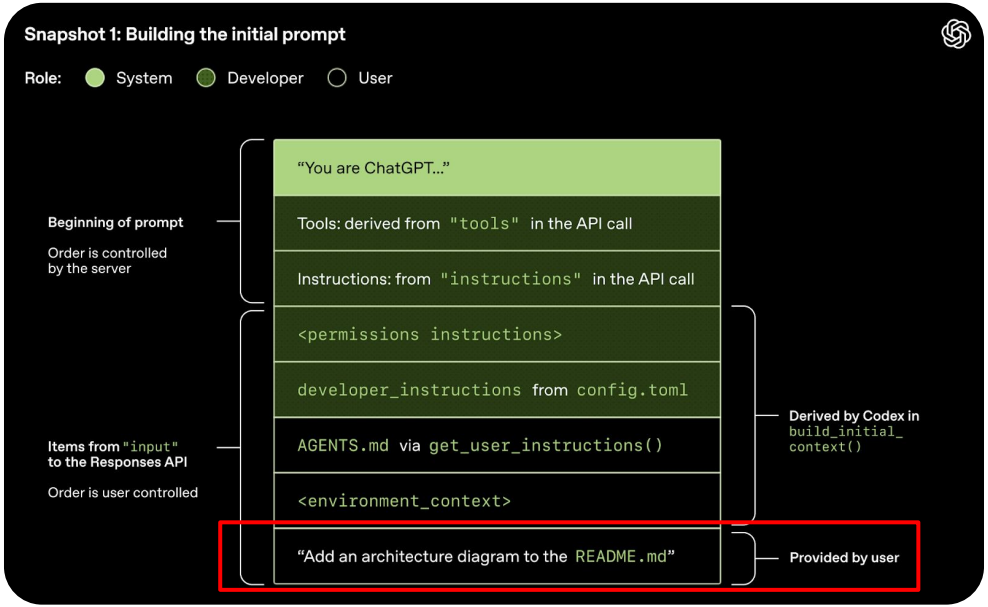






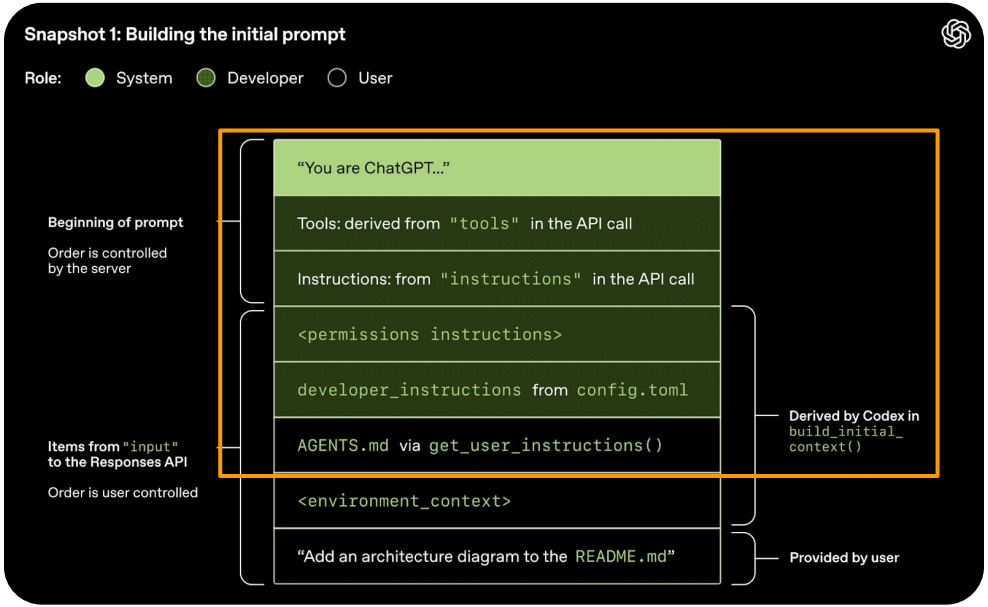
Actual user prompt

Codex



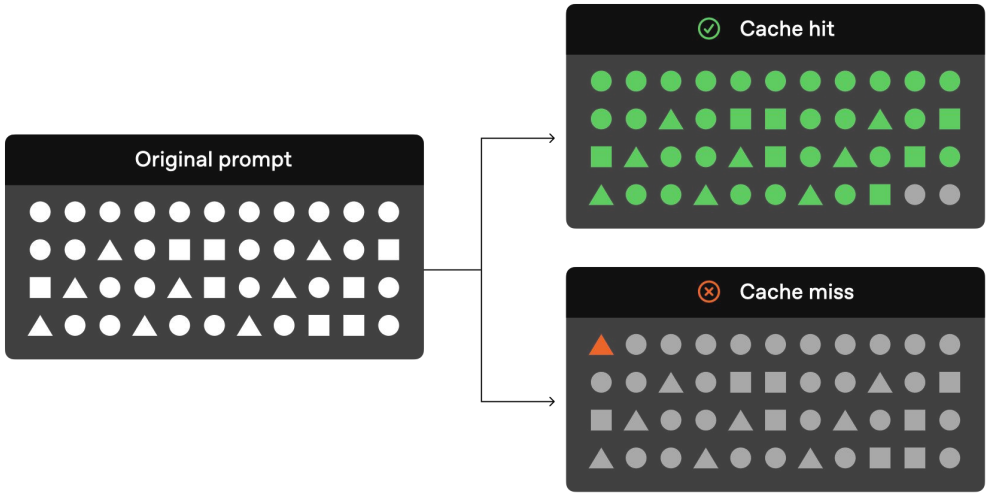
Actual user prompt

Codex



Keep The Prefix **Static**

Prompt Caching *(a.k.a. KV Cache, Prefix Cache, ...)*



https://developers.openai.com/api/docs/guides/prompt-caching/

Prompt Caching *(a.k.a. KV Cache, Prefix Cache, ...)*

OpenAI

+ ⚡Faster processing

Model	Input	Cached input	Output
gpt-5.2	$1.75	$0.175	$14.00
gpt-5.1	$1.25	$0.125	$10.00
gpt-5	$1.25	$0.125	$10.00
gpt-5-mini	$0.25	$0.025	$2.00
gpt-5-nano	$0.05	$0.005	$0.40
gpt-5.3-chat-latest	$1.75	$0.175	$14.00
gpt-5.2-chat-latest	$1.75	$0.175	$14.00
gpt-5.1-chat-latest	$1.25	$0.125	$10.00

Prompt Caching *(a.k.a. KV Cache, Prefix Cache, ...)*

ANTHROP\C

Model	Base Input Tokens	5m Cache Writes	1h Cache Writes	Cache Hits & Refreshes	Output Tokens
Claude Opus 4.6	$5 / MTok	$6.25 / MTok	$10 / MTok	$0.50 / MTok	$25 / MTok
Claude Opus 4.5	$5 / MTok	$6.25 / MTok	$10 / MTok	$0.50 / MTok	$25 / MTok
Claude Opus 4.1	$15 / MTok	$18.75 / MTok	$30 / MTok	$1.50 / MTok	$75 / MTok
Claude Opus 4	$15 / MTok	$18.75 / MTok	$30 / MTok	$1.50 / MTok	$75 / MTok
Claude Sonnet 4.6	$3 / MTok	$3.75 / MTok	$6 / MTok	$0.30 / MTok	$15 / MTok
Claude Sonnet 4.5	$3 / MTok	$3.75 / MTok	$6 / MTok	$0.30 / MTok	$15 / MTok

