Loop agents

The `LoopAgent`

The `LoopAgent` is a workflow agent that executes its sub-agents in a loop (i.e. iteratively). It ***repeatedly runs* a sequence of agents** for a specified number of iterations or until a termination condition is met.

Use the `LoopAgent` when your workflow involves repetition or iterative refinement, such as like revising code.

Example

- You want to build an agent that can generate images of food, but sometimes when you want to generate a specific number of items (e.g. 5 bananas), it generates a different number of those items in the image (e.g. an image of 7 bananas). You have two tools: `Generate Image`, `Count Food Items`. Because you want to keep generating images until it either correctly generates the specified number of items, or after a certain number of iterations, you should build your agent using a `LoopAgent`.

As with other workflow agents, the `LoopAgent` is not powered by an LLM, and is thus deterministic in how it executes. That being said, workflow agents are only concerned only with their execution (i.e. in a loop), and not their internal logic; the tools or sub-agents of a workflow agent may or may not utilize LLMs.

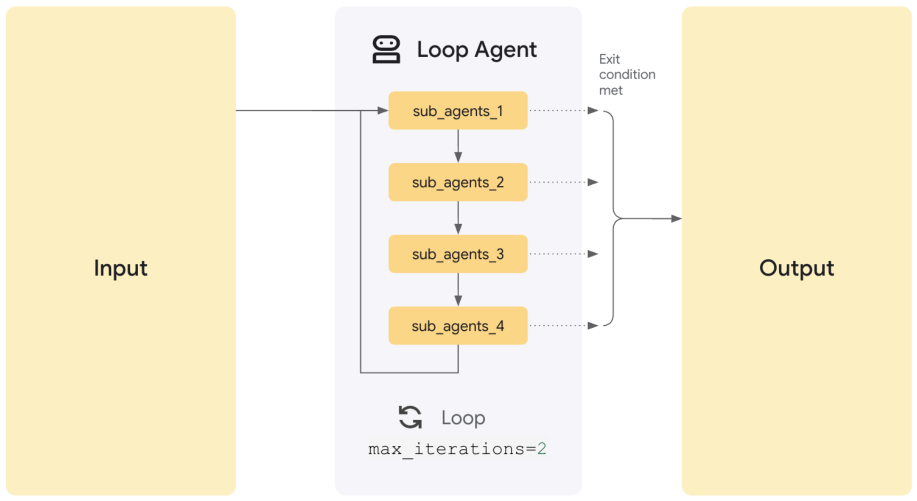
How it Works

When the `LoopAgent`'s `Run Async` method is called, it performs the following actions:

1. **Sub-Agent Execution:** It iterates through the Sub Agents list *in order*. For *each* sub-agent, it calls the agent's `Run Async` method.
2. **Termination Check:**

Crucially, the `LoopAgent` itself does *not* inherently decide when to stop looping. You *must* implement a termination mechanism to prevent infinite loops. Common strategies include:

- **Max Iterations**: Set a maximum number of iterations in the `LoopAgent`. **The loop will terminate after that many iterations**.
- **Escalation from sub-agent**: Design one or more sub-agents to evaluate a condition (e.g., "Is the document quality good enough?", "Has a consensus been reached?"). If the condition is met, the sub-agent can signal termination (e.g., by raising a custom event, setting a flag in a shared context, or returning a specific value).



Full Example: Iterative Document Improvement

Imagine a scenario where you want to iteratively improve a document:

- **Writer Agent:** An `LlmAgent` that generates or refines a draft on a topic.
- **Critic Agent:** An `LlmAgent` that critiques the draft, identifying areas for improvement.

```
LoopAgent(sub_agents=[WriterAgent, CriticAgent],
max_iterations=5)
```

In this setup, the `LoopAgent` would manage the iterative process. The `CriticAgent` could be **designed to return a "STOP" signal when the document reaches a satisfactory quality level**, preventing further iterations. Alternatively, the `max iterations` parameter could be used to limit the process to a fixed number of cycles, or external logic could be implemented to make stop decisions. The **loop would run at most five times**, ensuring the iterative refinement doesn't continue indefinitely.

