WORKSHOP

AI Agents in Python

Build agents that think, use tools, and take action

Uzair Ali

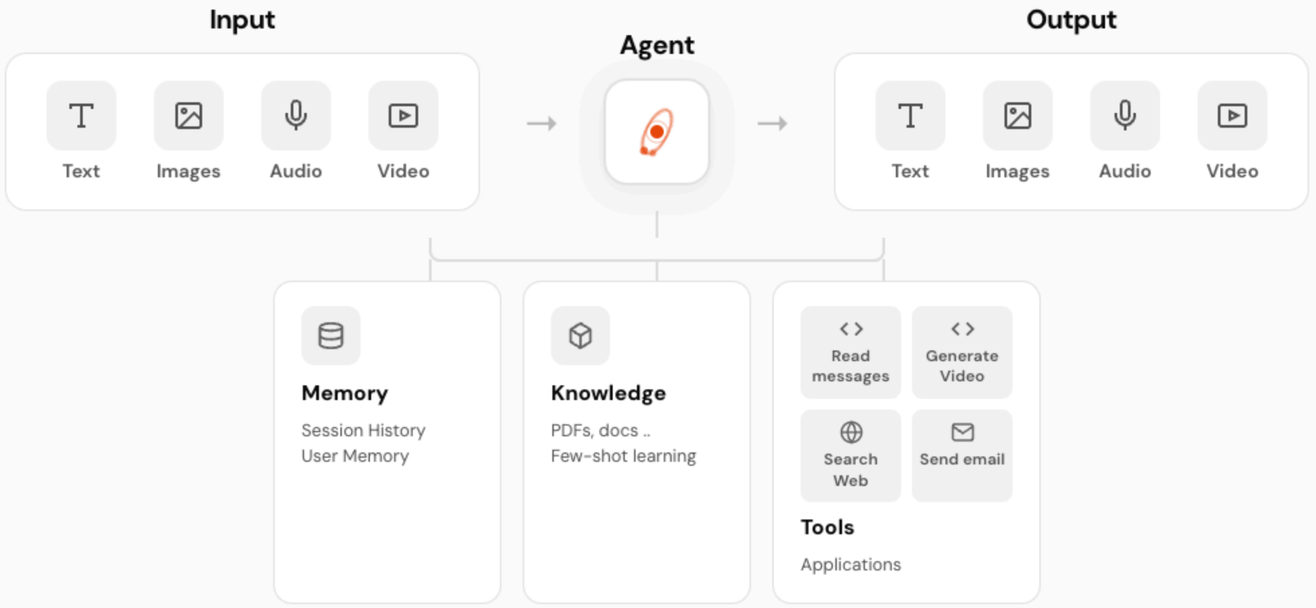
Scan for All Code Snippets

or visit: is.gd/ns_agents

What is an Agent?

- Software where the **LLM controls** the execution flow
- Agent = **LLM + tools + instructions**
- Analyzes information, makes decisions, acts **autonomously**

What is an Agent?



What is Agno?

- A **lightweight, open-source** Python framework for building AI agents
- **Model-agnostic** — works with OpenAI, Anthropic, Google, local models, and more
- Batteries included: **50+ built-in tools**, knowledge bases, memory, teams, workflows

⚡

Fast

Agent creation in ~2µs. No bloated abstractions.

🔌

Modular

Swap models, tools, storage with one line change.

🚀

Production-ready

Built-in FastAPI server, sessions, auth, monitoring.

Get Started

1. Install Agno
   `pip install agno`
2. Install a model provider
   `pip install anthropic` or `pip install openai` or `pip install google-genai`
3. Set your API key
   `export ANTHROPIC_API_KEY=sk-...`
4. Write 5 lines of Python. Done.

Your First Agent

```python
from agno.agent import Agent
from agno.models.anthropic import Claude

agent = Agent(
    model=Claude(id="claude-sonnet-4-5-20250929"),
    markdown=True,
)
agent.print_response("Write a haiku about programming")
```

3 lines of real code. That's your first agent.