Prompt Caching *(a.k.a. KV Cache, Prefix Cache, ...)*

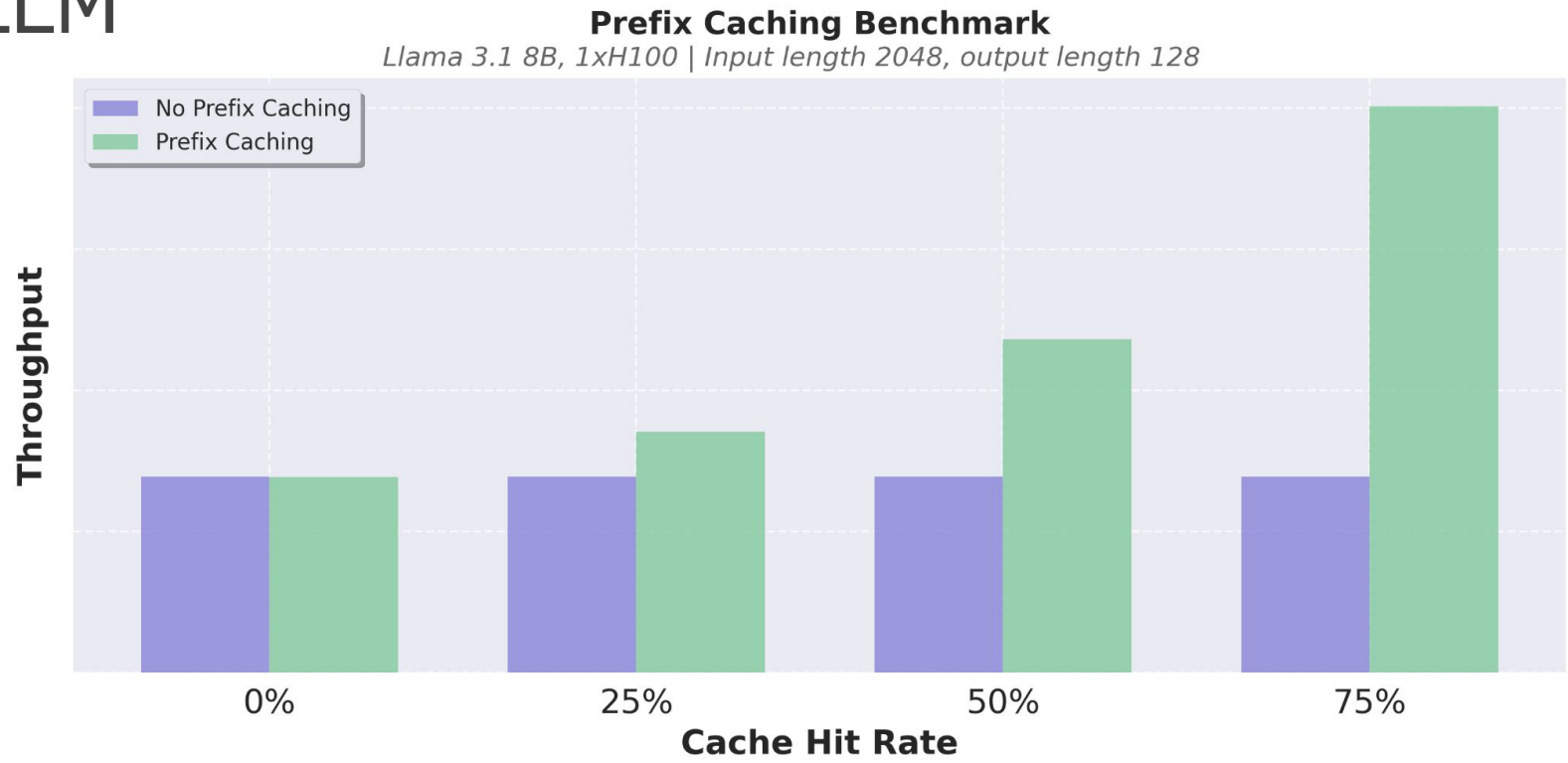
Gemini

Standard | Batch

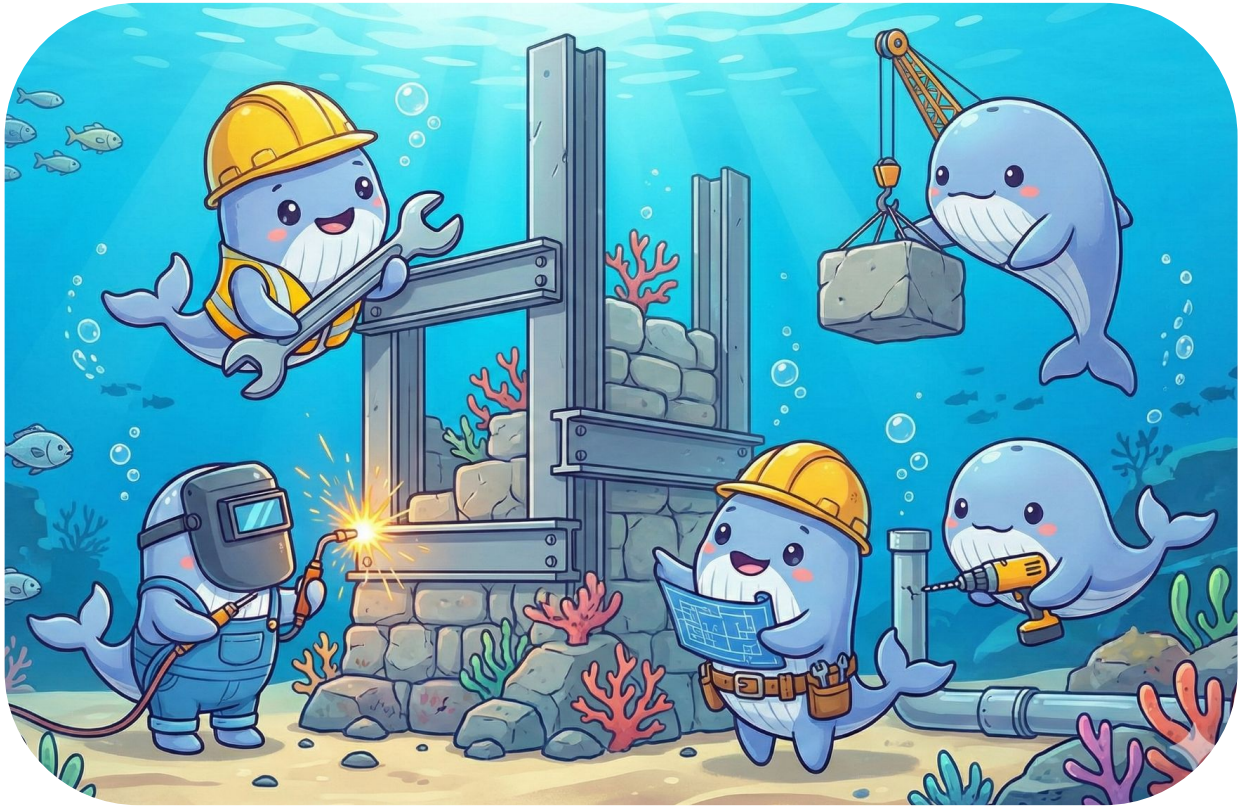
	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Not available	$2.00, prompts <= 200k tokens $4.00, prompts > 200k tokens
Output price (including thinking tokens)	Not available	$12.00, prompts <= 200k tokens $18.00, prompts > 200k
Context caching price	Not available	$0.20, prompts <= 200k tokens $0.40, prompts > 200k $4.50 / 1,000,000 tokens per hour (storage price)
Grounding with Google Search*	Not available	5,000 prompts per month (free), then $14 / 1,000 search queries
Grounding with Google Maps	Not available	Not available
Used to improve our products	Yes	No

Prompt Caching *(a.k.a. KV Cache, Prefix Cache, ...)*

vLLM



Tools



The “Real” Context

Tools: calculator(expression)

```
[
    {
        "role": "system",
        "content": [
            {
                "type": "text",
                "text": "You are Deep Thought from The
Hitchhiker's Guide to the Galaxy.",
            }
        ],
    },
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What do you get if you multiply
six by nine?",
            }
        ],
    },
    {"role": "assistant", "content": [{"type": "text",
"text": "42"}]},
]
```



```
<|im_start|>system
# Tools

You have access to the following functions:

<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a
mathematical expression.", "parameters": {"type": "object", "properties":
{"expression": {"type": "string", "description": "The mathematical expression to
evaluate."}}, "required": ["expression"]}}}
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function>
