

nanoagents: a worklog



Greg Brockman  
@gdb



2025 is the year of agents.

12:48 AM · 17 May 2025

Microsoft sees AI agents shaking up org charts, eliminating traditional functions

News / Technology / News / IBM replaces 200 HR roles with AI agents as part of automation push

News
Apr 28, 2025 • 4 mins
Artificial Intelligence

As companies inc
marketing, and er
orchestration of r

05-16-2025 | APPLIED AI

OpenAI launches Codex, an AI agent for coding

The research preview of Codex offers powerful code-writing, debugging, and collaboration features, marking a major step in OpenAI’s push to automate software development.


SHARE 

Upgrade

NVIDIA

accuracy and personalized user experiences

May 15, 2025

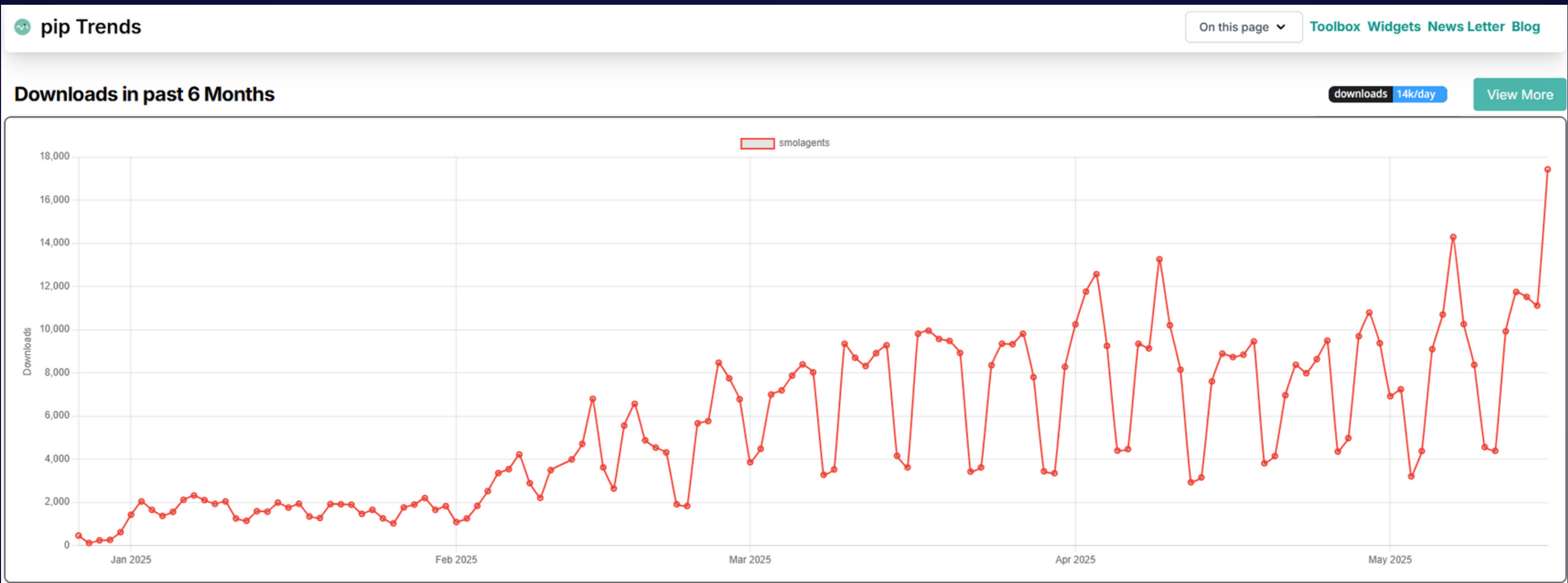
