O decorador @classmethod no Python é usado para definir um método da classe que pode ser chamado diretamente na classe **sem precisar de uma instância**. Ele é útil quando o método precisa acessar ou modificar atributos da **classe** (e não de uma instância específica).

**Problema**

Considere uma classe Pessoa que tem um atributo da **classe** chamado especie, indicando que todas as pessoas pertencem à espécie "Homo sapiens":

python

Copy code

class Pessoa:

especie = "Homo sapiens"

def mostrar\_especie(self):

return Pessoa.especie

# Usando a classe

print(Pessoa().mostrar\_especie()) # "Homo sapiens"

Neste caso, criamos uma **instância** de Pessoa para acessar especie. Mas e se quisermos acessar especie diretamente pela classe? Podemos usar um @classmethod!

**Usando @classmethod**

Com @classmethod, criamos um método que recebe a própria classe como primeiro argumento (convencionalmente chamado de cls). Isso permite que o método acesse e manipule atributos da classe.

**Exemplo básico:**

python

Copy code

class Pessoa:

especie = "Homo sapiens"

@classmethod

def mostrar\_especie(cls):

return cls.especie

# Usando a classe

print(Pessoa.mostrar\_especie()) # "Homo sapiens"

Aqui, mostrar\_especie é um método da classe e pode ser chamado sem criar uma instância.

**Quando usar @classmethod**

1. **Para inicialização alternativa**  
   Ele é frequentemente usado como um "construtor alternativo", ou seja, para criar instâncias da classe de maneiras diferentes.

**Exemplo: Criando instâncias de formas alternativas**

python

Copy code

class Pessoa:

def \_\_init\_\_(self, nome, idade):

self.nome = nome

self.idade = idade

@classmethod

def criar\_a\_partir\_ano\_nascimento(cls, nome, ano\_nascimento):

idade = 2024 - ano\_nascimento # Supondo o ano atual

return cls(nome, idade)

# Criando instâncias

p1 = Pessoa("Ana", 25)

p2 = Pessoa.criar\_a\_partir\_ano\_nascimento("João", 1995)

print(p1.nome, p1.idade) # Ana, 25

print(p2.nome, p2.idade) # João, 29

Aqui, usamos o @classmethod para calcular a idade automaticamente e criar uma instância de Pessoa com base no ano de nascimento.

1. **Para manipular atributos da classe**  
   Se você precisa modificar valores que pertencem à classe (e não às instâncias), @classmethod é a melhor opção.

**Exemplo: Atualizando atributos da classe**

python

Copy code

class Pessoa:

especie = "Homo sapiens"

@classmethod

def mudar\_especie(cls, nova\_especie):

cls.especie = nova\_especie

# Antes de alterar

print(Pessoa.especie) # "Homo sapiens"

# Alterando o atributo da classe

Pessoa.mudar\_especie("Homo sapiens sapiens")

# Depois de alterar

print(Pessoa.especie) # "Homo sapiens sapiens"

**Diferença entre @classmethod e @staticmethod**

* **@classmethod**: Recebe a classe como primeiro argumento (cls), podendo acessar e alterar atributos da classe.
* **@staticmethod**: Não recebe a classe ou a instância automaticamente. É apenas uma função que pertence à classe, sem qualquer conexão com ela.

**Exemplo de ambos:**

python

Copy code

class Exemplo:

valor\_da\_classe = 10

@classmethod

def obter\_valor\_da\_classe(cls):

return cls.valor\_da\_classe

@staticmethod

def soma(a, b):

return a + b

# Usando os métodos

print(Exemplo.obter\_valor\_da\_classe()) # 10 (acessa a classe)

print(Exemplo.soma(5, 7)) # 12 (apenas uma função)

Se quiser mais exemplos ou detalhes, é só perguntar! 😊

4o