Mini curso CrewAI

youtube.com/@dcodecanal

O que é o CrewAI?

CrewAI é um framework projetado para orquestrar agentes de inteligência artificial (IA) autônomos em um ambiente de jogo de papéis, permitindo-lhes assumir papéis específicos, compartilhar objetivos e operar como uma unidade coesa. Essa abordagem tem como objetivo facilitar interações sofisticadas entre múltiplos agentes de IA, cada um com suas especializações, habilidades e objetivos, trabalhando juntos para alcançar um objetivo comum. O CrewAI é construído sobre a langchain, o que significa que ele pode utilizar todas as ferramentas públicas existentes da langchain, proporcionando uma base robusta para a criação de experiências interativas e colaborativas entre agentes.

Requisitos para instalação e utilização do CrewAI

Recomendado ter pelo menos 16gb de ram e um processador i5 (caso utilize I.A em ambiente local)

LM Studio ou Ollama para usar os modelos de I.A (caso utilize I.A em ambiente local)

Python em uma versão superior a 3.8

Pycharm (opcional)

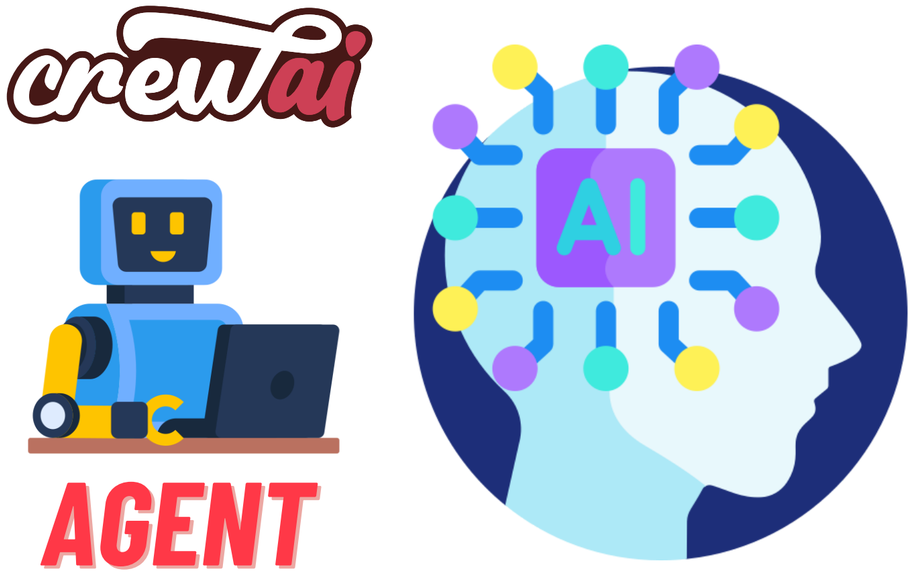

PC

1° criar um ambiente virtual no python pelo pycharm

2° instalar o CrewAI

```
pip install crewai
```

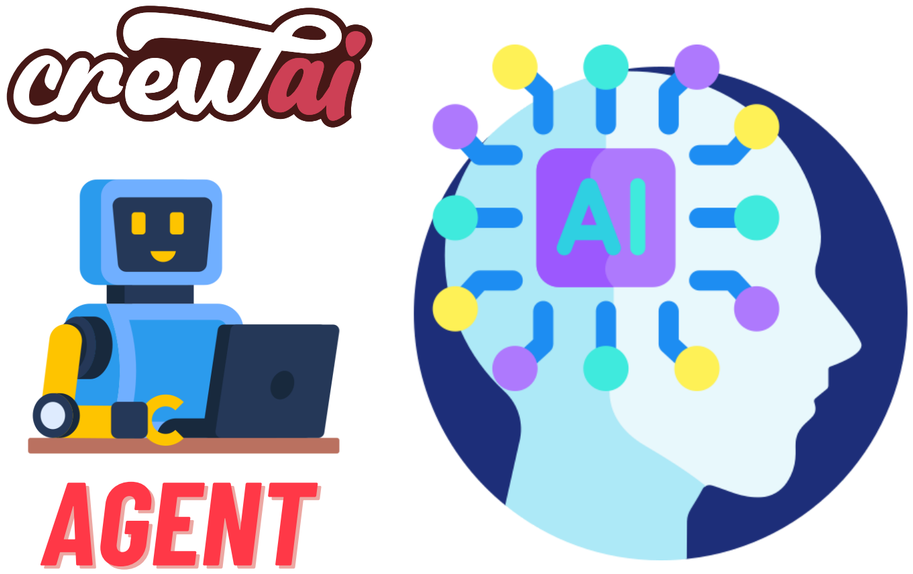
3° Estruturar um Agent e uma Task



Usando a classe Agent

```
research_agent = Agent(
    role='Researcher',
    goal='To gather information from the web.',
    backstory="You're a web researcher who updat
    tools=[search_tool],
    verbose=True,
    allow_delegation=False
)
```

A classe Agent é uma construção fundamental no framework CrewAI, projetada para representar **agentes autônomos que desempenham papéis específicos dentro de uma equipe** (ou "Crew"). Esses agentes podem ser configurados com objetivos, ferramentas e comportamentos específicos, permitindo uma interação dinâmica e colaborativa entre eles para a realização de tarefas complexas.



A classe Agent é uma construção fundamental no framework CrewAI, projetada para representar **agentes autônomos que desempenham papéis específicos dentro de uma equipe** (ou "Crew"). Esses agentes podem ser configurados com objetivos, ferramentas e comportamentos específicos, permitindo uma interação dinâmica e colaborativa entre eles para a realização de tarefas complexas.