 Share


ents as

Agents Across

inference costs, and faster

with the integration of
that enhances model



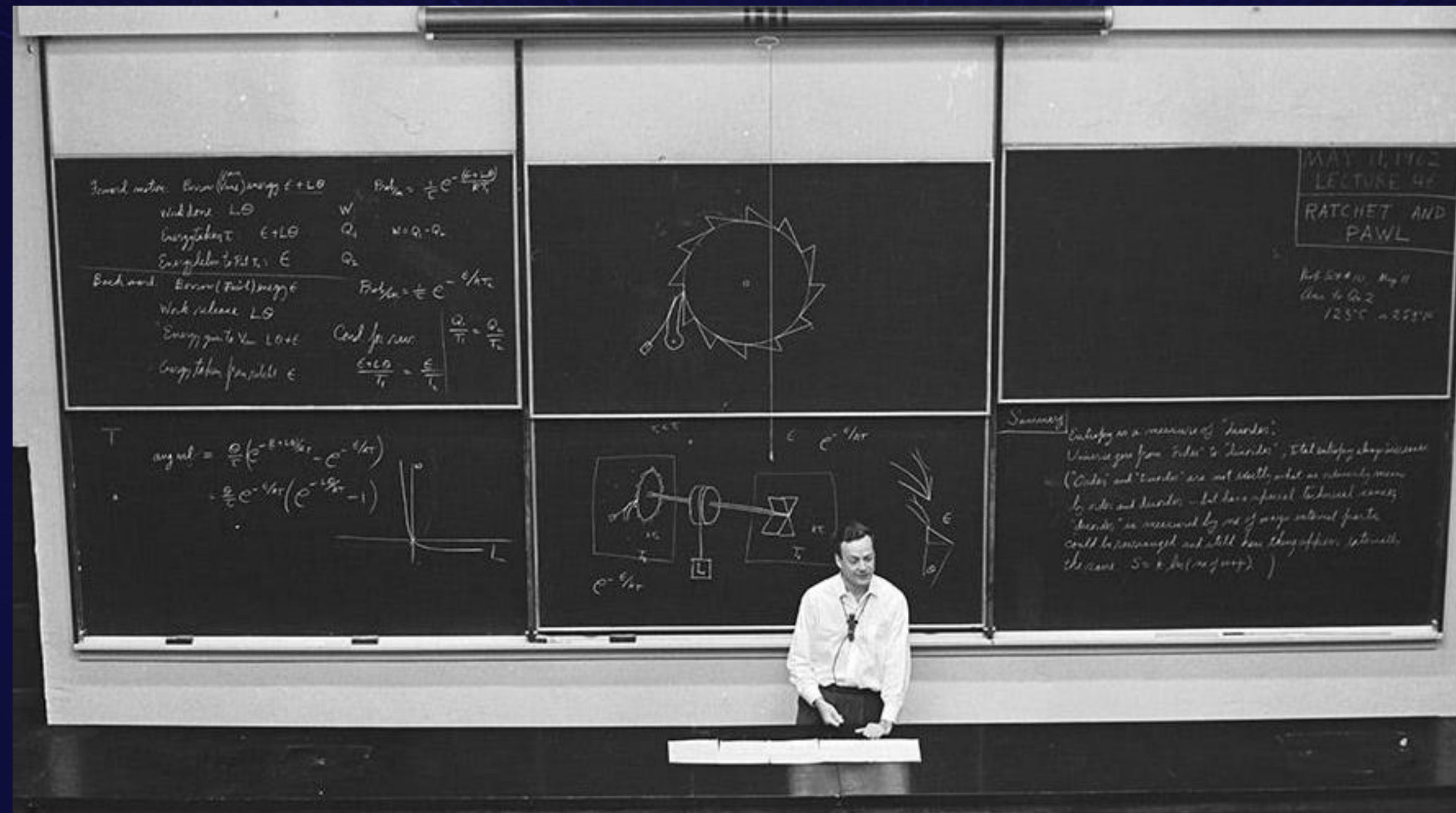


```
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.reasoning import ReasoningTools
from agno.tools.yfinance import YFinanceTools

reasoning_agent = Agent(
    model=Claude(id="claude-sonnet-4-20250514"),
    tools=[
        ReasoningTools(add_instructions=True),
        YFinanceTools(
            stock_price=True,
            analyst_recommendations=True,
            company_info=True,
            company_news=True),
    ],
    instructions="Use tables to display data.",
    markdown=True,
)
```


BUT
WHAT ARE
AGENTS





What I cannot create, I do not understand.