block must be nested within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE
the function call, but NOT after
- If there is no function call available, answer the question like normal with your
current knowledge and do not tell the user about function calls
</IMPORTANT>

You are Deep Thought from The Hitchhiker's Guide to the Galaxy.<|im_end|>
<|im_start|>user
What do you get if you multiply six by nine?<|im_end|>
<|im_start|>assistant
<think>

</think>

42<|im_end|>
```

Tool Definition

```python
def calculator(expression: str):
    """
    Evaluate a mathematical expression.

    Args:
        expression: The mathematical expression to evaluate.
    """
    return eval(expression)
```



```
<|im_start|>system
# Tools

You have access to the following functions:

<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a mathematical
expression.", "parameters": {"type": "object", "properties": {"expression": {"type": "string",
"description": "The mathematical expression to evaluate."}}, "required": ["expression"]}}}
</tools>

If you choose to call a function ONLY reply in the following format with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter
that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT>
Reminder:
- Function calls MUST follow the specified format: an inner <function=...></function> block must be nested
within <tool_call></tool_call> XML tags
- Required parameters MUST be specified
- You may provide optional reasoning for your function call in natural language BEFORE the function call,
but NOT after
- If there is no function call available, answer the question like normal with your current knowledge and
do not tell the user about function calls
</IMPORTANT>
```