role (papel): Define o papel ou a função do agente dentro da equipe. No seu exemplo, o papel do agente é `'Researcher'`, indicando que seu principal objetivo é realizar pesquisas.

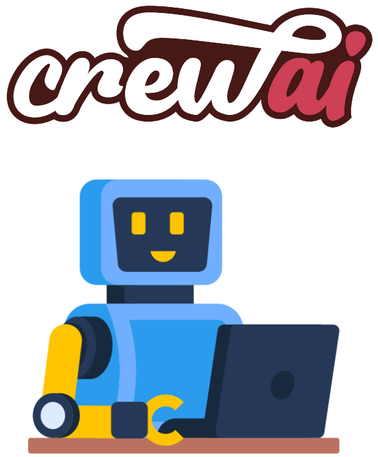
goal (objetivo): Descreve o objetivo que o agente busca alcançar. Para o seu `research_agent`, o objetivo é `'To gather information from the web.'`, significando que o agente se concentra em coletar informações da web.

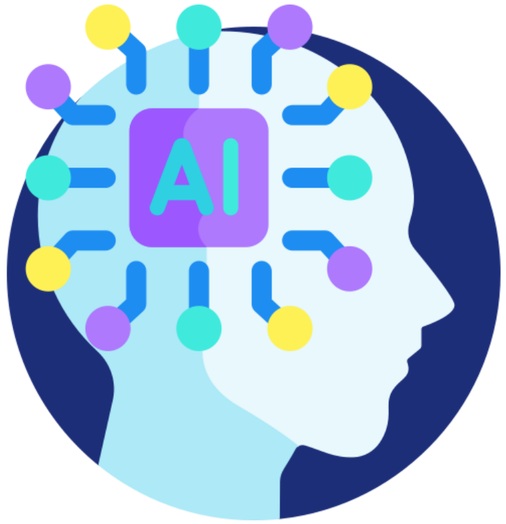
backstory (história): Fornece um contexto narrativo para o agente, o que pode ajudar a entender sua função e motivação de uma maneira mais rica e humana. No exemplo, a história é `"You're a web researcher who updates me with information on currency quotes"`, o que sugere uma especialização em atualizar informações sobre cotações de moedas.

tools (ferramentas): Uma lista das ferramentas que o agente pode utilizar para realizar suas tarefas. No código exemplo, `tools=[search_tool]` indica que o agente tem acesso a uma ferramenta chamada `search_tool`, que provavelmente é usada para realizar pesquisas na internet. As ferramentas são implementações específicas que facilitam a execução de ações ou tarefas pelo agente, como fazer buscas na web, processar dados, etc.

verbose: Um booleano que, quando verdadeiro (`True`), habilita a exibição de mensagens detalhadas sobre as atividades do agente, proporcionando insights sobre o processo de trabalho do agente. Isso pode ser útil para fins de depuração ou simplesmente para acompanhar o progresso do agente de maneira mais interativa.

allow_delegation (permitir delegação): Este parâmetro determina se o agente pode delegar tarefas a outros agentes. No exemplo fornecido, `allow_delegation=False` indica que este agente específico não tem permissão para delegar tarefas a outros. Isso pode ser útil em situações em que o controle do fluxo de trabalho precisa ser mantido ou quando as tarefas são tão especializadas que a delegação não é desejável.