IN THIS SESSION

- **What are Agents?**
- **How do these agentic frameworks work?**
- **How to build a minimal agentic framework?**

IN THIS SESSION

[Help](#)[Sponsors](#)[Log in](#)[Register](#)

nanoagents 0.1.0

```
pip install nanoagents
```

[Latest version](#)

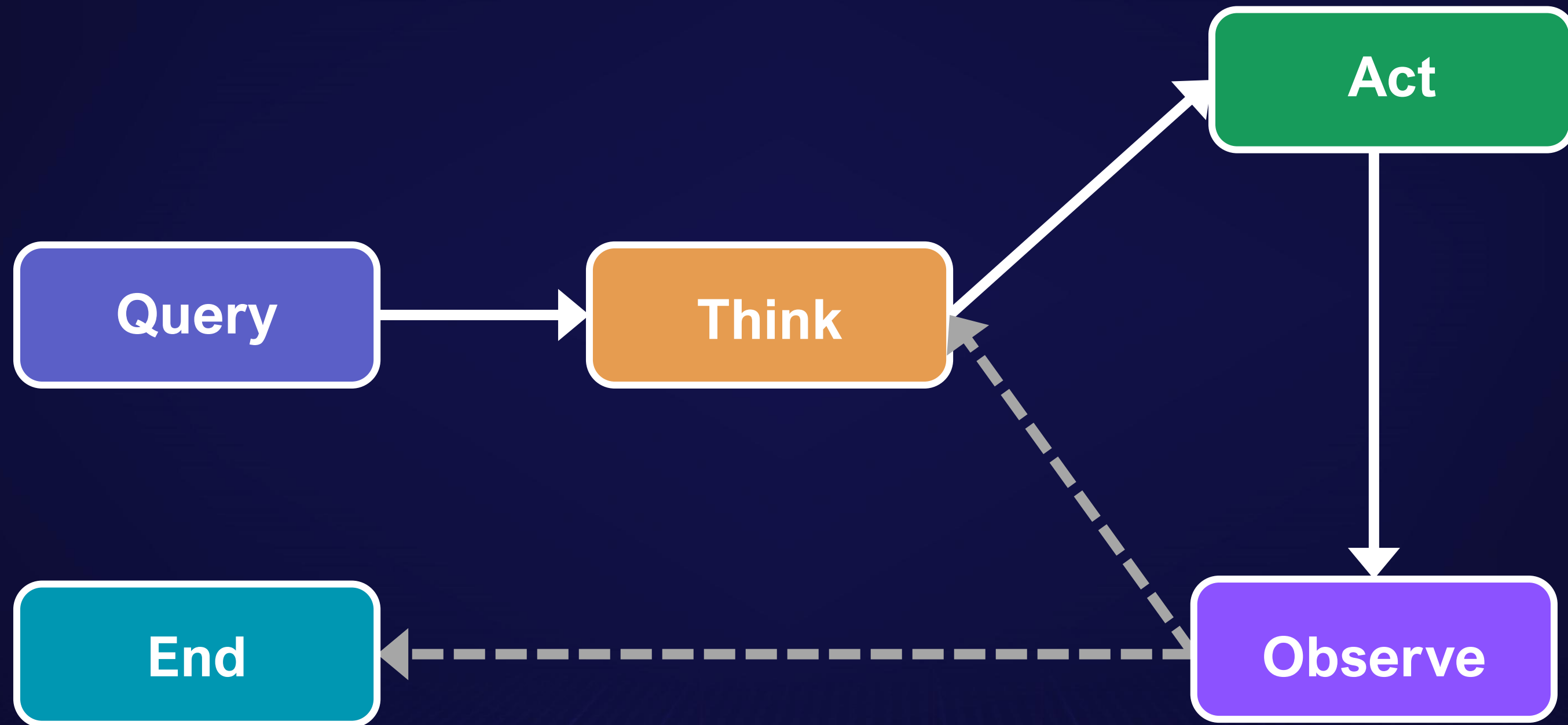
Released: Feb 26, 2025

A nano, minimalistic, and lightweight library for building ReACT-based AI agents that use tools to solve tasks.

What are AI Agents?