See It in Action

Same agent, **3 new lines** — now it's a web app with a full chat UI

```python
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.os import AgentOS  # + new

# Same agent as before
agent = Agent(
    model=Claude(id="claude-sonnet-4-5-20250929"),
    markdown=True,
)

# Wrap it in AgentOS  + new
app = AgentOS(
    id="my-first-agent",
    agents=[agent],
).get_app()

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
$ python app.py
```

`os.agno.com`

- Chat UI & Playground
- Memory & Knowledge
- Team & Workflow Support
- Session Management
- Monitoring & Metrics
- One-command Deploy

What are Tools?

- LLMs are smart, but they **can't do things** in the real world on their own
- Tools are **Python functions** you give to the agent — so it *can* take action
- The agent **decides on its own** when to call a tool and what arguments to pass

🌐

Search the Web

Google, DuckDuckGo, Hacker News, Wikipedia

📦

Use APIs

Slack, GitHub, databases, any REST API

📝

Read & Write Files

PDFs, CSVs, images, code files

Think of tools as the agent's hands. The LLM is the brain — tools let it interact with the world.

Tools + Instructions in Code

- **Tools:** Python functions that can do... anything
- Search the web, generate images, send emails
- **Instructions:** Tell the agent how to behave

```python
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.duckduckgo import DuckDuckGoTools

agent = Agent(
    model=Claude(id="claude-sonnet-4-5-20250929"),
    tools=[DuckDuckGoTools()],
    instructions="Use tables to display data.",
    markdown=True,
)
agent.print_response(
    "Top 5 trending AI tools in 2025",
    stream=True
)
```

Write Your Own Tool

- Any Python function can become a **tool**
- The agent reads the **docstring** to know when and how to use it
- Call APIs, query databases, run calculations — **anything**

```python
import json, urllib.parse, urllib.request

def get_weather(city: str) -> str:
    """Get current weather for a city."""
    q = urllib.parse.quote(city)
    geo = json.loads(urllib.request.urlopen(
        f"https://geocoding-api.open-meteo.com/v1/search?name={q}&count=1"
    ).read())
    lat = geo["results"][0]["latitude"]
    lon = geo["results"][0]["longitude"]
    weather = json.loads(urllib.request.urlopen(
        f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}
        f"¤t=temperature_2m,wind_speed_10m"
    ).read())
    return json.dumps({"city": city, "current": weather["current"]})

agent = Agent(
    model=Claude(id="claude-sonnet-4-5-20250929"),
    tools=[get_weather],
)
agent.print_response("Weather in Tokyo?")
```

Give Your Agent Memory

- Without memory, every conversation **starts from scratch**
- With memory, the agent **remembers facts about you** across sessions
- Just add a **db** and set **enable_agentic_memory=True**

HOW IT WORKS

Tell it something
"I'm vegan"
→
Agent stores it
Automatically
→
Next session
Still knows you're vegan

```python
from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.memory import MemoryManager
from agno.db.sqlite import SqliteDb

db = SqliteDb(db_file="agents.db")

agent = Agent(
    model=Claude(id="claude-sonnet-4-5-20250929"),
    db=db,
    memory_manager=MemoryManager(
        model=Claude(id="claude-sonnet-4-5-20250929"),
        db=db,
    ),
    enable_agentic_memory=True,
    markdown=True,
)

# Session 1: tell it about yourself
agent.print_response(
    "I'm interested in AI stocks",
    user_id="uzair"
)

# Session 2: it still remembers
agent.print_response(
    "What stocks should I buy?",
    user_id="uzair"
)
```

What is MCP?

- **Model Context Protocol** — an open standard created by Anthropic
- Think of it like a **USB port for AI** — one standard plug, works with everything
- Instead of building custom tools, just **point your agent at an MCP server** and it gets access to all its tools

🔌

Plug & Play
Connect to any MCP server with one URL

🌍

Growing Ecosystem
GitHub, Slack, databases, docs — more every day

📷

No Extra Code
The server exposes tools. Your agent discovers them automatically.

Without MCP: you build each tool yourself. With MCP: someone else already built it.

