03-POO-atributos_instancia_vs_classe

March 3, 2020

1 Programação Orientada aos Objetos (POO) - parte III

Pedro Cardoso
(ISE/UAlg - pcardoso@ualg.pt)

1.1 Atributos de instância vs. de classe

- Um **atributo de instância** é uma variável Python pertencente a um e apenas um objeto. Essa variável é acessível apenas no âmbito deste objeto e é definida dentro da função construtora `__init__(self, ..)` ou outro método do objeto.
- Um **atributo de classe** é uma variável Python que pertence a uma classe e não a um objeto específico, sendo compartilhado entre todos os objetos dessa classe e é definido **fora** da função do construtor, `__init__(self, ..)`, da classe ou outro método do objeto.

```
In [1]: class Carro:
            # atributo da classe
            numero_carros = 0

            def __init__(self, marca, modelo):
                # atributos da instancia
                self.marca = marca
                self.modelo = modelo

                # acesso ao atributo da classe (self.__class__ == Carro)
                # valor incrementado cada vez que é criado um objeto (contador de instanciaçõe
                self.__class__.numero_carros +=1
```

```
In [2]: carro_a = Carro('Fiat', '500')
        carro_b = Carro('Audi', 'A4')
        carro_c = Carro('Seat', 'Ibiza')
```

Podemos saber quantos objetos fda classe Carro foram instanciados

```
In [3]: print('numero de carros instanciados: {}'.format(Carro.numero_carros))
```

```
numero de carros instanciados: 3
```

Tambem podemos, mas "não o devemos" fazer do seguinte modo

```
In [4]: print('numero de carros instanciados: {}'.format(carro_a.numero_carros))
```

```
numero de carros instanciados: 3
```

Vejamos quais são os atributos do objeto `carro_a`

```
In [5]: carro_a.__dict__
```

```
Out[5]: {'marca': 'Fiat', 'modelo': '500'}
```

Note-se que se fizer

```
In [6]: carro_a.numero_carros = 10
```

então, passou a existir um atibuto `numero_carros` no objeto `carro_a`

```
In [7]: carro_a.__dict__
```

```
Out[7]: {'marca': 'Fiat', 'modelo': '500', 'numero_carros': 10}
```

que não existe nos outros objetos

```
In [8]: carro_b.__dict__
```

```
Out[8]: {'marca': 'Audi', 'modelo': 'A4'}
```

e que não é o (mesmo) atributo da classe `Carro`

```
In [9]: Carro.__dict__
```

```
Out[9]: mappingproxy({'__module__': '__main__',
                      'numero_carros': 3,
                      '__init__': <function __main__.Carro.__init__>,
                      '__dict__': <attribute '__dict__' of 'Carro' objects>,
                      '__weakref__': <attribute '__weakref__' of 'Carro' objects>,
                      '__doc__': None})
```

Porque...

```
In [10]: id(Carro.numero_carros) != id(carro_a.numero_carros)
```

```
Out[10]: True
```

mas

```
In [11]: id(Carro.numero_carros) != id(carro_b.numero_carros)
```

```
Out[11]: False
```

```
In [ ]:
```