Full Code

Python

```
# Part of agent.py --> Follow https://google.github.io/adk-
docs/get-started/quickstart/ to learn the setup

import asyncio
import os
from google.adk.agents import LoopAgent, LlmAgent, BaseAgent,
SequentialAgent
from google.genai import types
from google.adk.runners import InMemoryRunner
from google.adk.agents.invocation_context import
InvocationContext
from google.adk.tools.tool_context import ToolContext
from typing import AsyncGenerator, Optional
from google.adk.events import Event, EventActions

# --- Constants ---
APP_NAME = "doc_writing_app_v3" # New App Name
USER_ID = "dev_user_01"
SESSION_ID_BASE = "loop_exit_tool_session" # New Base Session ID
GEMINI_MODEL = "gemini-2.0-flash"
STATE_INITIAL_TOPIC = "initial_topic"

# --- State Keys ---
STATE_CURRENT_DOC = "current_document"
STATE_CRITICISM = "criticism"
# Define the exact phrase the Critic should use to signal
completion
COMPLETION_PHRASE = "No major issues found."

# --- Tool Definition ---
def exit_loop(tool_context: ToolContext):
  """Call this function ONLY when the critique indicates no
further changes are needed, signaling the iterative process
should end."""
  print(f"  [Tool Call] exit_loop triggered by
{tool_context.agent_name}")
  tool_context.actions.escalate = True
  # Return empty dict as tools should typically return JSON-
serializable output
  return {}

# --- Agent Definitions ---

# STEP 1: Initial Writer Agent (Runs ONCE at the beginning)
initial_writer_agent = LlmAgent(
    name="InitialWriterAgent",
    model=GEMINI_MODEL,
    include_contents='none',
    # MODIFIED Instruction: Ask for a slightly more developed
start
```

```
    instruction=f"""You are a Creative Writing Assistant tasked
with starting a story.
    Write the *first draft* of a short story (aim for 2-4
sentences).
    Base the content *only* on the topic provided below. Try to
introduce a specific element (like a character, a setting
detail, or a starting action) to make it engaging.
    Topic: {{initial_topic}}

    Output *only* the story/document text. Do not add
introductions or explanations.
""",
    description="Writes the initial document draft based on the
topic, aiming for some initial substance.",
    output_key=STATE_CURRENT_DOC
)

# STEP 2a: Critic Agent (Inside the Refinement Loop)
critic_agent_in_loop = LlmAgent(
    name="CriticAgent",
    model=GEMINI_MODEL,
    include_contents='none',
    # MODIFIED Instruction: More nuanced completion criteria,
look for clear improvement paths.
    instruction=f"""You are a Constructive Critic AI reviewing a
short document draft (typically 2-6 sentences). Your goal is
balanced feedback.

    **Document to Review:**
    ```
    {{current_document}}
    ```

    **Task:**
    Review the document for clarity, engagement, and basic
coherence according to the initial topic (if known).

    IF you identify 1-2 *clear and actionable* ways the document
could be improved to better capture the topic or enhance reader
engagement (e.g., "Needs a stronger opening sentence", "Clarify
the character's goal"):
    Provide these specific suggestions concisely. Output *only*
the critique text.

    ELSE IF the document is coherent, addresses the topic
adequately for its length, and has no glaring errors or obvious
omissions:
    Respond *exactly* with the phrase "{COMPLETION_PHRASE}" and
nothing else. It doesn't need to be perfect, just functionally
complete for this stage. Avoid suggesting purely subjective
stylistic preferences if the core is sound.

    Do not add explanations. Output only the critique OR the
exact completion phrase.
""",
    description="Reviews the current draft, providing critique
if clear improvements are needed, otherwise signals
```

```
completion.",
    output_key=STATE_CRITICISM
)


# STEP 2b: Refiner/Exiter Agent (Inside the Refinement Loop)
refiner_agent_in_loop = LlmAgent(
    name="RefinerAgent",
    model=GEMINI_MODEL,
    # Relies solely on state via placeholders
    include_contents='none',
    instruction=f"""You are a Creative Writing Assistant
refining a document based on feedback OR exiting the process.
    **Current Document:**
    ```
    {{current_document}}
    ```
    **Critique/Suggestions:**
    {{criticism}}

    **Task:**
    Analyze the 'Critique/Suggestions'.
    IF the critique is *exactly* "{COMPLETION_PHRASE}":
    You MUST call the 'exit_loop' function. Do not output any
text.
    ELSE (the critique contains actionable feedback):
    Carefully apply the suggestions to improve the 'Current
Document'. Output *only* the refined document text.

    Do not add explanations. Either output the refined document
OR call the exit_loop function.
""",
    description="Refines the document based on critique, or
calls exit_loop if critique indicates completion.",
    tools=[exit_loop], # Provide the exit_loop tool
    output_key=STATE_CURRENT_DOC # Overwrites
state['current_document'] with the refined version
)


# STEP 2: Refinement Loop Agent
refinement_loop = LoopAgent(
    name="RefinementLoop",
    # Agent order is crucial: Critique first, then Refine/Exit
    sub_agents=[
        critic_agent_in_loop,
        refiner_agent_in_loop,
    ],
    max_iterations=5 # Limit loops
)

# STEP 3: Overall Sequential Pipeline
# For ADK tools compatibility, the root agent must be named
`root_agent`
root_agent = SequentialAgent(
    name="IterativeWritingPipeline",
    sub_agents=[
```

```
        initial_writer_agent, # Run first to create initial doc
        refinement_loop       # Then run the critique/refine
loop
    ],
    description="Writes an initial document and then iteratively
refines it with critique using an exit tool."
)
```