Type	Reactive	Tool Use	Reasoning	Planning	Proactivity
RAG Chatbot	✓	✗	✗	✗	✗
Tool-Augmented Chatbot	✓	✓	✗	✗	✗
Agentic AI	✓	✓	✓	✓	✓

How do Agents Work?



Components

Agent

Tool Registry

Tool 1

Tool 2

Tool 3

Steps

Thought

Action

Action Input

Observation

Response Parser

Prompt Formatter

Tool Registry

Tool 1

Tool 2

Tool 3

```
class ToolRegistry:
    """Manages tool collection."""

    def __init__(self): self.tools: Dict[str, Tool] = {}

    def register(self, tool: Tool): self.tools[tool.name] = tool

    def get(self, name: str) -> Optional[Tool]: return self.tools.get(name)

    def list(self) -> str: return "\n\n".join(map(str, self.tools.values()))

    def decorator(self, name: str, desc: str = None):
        def wrap(func):
            self.register(Tool(name, func, desc))
            return func

        return wrap
```


Tool Registry

Tool 1

Tool 2

Tool 3

```
class Tool:
    """Represents an agent tool."""

    def __init__(self, name: str, func: Callable, desc: str = None):
        self.name = name
        self.func = func
        self.desc = desc or func.__doc__.strip() or f"{name} tool"
        self.schema = _parse_schema(func)

    def execute(self, **kwargs):
        return self.func(**kwargs)

    def __str__(self):
        return f"Tool: {self.name}\nDesc: {self.desc}\nArgs: {self.schema}"
```


Steps

Thought

Action

Action Input

Observation

```
class Step:
    """Single reasoning step."""

    def __init__(self, thought: str, index: int = 1):
        self.index = index
        self.thought = thought
        self.action = None
        self.input = None
        self.obs = None

    def set_action(self, action: str, input: dict): self.action, self.input = action, input

    def set_obs(self, obs: str): self.obs = obs

    def __str__(self):
        return f"\n--- Iteration:{self.index} ---\nthought: {self.thought}\n" + \
            (f"action: {self.action}\n" if self.action else "") + \
            (f"action_input: {self.input}\n" if self.input else "") + \
            (f"observation: {self.obs}\n" if self.obs else "")
```


Response Parser

```
class ResponseParser:
    """Parses LLM responses."""

    @staticmethod
    def parse(text: str) -> dict:
        if match := re.search(r"```json([\s\S]*?)```", text, re.DOTALL):
            try:
                return json.loads(match.group(1))
            except json.JSONDecodeError:
                pass
        return {"thought": text, "action": "", "action_input": ""}
```

Prompt Formatter

```
class PromptFormatter:
    """Formats LLM prompts."""

    @staticmethod
    def format(task: str, tools: ToolRegistry, history: List[Step]) -> str:
        return f"""
You are an AI agent tasked with {task}. Use critical reasoning and these tools:











Tools:
{tools.list()}

Respond with JSON in markdown code block format:
{
  "thought": <your internal reasoning>,
  "action": <tool name>,
  "action_input": <params as JSON string>
  "final_answer": <when you have the final answer after a few iterations, provide it here>
}

History:
{"".join(map(str, history)) if history else "No history present, this is the first iteration"}

Important: Provide only valid JSON without any introduction, explanation, or additional text. No Preamble.
""".strip()
```


nanoagents

- ▼  nanoagents
 -  `__init__.py`
 -  `nanoagent.py`
- ▼  tests
 -  `__init__.py`
 -  `test.py`
-  `.gitignore`
-  `LICENSE`
-  `pyproject.toml`
-  `README.md`

Publishing Our Library



Type '/' to search projects



tnahddistud ▼

nanoagents 0.1.0

```
pip install nanoagents
```



[Latest version](#)

Released: Feb 26, 2025

API tokens

API tokens provide an alternative way to authenticate when uploading packages to PyPI. [Learn more about API tokens.](#)

Name	Scope	Created	Last used	
nanoag	All projects	Feb 26, 2025	Feb 26, 2025	<button>Options ▼</button>

[Add API token](#)

```
poetry config pypi-token.pypi <my-token>
```

```
poetry publish
```

Thank You!

