# RoundRobin: Round-Robin Task Execution in a Swarm

## Introduction

The `RoundRobinSwarm` class is designed to manage and execute tasks among multiple agents in a round-robin fashion. This approach ensures that each agent in a swarm receives an equal opportunity to execute tasks, which promotes fairness and efficiency in distributed systems. It is particularly useful in environments where collaborative, sequential task execution is needed among various agents.

## Conceptual Overview

### What is Round-Robin?

Round-robin is a scheduling technique commonly used in computing for managing processes in shared systems. It involves assigning a fixed time slot to each process and cycling through all processes in a circular order without prioritization. In the context of swarms of agents, this method ensures equitable distribution of tasks and resource usage among all agents.

### Application in Swarms

In swarms, `RoundRobinSwarm` utilizes the round-robin scheduling to manage tasks among agents like software components, autonomous robots, or virtual entities. This strategy is beneficial where tasks are interdependent or require sequential processing.

## Class Attributes

- `agents (List[Agent])`: List of agents participating in the swarm.
- `verbose (bool)`: Enables or disables detailed logging of swarm operations.
- `max_loops (int)`: Limits the number of times the swarm cycles through all agents.
- `index (int)`: Maintains the current position in the agent list to ensure round-robin execution.

## Methods

`__init__`

Initializes the swarm with the provided list of agents, verbosity setting, and operational parameters.

Parameters:
- `agents`: Optional list of agents in the swarm.
- `verbose`: Boolean flag for detailed logging.
- `max_loops`: Maximum number of execution cycles.
- `callback`: Optional function called after each loop.

### `run`

Executes a specified task across all agents in a round-robin manner, cycling through each agent repeatedly for the number of specified loops.

Conceptual Behavior:
- Distribute the task sequentially among all agents starting from the current index.
- Each agent processes the task and potentially modifies it or produces new output.

- After an agent completes its part of the task, the index moves to the next agent.
- This cycle continues until the specified maximum number of loops is completed.
- Optionally, a callback function can be invoked after each loop to handle intermediate results or perform additional actions.

## Examples

### Example 1: Load Balancing Among Servers

In this example, `RoundRobinSwarm` is used to distribute network requests evenly among a group of servers. This is common in scenarios where load balancing is crucial for maintaining system responsiveness and scalability.

```python
from swarms import Agent, RoundRobinSwarm
from swarm_models import OpenAIChat

# Initialize the LLM
llm = OpenAIChat()

# Define sales agents
sales_agent1 = Agent(
    agent_name="Sales Agent 1 - Automation Specialist",
    system_prompt="You're Sales Agent 1, your purpose is to generate sales for a company by focusing on the benefits of automating accounting processes!",
    agent_description="Generate sales by focusing on the benefits of automation!",
    llm=llm,
    max_loops=1,
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    context_length=1000,
)

sales_agent2 = Agent(
    agent_name="Sales Agent 2 - Cost Saving Specialist",
    system_prompt="You're Sales Agent 2, your purpose is to generate sales for a company by emphasizing the cost savings of using swarms of agents!",
    agent_description="Generate sales by emphasizing cost savings!",
    llm=llm,
    max_loops=1,
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    context_length=1000,
)

sales_agent3 = Agent(
    agent_name="Sales Agent 3 - Efficiency Specialist",
    system_prompt="You're Sales Agent 3, your purpose is to generate sales for a company by
```

```
highlighting the efficiency and accuracy of our swarms of agents in accounting processes!",
    agent_description="Generate sales by highlighting efficiency and accuracy!",
    llm=llm,
    max_loops=1,
    autosave=True,
    dashboard=False,
    verbose=True,
    streaming_on=True,
    context_length=1000,
)

# Initialize the swarm with sales agents
sales_swarm = RoundRobinSwarm(agents=[sales_agent1, sales_agent2, sales_agent3], verbose=True)

# Define a sales task
task = "Generate a sales email for an accountant firm executive to sell swarms of agents to automate their accounting processes."

# Distribute sales tasks to different agents
for _ in range(5):  # Repeat the task 5 times
    results = sales_swarm.run(task)
    print("Sales generated:", results)
```

## Conclusion

The RoundRobinSwarm class provides a robust and flexible framework for managing tasks among multiple agents in a fair and efficient manner. This class is especially useful in environments where tasks need to be distributed evenly among a group of agents, ensuring that all tasks are handled timely and effectively. Through the round-robin algorithm, each agent in the swarm is guaranteed an equal opportunity to contribute to the overall task, promoting efficiency and collaboration.