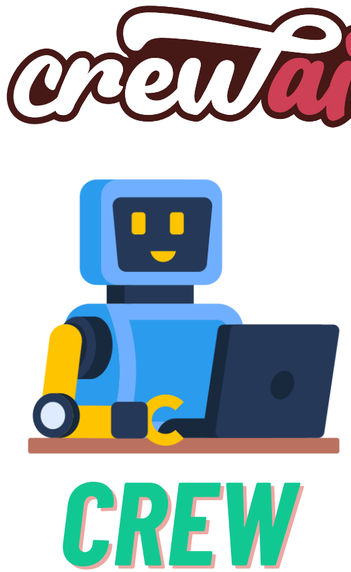


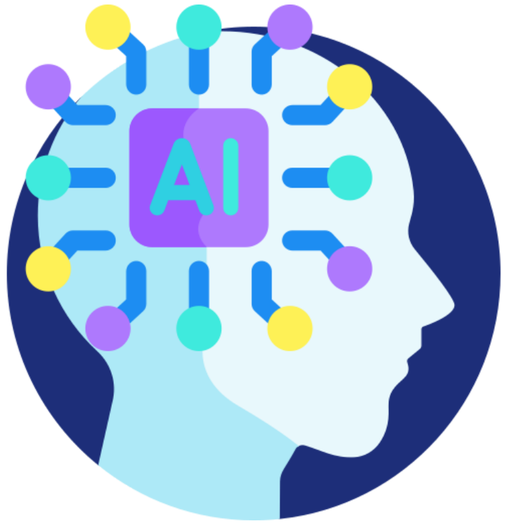

```python
search_task = Task(
    description='me responda apenas em português pt-BR e faça a
    agent=research_agent
)
```

A classe `Task` no CrewAI é utilizada para definir tarefas específicas que os agentes devem realizar. Cada tarefa é atribuída a um agente específico e contém instruções ou descrições detalhadas do que é esperado. Vou detalhar a definição da tarefa usando o exemplo de código que você forneceu:

- **description (descrição):** Define o que a tarefa envolve ou o que deve ser alcançado com ela. No exemplo, a descrição da tarefa é `'me responda apenas em português pt-BR e faça a pesquisa rápido no site infomoney: Qual a cotação atual do bitcoin?'`. Isso instrui o agente a responder em português do Brasil e a realizar uma pesquisa rápida no site InfoMoney para encontrar a cotação atual do bitcoin. A descrição serve como uma diretriz clara para o agente sobre o que é esperado dele.
- **agent:** Especifica o agente ao qual a tarefa é atribuída. No exemplo, a tarefa é atribuída ao `research_agent`, o que significa que é esse agente específico que deve executar a tarefa conforme descrito. Atribuir uma tarefa a um agente específico permite uma distribuição eficaz do trabalho dentro de uma equipe de agentes, garantindo que as tarefas sejam realizadas por agentes com as habilidades e ferramentas apropriadas para elas.

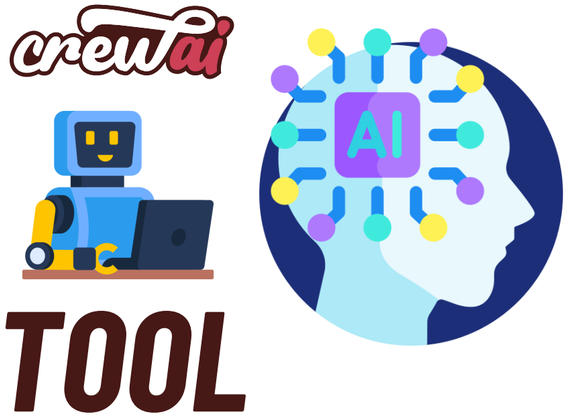
A classe `Task` é fundamental no CrewAI para organizar o fluxo de trabalho e as responsabilidades dentro de uma equipe de agentes. Cada tarefa criada e atribuída contribui para a realização dos objetivos globais da equipe, permitindo uma orquestração eficiente e coordenada das atividades dos agentes. As tarefas podem variar amplamente em complexidade e escopo, desde pesquisas simples na web até análises de dados complexas, dependendo das habilidades e ferramentas disponíveis para os agentes designados.