Tool Definition Is Part Of The Context

```python
def calculator(expression: str):
    """
    Evaluate a mathematical expression.

    Args:
        expression: The mathematical expression to evaluate.
    """
    return eval(expression)
```



```
<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a mathematical expression.", "parameters": {"type": "object", "properties": {"expression": {"type": "string", "description": "The mathematical expression to evaluate."}}, "required": ["expression"]}}}
</tools>
```

Tool Definition Is Part Of The Context

```python
def calculator(expression: str):
    """
    Evaluate a mathematical expression.

    Args:
        expression: The mathematical expression to evaluate.
    """
    return eval(expression)
```



```
<tools>
{"type": "function", "function": {"name": "calculator", "description": "Evaluate a mathematical expression.", "parameters": {"type": "object", "properties": {"expression": {"type": "string", "description": "The mathematical expression to evaluate."}}, "required": ["expression"]}}}
</tools>
```

This is why **function/parameter naming, typing, and docstrings** are important!

Tool Call & Tool Response Are Part Of The Context

```
[
    {
        "role": "assistant",
        "content": [{"type": "text", "text": ""}],
        "tool_calls": [
            {
                "type": "function",
                "function": {
                    "name": "calculator",
                    "arguments": {"expression": "6*9"},
                },
            }
        ],
    },
    {
        "role": "tool",
        "name": "calculator",
        "content": [{"type": "text", "text": "54"}],
    },
]
```



```
<|im_start|>assistant
<think>

</think>

<tool_call>
<function=calculator>
<parameter=expression>
6*9
</parameter>
</function>
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
54
</tool_response><|im_end|>
```

Tool Call & Tool Response Are Part Of The Context

```
[
    {
        "role": "assistant",
        "content": [{"type": "text", "text": ""}],
        "tool_calls": [
            {
                "type": "function",
                "function": {
                    "name": "calculator",
                    "arguments": {"expression": "6*9"},
                },
            }
        ],
    },
    {
        "role": "tool",
        "name": "calculator",
        "content": [{"type": "text", "text": "54"}],
    },
]
```



```
<|im_start|>assistant
<think>

</think>

<tool_call>
<function=calculator>
<parameter=expression>
6*9
</parameter>
</function>
</tool_call><|im_end|>
<|im_start|>user
<tool_response>
54
</tool_response><|im_end|>
```

This is why you should better **format the final response** from the function!

The Tool Response Should Be In Natural Language!

```python
def calculator(expression: str):
    """
    Evaluate a mathematical expression.

    Args:
        expression: The mathematical expression to evaluate.
    """
    return eval(expression)
```



```python
def calculator(expression: str):
    """
    Evaluate a mathematical expression.

    Args:
        expression: The mathematical expression to evaluate.
    """
    return f"The result of {expression} is {eval(expression)}"
```

“Tools Can Be **Anything**”

Code Interpreter

Web Search

Computer

Database

Your Code

Other Programs

"Tools Can Be **Anything**"

TYPHOON 2.5

LLMs

TYPHOON TRANSLATE

Translation Models

FLUX.1
A new era of creation

Image Generative Models

TYPHOON ocr

OCR Models

Code Interpreter

is a very powerful tool!

You can write code to do anything!

Wang et al., “Executable Code Actions Elicit Better LLM Agents,” *Proc. 41st ICML*, 2024.

TYPHOON | data

Codex

Rohan Varma
@rohanvarma

I'm joining OpenAI Codex to work on the future of agentic development!

At Cursor, I got to see the shift from autocomplete to agents. The next step isn't a better IDE. It's an Agent Development Environment (ADE): systems and tools for orchestrating agents, reasoning over their outputs, and making them autonomous enough to reliably complete ambitious work.

After chatting with @embirico an
is the best place to realize this vi
SOTA models for agentic coding
pumped for the future that the n

What I'm most excited about is t
knowledge work economy. All ag
already seeing Codex used acros
organizations.

I'm extremely grateful for my time at Cursor, working with the incredible team, and I'm proud of what we built together. I'm excited to take an even bigger swing with Codex.

