

# Introdução a Algoritmos

Disciplina: Programação aplicada à engenharia cartográfica

Maurício C. M. de Paulo - D.Sc.

11 de fevereiro de 2026

**Strands Agents** é um framework para:

- Construir agentes baseados em LLMs
- Orquestrar raciocínio, ferramentas e memória
- Criar aplicações de IA orientadas a tarefas

**Ideia central:** Um agente que pensa, decide e age de forma estruturada.

Um **agente** é um sistema que:

- Recebe um objetivo
- Observa o contexto
- Decide ações
- Executa ferramentas

**Diferença para um chatbot simples:**

- Chatbot: responde mensagens
- Agente: resolve tarefas

# Por que Strands Agents?

Sem um framework:

- Lógica de decisão fica espalhada
- Código difícil de manter
- Pouca reutilização

Strands fornece:

- Estrutura clara para agentes
- Separação de responsabilidades
- Orquestração padronizada

# Componentes de um Strands Agent

Um agente é composto por:

- **Modelo (LLM)**: raciocínio e decisões
- **Ferramentas (Tools)**: ações externas
- **Memória**: contexto e histórico
- **Objetivo**: o que deve ser resolvido

Cada componente tem um papel bem definido.

Ferramentas permitem ao agente:

- Executar código
- Consultar APIs
- Acessar bancos de dados
- Ler e escrever arquivos

**Importante:**

- O LLM decide *quando* usar a ferramenta
- O código define *o que* a ferramenta faz

# Memória do agente

A memória permite:

- Manter contexto da conversa
- Lembrar decisões anteriores
- Trabalhar em tarefas longas

Tipos comuns:

- Memória de curto prazo
- Memória persistente

Fluxo típico de um Strands Agent:

- 1 Recebe um objetivo
- 2 Analisa o contexto
- 3 Planeja ações
- 4 Executa ferramentas
- 5 Avalia o resultado

Esse ciclo pode se repetir até o objetivo ser atingido.

Strands Agents podem ser usados para:

- Assistentes inteligentes
- Automação de tarefas
- Análise de dados guiada por IA
- Agentes de suporte técnico

**Ideais para tarefas complexas e multi-etapas.**

- Strands Agents estruturam agentes de IA
- Integram LLMs, ferramentas e memória
- Facilitam aplicações orientadas a objetivos

**Mensagem-chave:** Strands não é apenas IA conversacional — é IA agente e operacional.

# Exemplo — Estrutura de um agente

**Ideia:** O agente recebe uma pergunta, decide o que fazer e executa ações.

```
class AgenteSimples:
    def __init__(self):
        self.memoria = {}

    def responder(self, pergunta: str) -> str:
        # Decide o que fazer com base na pergunta
        if self._eh_calculo(pergunta):
            return self._resolver_calculo(pergunta)
        else:
            return self._responder_texto(pergunta)
```

## Conceitos envolvidos:

- Agente encapsula lógica de decisão
- Não executa tudo diretamente

# Exemplo — Decisão e ferramentas

```
def _eh_calculo(self, pergunta: str) -> bool:
    return any(op in pergunta for op in ["+", "*", "/"])

def _resolver_calculo(self, pergunta: str) -> str:
    # Exemplo simplificado
    resultado = calcular(2, 3, "soma")
    registrar_log("Cálculo executado")
    return f"Resultado: {resultado}"

def _responder_texto(self, pergunta: str) -> str:
    resposta = buscar_informacao(pergunta)
    salvar_memoria("ultima_pergunta", pergunta)
    return resposta
```

## Fluxo do agente:

- 1 Analisa a pergunta
- 2 Decide usar (ou não) uma ferramenta
- 3 Retorna a resposta

**Objetivo:** Executar cálculos quando o agente precisar de resultados exatos.

```
def calcular(a: float, b: float, operacao: str) -> float:
    if operacao == "soma":
        return a + b
    elif operacao == "multiplicacao":
        return a * b
    elif operacao == "divisao":
        if b == 0:
            raise ValueError("Divisão por zero")
        return a / b
    else:
        raise ValueError("Operação inválida")
```

**Uso pelo agente:**

- Quando a pergunta envolver números
- Quando precisão for necessária

**Objetivo:** Simular uma busca de informações externas.

```
def buscar_informacao(tema: str) -> str:
    base_conhecimento = {
        "python": "Python é uma linguagem de programação.",
        "git": "Git é um sistema de controle de versão.",
        "github": "GitHub hospeda repositórios Git."
    }
    return base_conhecimento.get(
        tema.lower(),
        "Informação não encontrada."
    )
```

**Uso pelo agente:**

- Perguntas factuais
- Consultas repetíveis

**Objetivo:** Registrar ações executadas pelo agente.

```
def registrar_log(mensagem: str) -> None:
    with open("log_agente.txt", "a") as arquivo:
        arquivo.write(mensagem + "\n")
```

**Exemplos de uso:**

- Registrar decisões do agente
- Acompanhar execução

**Importante:** Ferramentas podem causar efeitos fora do agente.

**Objetivo:** Armazenar e recuperar informações entre interações.

```
memoria = {}

def salvar_memoria(chave: str, valor: str) -> None:
    memoria[chave] = valor

def ler_memoria(chave: str) -> str:
    return memoria.get(chave, "Nada armazenado")
```

**Uso pelo agente:**

- Lembrar perguntas anteriores
- Manter contexto

# Exercício — Strands Agent básico

**Objetivo do exercício:** Compreender como estruturar um agente usando Strands.

**Cenário:** Você deve criar um agente que ajude um usuário a:

- Receber uma pergunta
- Decidir se precisa usar uma ferramenta
- Retornar uma resposta adequada

**Tarefas:**

- 1 Defina o **objetivo** do agente
- 2 Liste quais **ferramentas** ele pode usar
- 3 Descreva como o agente decide usar uma ferramenta

**Ferramentas sugeridas:**

- Uma função para calcular valores numéricos
- Uma função para buscar informações simuladas

## Dicas:

- Pense no agente como um **resolvedor de tarefas**
- Separe claramente:
  - Raciocínio (LLM)
  - Ação (ferramentas)
- O agente não deve executar tudo sozinho

## Pergunta para reflexão:

- Em quais situações o agente deve usar ferramentas?
- Quando apenas responder com texto é suficiente?

## Desafio:

- Adicione **memória** ao agente
- Faça com que ele:
  - Lembre a última pergunta do usuário
  - Use essa informação na próxima resposta

**Objetivo pedagógico:** Entender a diferença entre respostas isoladas e agentes com contexto.