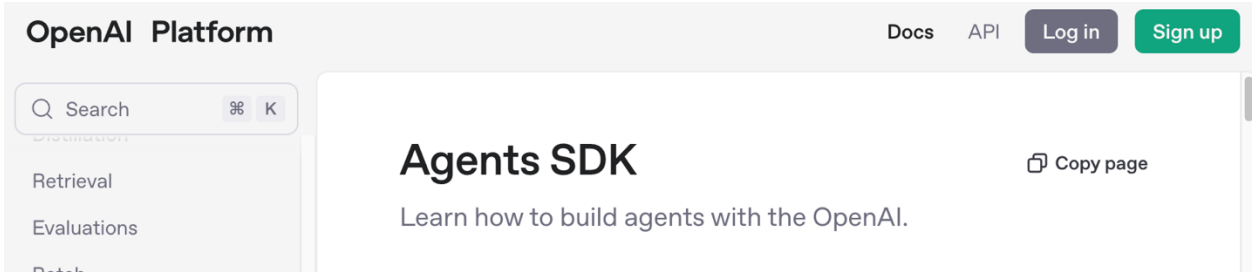
You think, saw, remembered, can

Creating or Cloning Agents

```python
from agents import Agent

# Basic
my_agent = Agent(
    name="my_agent",
    instructions="You are extremely friendly and helpful."
)

# Cloning with modifications
new_agent = my_agent.clone(
    name="my_new_agent",
    instructions="Now you're more formal."
)
```



https://platform.openai.com/docs/guides/agents-sdk#tools

Context

Sometimes you need to keep session data, credentials, or ephemeral state across multiple tool calls or steps. The Agents SDK allows a **context** object to be passed into the run.

- You create any Python class as context, or use a typed approach with `Agent(context_type=MyContext)`.
- Tools can then access that context, read data, and update it.

Using Context for User Sessions

```python
from agents.run_context import AgentContextWrapper
from agents.tool import function_tool

class MyContext:
  def __init__(self, user_id: str):
      self.user_id = user_id
      self.seen_messages = []

@function_tool
def greet_user(context: AgentContextWrapper[MyContext], greeting: str) -> str:
  user_id = context.agent_context.user_id
  return f"Hello {user_id}, you said: {greeting}"

agent = Agent(
  name="my_agent_with_context",
  context_type=MyContext,
  tools=[greet_user],
)

my_ctx = MyContext(user_id="alice")
result = await AgentRunner.run(
  agent,
  input=["Hi agent!"],
  context=my_ctx,
)
```