Java

```java
import static
com.google.adk.agents.LlmAgent.IncludeContents.NONE;

import com.google.adk.agents.LlmAgent;
import com.google.adk.agents.LoopAgent;
import com.google.adk.agents.SequentialAgent;
import com.google.adk.events.Event;
import com.google.adk.runner.InMemoryRunner;
import com.google.adk.sessions.Session;
import com.google.adk.tools.Annotations.Schema;
import com.google.adk.tools.FunctionTool;
import com.google.adk.tools.ToolContext;
import com.google.genai.types.Content;
import com.google.genai.types.Part;
import io.reactivex.rxjava3.core.Flowable;
import java.util.Map;

public class LoopAgentExample {

  // --- Constants ---
  private static final String APP_NAME =
"IterativeWritingPipeline";
  private static final String USER_ID = "test_user_456";
  private static final String MODEL_NAME = "gemini-2.0-flash";

  // --- State Keys ---
  private static final String STATE_CURRENT_DOC =
"current_document";
  private static final String STATE_CRITICISM = "criticism";

  public static void main(String[] args) {
    LoopAgentExample loopAgentExample = new LoopAgentExample();
    loopAgentExample.runAgent("Write a document about a cat");
  }

  // --- Tool Definition ---
  @Schema(
      description =
          "Call this function ONLY when the critique indicates
no further changes are needed,"
              + " signaling the iterative process should end.")
  public static Map<String, Object> exitLoop(@Schema(name =
"toolContext") ToolContext toolContext) {
    System.out.printf("[Tool Call] exitLoop triggered by %s \n",
toolContext.agentName());
    toolContext.actions().setEscalate(true);
    //  Return empty dict as tools should typically return JSON-
serializable output
    return Map.of();
  }

  // --- Agent Definitions ---
  public void runAgent(String prompt) {
    // STEP 1: Initial Writer Agent (Runs ONCE at the beginning)
    LlmAgent initialWriterAgent =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("InitialWriterAgent")
            .description(
                "Writes the initial document draft based on the
topic, aiming for some initial"
                    + " substance.")
            .instruction(
                """
                    You are a Creative Writing Assistant tasked
with starting a story.
                    Write the *first draft* of a short story
(aim for 2-4 sentences).
                    Base the content *only* on the topic
provided below. Try to introduce a specific element (like a
character, a setting detail, or a starting action) to make it
engaging.

                    Output *only* the story/document text. Do
not add introductions or explanations.
                """)
            .outputKey(STATE_CURRENT_DOC)
            .includeContents(NONE)
            .build();

    // STEP 2a: Critic Agent (Inside the Refinement Loop)
    LlmAgent criticAgentInLoop =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("CriticAgent")
            .description(
                "Reviews the current draft, providing critique
if clear improvements are needed,"
                    + " otherwise signals completion.")
            .instruction(
                """
                    You are a Constructive Critic AI reviewing a
short document draft (typically 2-6 sentences). Your goal is
balanced feedback.

                    **Document to Review:**
                    ```
                    {{current_document}}
                    ```

                    **Task:**
                    Review the document for clarity, engagement,
and basic coherence according to the initial topic (if known).
```

```
                    IF you identify 1-2 *clear and actionable*
ways the document could be improved to better capture the topic