MCP in Code: Playwright

SETUP (ONE TIME)

1. Install Node.js
   `brew install node`
2. Install Playwright MCP
   `npx @playwright/mcp@latest`
3. Install Agno with MCP
   `pip install agno[mcp]`

Your agent can now browse the web, click buttons, fill forms, and scrape pages.

```python
import asyncio
from agno.agent import Agent
from agno.models.openai import OpenAIChat
from agno.tools.mcp import MCPTools

agent = Agent(
    name="Browser Agent",
    model=OpenAIChat(id="gpt-4o-mini"),
    markdown=True,
    tools=[
        MCPTools(
            command="npx @playwright/mcp@latest",
            timeout_seconds=30,
        )
    ],
)

asyncio.run(agent.aprint_response(
    "Go to news.ycombinator.com and get the top post"
))
```

`command` = the same command you'd run in your terminal to start the MCP server

MCP in Code: Filesystem

ZERO SETUP. NO ACCOUNTS. NO TOKENS.

- Reads, searches, and analyzes **files on your machine**
- You choose **which directories** the agent can access
- Great for **codebase Q&A**, log analysis, doc summaries

The simplest MCP example — just point it at a folder and ask questions.

```python
import asyncio
from agno.agent import Agent
from agno.models.openai import OpenAIChat
from agno.tools.mcp import MCPTools

agent = Agent(
    name="File Explorer",
    model=OpenAIChat(id="gpt-4o-mini"),
    markdown=True,
    tools=[
        MCPTools(
            command="npx -y @modelcontextprotocol/server-filesystem .",
        )
    ],
)

asyncio.run(agent.aprint_response(
    "What files are here? Summarize the project."
))
```

MCP in Code: GitHub

GET A GITHUB PAT

1. Go to **github.com** → Profile → **Settings**
2. Scroll to **Developer settings**
3. **Personal access tokens** → **Fine-grained** → Generate
4. Name it, set expiry, grant **repo** access → copy token
5. Set it in terminal
   `export GITHUB_TOKEN=github_pat_...`
6. Install Agno with MCP
   `pip install agno[mcp]`

```python
import os, asyncio
from agno.agent import Agent
from agno.models.openai import OpenAIChat
from agno.tools.mcp import MCPTools, StreamableHTTPClientParams

agent = Agent(
    name="GitHub Agent",
    model=OpenAIChat(id="gpt-4o-mini"),
    markdown=True,
    tools=[
        MCPTools(
            transport="streamable-http",
            server_params=StreamableHTTPClientParams(
                url="https://api.githubcopilot.com/mcp/",
                headers={"Authorization": f"Bearer {os.getenv('GITHUB_TOKEN')}"},
            ),
        )
    ],
)

asyncio.run(agent.aprint_response(
    "List the top 5 open issues on agno-agi/agno"
))
```

Hosted by GitHub — just your PAT. No Docker, no npx, no local server.

LIVE DEMO

Let's build a Calorie Counter

Upload a photo of your food → get calories & macros back

Vision

Model sees your food photo

Markdown Tables

Clean calorie & macro breakdown

AgentOS

Chat UI where you upload images

Calorie Counter Agent

- Uses **vision** — the model sees images you upload
- Wrapped in **AgentOS** — chat UI handles image uploads for you
- **Markdown tables** — clean calories & macro breakdown

1. `python calorie_counter.py`
2. Open os.agno.com → upload a food photo → done

```python
from agno.agent import Agent
from agno.models.openai import OpenAIChat
from agno.os import AgentOS

agent = Agent(
    model=OpenAIChat(id="gpt-4o-mini"),
    instructions="""You are a calorie counting assistant.
When a user sends a food image, analyze it and respond with:
- Food name
- Estimated calories
- Protein, carbs, and fat in grams
Be concise. Use a table for the macros.""",
    markdown=True,
)

app = AgentOS(
    id="calorie-counter",
    agents=[agent],
).get_app()

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Where to Go Next

LEVEL UP

Knowledge & RAG

Give agents your data — PDFs, docs, databases

LEVEL UP

Reasoning

Chain-of-thought problem solving

LEVEL UP

Teams

Multiple agents collaborating together

LEVEL UP

Workflows

Deterministic multi-agent pipelines

GitHub

`agno-agi/agno`

Docs

`docs.agno.com`

Thank you!

Now go build something the world hasn't seen yet.

Follow me on X

@uzaxirr

Share what you build — tag **@uzaxirr**

Uzair Ali