08

Para saber mais: Atributos de classe

Quando criamos atributos no inicializador, estamos definindo quais serão as características do objeto sendo definido. Mas esta não é a única forma de adicionar características ao objeto ou mesmo à classe.

O aspecto dinâmico da linguagem permite que seja possível adicionar atributos até sem precisar do `__init__` . Veja abaixo:

```
class Pessoa:
    pass

pessoa = Pessoa()
pessoa.nome = 'Jade'
print(pessoa.nome)
```

Se você tentar executar este código, verá que funciona perfeitamente.

Optamos em usar o inicializador, primeiro para facilitar a criação de novos objetos e segundo para diminuir a confusão em saber o que a classe precisa para criar um objeto aceitável. Sem o `init` , não dá para saber facilmente quais atributos a classe possui.

Normalmente usamos o `init` para definir os atributos, mas o que fazer se precisarmos definir um valor padrão para todos os objetos? Ou até criar um atributo que será compartilhado para todas as instâncias?

Para isto, vai ser necessário criar um atributo ligado à classe, ao invés de ligado à instância (`self`). Por exemplo:

```
class Pessoa:
    tamanho_cpf = 11

    def __init__(self, cpf, nome):
        self.cpf = cpf
        self.nome = nome

    def valida_cpf(self):
        return True if len(self.cpf) == __class__.tamanho_cpf else False

pe = Pessoa('00000000001', 'Ruby')
print(pe.valida_cpf())

pe = Pessoa('0000000000', 'Cristal')
print(pe.valida_cpf())
```

Veja como o valor de `tamanho_cpf` é usado por todas as instâncias.

Esse é um atributo de classe. É possível alterar o valor deste atributo, mudando seu estado e não é necessário criar uma instância para acessá-lo.

No trecho de código acima, precisamos usar o `__class__` para definir que queremos o atributo de classe. Dentro do nosso método de instância precisamos fazer desta forma.

Se não fizermos deste jeito, podemos ter problemas, como no código abaixo. Faça um teste:

```
class Pessoa:
    tamanho_cpf = 11

p = Pessoa()

print(p.tamanho_cpf)

p.tamanho_cpf = 12

print(p.tamanho_cpf)

print(Pessoa.tamanho_cpf)
```

O que acontece é que, caso não exista o atributo `tamanho_cpf` na instância, o Python busca o atributo na classe. Em seguida, adicionamos um atributo `tamanho_cpf` na instância e quando dizemos que o valor é 12, o atributo da classe não é alterado, já que são atributos diferentes, um da classe e outro só da instância.