A classe `Crew` no CrewAI é o núcleo que reúne agentes e tarefas para formar uma equipe funcional capaz de executar uma sequência de operações ou resolver problemas complexos de maneira colaborativa. Esta classe é responsável por orquestrar o trabalho dos agentes, assegurando que as tarefas sejam executadas de acordo com um processo definido, o que pode incluir a execução sequencial ou paralela de tarefas, dependendo das necessidades do problema em questão.

```
crew = Crew(
    agents=[research_agent],
    tasks=[search_task],
    verbose=True
)
```



Para criar suas próprias funções que possam ser utilizadas pelos agentes no CrewAI, você pode definir custom tools utilizando a funcionalidade de tools do LangChain. Essas custom tools permitem que você integre funcionalidades específicas ou lógicas de negócios personalizadas que seus agentes podem invocar durante a execução de suas tarefas. Vamos detalhar como você pode criar uma dessas funções com um exemplo.

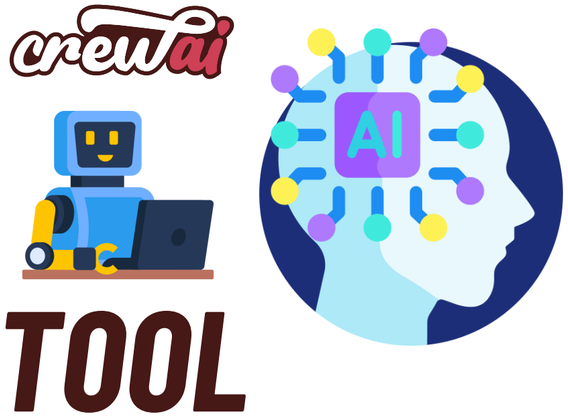
Exemplo de Criação de uma Custom Tool

Suponha que você queira criar uma ferramenta chamada `**multiplier**`, que tem como objetivo multiplicar dois números. Você pode fazer isso utilizando o decorador `**@tool**` fornecido pelo LangChain, que facilita a criação de ferramentas que podem ser facilmente integradas ao seu agente. Aqui está um exemplo de como isso pode ser feito:

python — Copy code

```python
from langchain.tools import tool

@tool
def multiplier(numbers: str) -> float:
    """Multiplies two numbers together.
    The input to this tool should be a string representing two numbers separated by a
    For example, the input '3,4' would result in 12 because 3 * 4 = 12.
    """
    # Separa os números com base na vírgula
    a, b = numbers.split(',')
    # Converte os números para float e realiza a multiplicação
    result = float(a) * float(b)
    # Retorna o resultado
    return result
```



Integração da Custom Tool com um Agente

Após criar a sua ferramenta, você pode integrá-la a um agente do CrewAI. Primeiro, você precisa registrar a ferramenta e, em seguida, associá-la ao seu agente, como mostrado abaixo:

python — Copy code

```python
from crewai import Agent
from langchain.agents import Tool

# Criação e registro da ferramenta
multiplier_tool = Tool(
    name="Multiplier Tool",
    func=multiplier,
    description="A tool to multiply two numbers"
)

# Definição do agente com a ferramenta customizada
math_agent = Agent(
    role='Mathematician',
    goal='Perform mathematical operations',
    tools=[multiplier_tool],
    verbose=True
)
```

Esse processo permite que o `math_agent` utilize a `multiplier_tool` para realizar operações matemáticas específicas, como parte de suas tarefas atribuídas. Essa abordagem de definir custom tools e integrá-las com agentes fornece uma poderosa maneira de estender a funcionalidade do CrewAI, permitindo a implementação de lógicas complexas e específicas de acordo com as necessidades do seu domínio de aplicação.