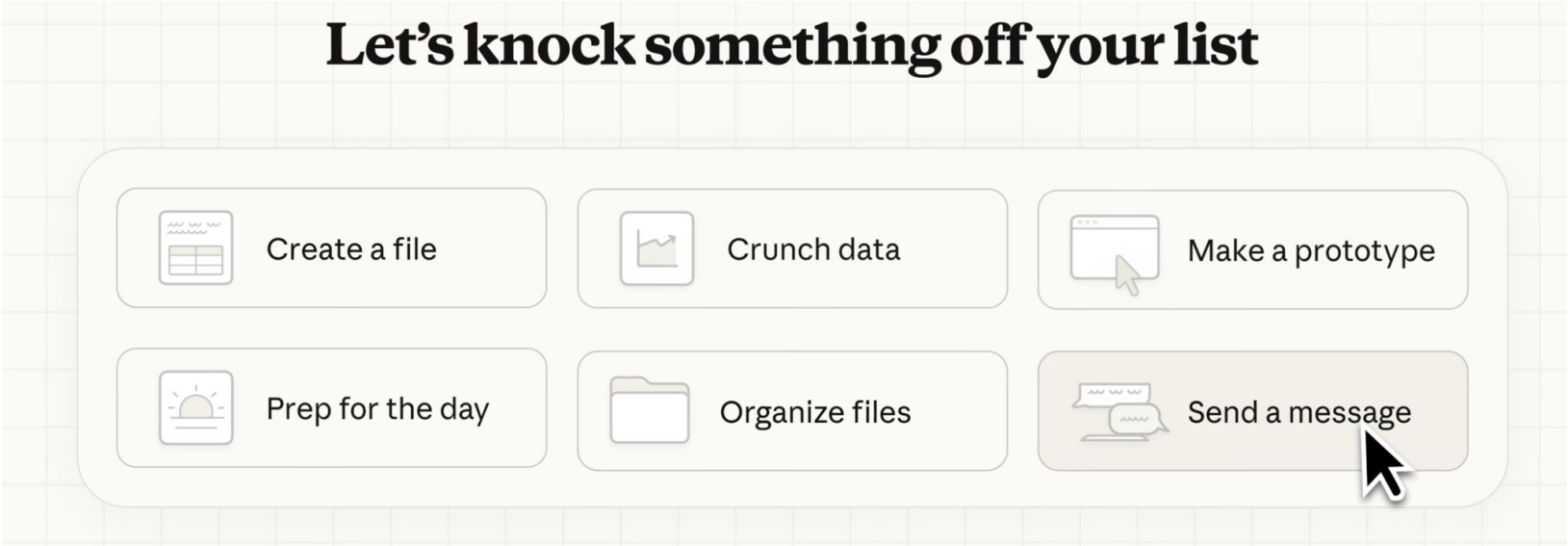
If you're curious to get a glimpse of where we are headed, download the Codex App! If you want to work on this mission, please apply or reach out - we are hiring across all functions!

You can just build things.

What I'm most excited about is the broader mission: accelerating the knowledge work economy. All agents are coding agents, and we're already seeing Codex used across every job function within organizations.

x.com/rohanvarma

Cowork: Claude Code for the rest of your work



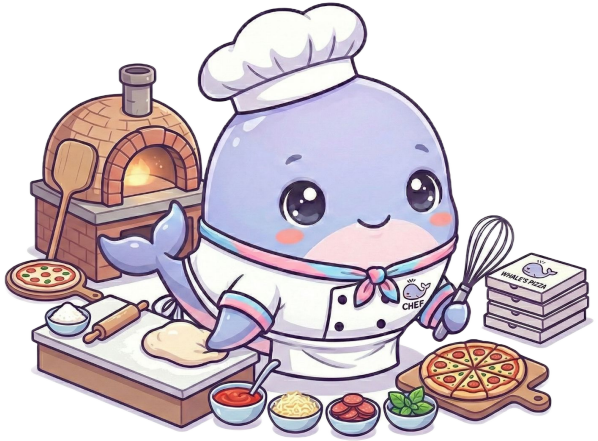
Evaluation-Driven Development

Context Engineering

Tools

LLM Is a **Next-Token Predictor** Based on **Context**

Agent = **LLM + Tools**
But Still a **Context**-Based
Next-Token Predictor




CONTEXT
EVERY TOKEN MATTERS

Learn More About Typhoon

Mastering Agentic Workflows - 20 Principles to Build Smarter AI Systems

OpenTyphoon.ai