Fundamentos de Linguagem Python
Do Básico a Aplicações de IA

Mini-Projeto 2
O Que São Decoradores em Python?

Um decorador em Python é um recurso que permite modificar ou estender o comportamento de funções, métodos ou classes sem alterar diretamente o seu código.

O decorador funciona como uma “função de ordem superior” que recebe outra função (ou método/classe) como argumento e retorna uma nova versão dela, geralmente adicionando alguma funcionalidade extra.

Na prática, um decorador é aplicado usando o símbolo @ antes da definição da função. Por exemplo:

```
def meu_decorador(func):
    def wrapper():
        print("Antes da função")
        func()
        print("Depois da função")
    return wrapper

@meu_decorador
def saudacao():
    print("Olá, mundo!")

saudacao()
```

Quando você chama saudacao(), na verdade está chamando a função wrapper, que adiciona o comportamento extra antes e depois de executar a função original. Esse mecanismo é muito usado em Python para logging, autenticação, controle de acesso, medição de tempo de execução e até frameworks web como Flask e Django.

No código mostrado nas aulas, temos três decoradores diferentes, cada um com um propósito específico em Python:

@property

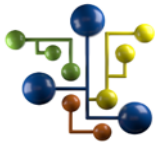
Esse decorador transforma um método em uma propriedade da classe. Em vez de chamar obj.saldo() como se fosse uma função, você pode acessar obj.saldo como se fosse um atributo comum. Isso permite controlar o acesso a atributos privados (como _saldo), aplicando lógica de leitura (getter) sem expor diretamente a variável interna.

@classmethod

Esse decorador indica que o método recebe a classe (cls) como primeiro argumento, em vez de receber a instância (self). Isso permite que o método atue no nível da classe, acessando ou modificando atributos compartilhados por todas as instâncias. No exemplo, ele consulta o número total de contas criadas (cls._total_contas).

@abstractmethod

Esse decorador, usado junto com classes abstratas (da biblioteca abc), obriga que qualquer subclasse da classe atual implemente esse método. Ele define um contrato: toda classe filha precisa fornecer a sua própria versão de sacar, senão não poderá ser instanciada.



Equipe DSA

Muito Obrigado!
Continue Trilhando Uma Excelente Jornada de Aprendizagem.