08

Para saber mais: Métodos estáticos e de classe

Da mesma forma que temos alguns **atributos diretamente da classe**, e que são acessíveis sem necessidade de criar uma instância, conseguimos também criar métodos ligados à classe.

Há duas formas de criar estes métodos:

Métodos de classe

São métodos decorados com `@classmethod`. Quando criamos um método de classe, temos acesso aos atributos da classe. Da mesma forma com os atributos de classe, podemos acessar estes métodos de dentro dos métodos de instância, a partir de `__class__`, se desejarmos:

```
class Funcionario:
    prefixo = 'Instrutor'

    @classmethod
    def info(cls):
        return f'Esse é um {cls.prefixo}'
```

Perceba que, ao invés de `self`, passamos `cls` para o método, já que neste caso sempre recebemos uma instância da classe como primeiro argumento. O nome `cls` é uma convenção, assim como `self`.

Métodos estáticos

A outra forma de criar métodos ligados à classe é com a decoração `@staticmethod`. Veja abaixo:

```
class FolhaDePagamento:
    @staticmethod
    def log():
        return f'Isso é um log qualquer'
```

Note que, no caso acima, não precisamos passar nenhum primeiro argumento fixo para o método estático. Nesse caso, não é possível acessar as informações da classe, porém o método estático é acessível a partir da classe e também da instância.

Cuidados a tomar...

Sempre que você usar métodos estáticos em classes que contém herança, observe se não está tentando acessar alguma informação da classe a partir do método estático, pois isso pode dar algumas dores de cabeça pra entender o motivo dos problemas.

Alguns *pythonistas* não aconselham o uso do `@staticmethod`, já que poderia ser substituído por uma simples função no corpo do módulo. Outros mais puristas entendem que os métodos estáticos fazem sentido, sim, e que devem ser vistos como responsabilidade das classes.