or enhance reader engagement (e.g., "Needs a stronger opening
sentence", "Clarify the character's goal"):
                    Provide these specific suggestions
concisely. Output *only* the critique text.

                    ELSE IF the document is coherent, addresses
the topic adequately for its length, and has no glaring errors
or obvious omissions:
                    Respond *exactly* with the phrase "No major
issues found." and nothing else. It doesn't need to be perfect,
just functionally complete for this stage. Avoid suggesting
purely subjective stylistic preferences if the core is sound.

                    Do not add explanations. Output only the
critique OR the exact completion phrase.
                    """)
            .outputKey(STATE_CRITICISM)
            .includeContents(NONE)
            .build();

    // STEP 2b: Refiner/Exiter Agent (Inside the Refinement
Loop)
    LlmAgent refinerAgentInLoop =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("RefinerAgent")
            .description(
                "Refines the document based on critique, or
calls exitLoop if critique indicates"
                    + " completion.")
            .instruction(
                """
                    You are a Creative Writing Assistant
refining a document based on feedback OR exiting the process.
                    **Current Document:**
                    ```
                    {{current_document}}
                    ```
                    **Critique/Suggestions:**
                    {{criticism}}

                    **Task:**
                    Analyze the 'Critique/Suggestions'.
                    IF the critique is *exactly* "No major
issues found.":
                    You MUST call the 'exitLoop' function. Do
not output any text.
                    ELSE (the critique contains actionable
feedback):
                    Carefully apply the suggestions to improve
the 'Current Document'. Output *only* the refined document text.

                    Do not add explanations. Either output the
refined document OR call the exitLoop function.
```

```
                """)
            .outputKey(STATE_CURRENT_DOC)
            .includeContents(NONE)
            .tools(FunctionTool.create(LoopAgentExample.class,
"exitLoop"))
            .build();

    // STEP 2: Refinement Loop Agent
    LoopAgent refinementLoop =
        LoopAgent.builder()
            .name("RefinementLoop")
            .description("Repeatedly refines the document with
critique and then exits.")
            .subAgents(criticAgentInLoop, refinerAgentInLoop)
            .maxIterations(5)
            .build();

    // STEP 3: Overall Sequential Pipeline
    SequentialAgent iterativeWriterAgent =
        SequentialAgent.builder()
            .name(APP_NAME)
            .description(
                "Writes an initial document and then iteratively
refines it with critique using an"
                    + " exit tool.")
            .subAgents(initialWriterAgent, refinementLoop)
            .build();

    // Create an InMemoryRunner
    InMemoryRunner runner = new
InMemoryRunner(iterativeWriterAgent, APP_NAME);
    // InMemoryRunner automatically creates a session service.
Create a session using the service
    Session session =
runner.sessionService().createSession(APP_NAME,
USER_ID).blockingGet();
    Content userMessage =
Content.fromParts(Part.fromText(prompt));

    // Run the agent
    Flowable<Event> eventStream = runner.runAsync(USER_ID,
session.id(), userMessage);

    // Stream event response
    eventStream.blockingForEach(
        event -> {
          if (event.finalResponse()) {
            System.out.println(event.stringifyContent());
          }
        });
  }
}
```