Acessos genérico a atributos (getattr e setattr)

Juliana Pirolla - Revisão

Os métodos **getattr** e **setattr** nos permite lidar de forma especial com acesso a atributos de objetos

___gettattr_-

- Chamado sempre que tentamos acessar para *leitura* um atributo que **não foi definido** para esse objeto.

setattr

- Chamado quando se deseja **alterar** o valor de um atributo, seja ele **definido ou não** (sempre é chamado).
- **importante**: cuidado com o acesso a atributos dentro do próprio `__setattr__`**: os acessos devem ser feitos através do dicionário do objeto, `__dict__`, que contém os pares nome/valor dos atributos.

```
class AcessosAtributos:
    def __init__(self):
        # criando variaveis quaisquer
        self._a = 1
        self._b = 2

    # metodo chamado qnd queremos fazer a leitura de um atributo que não foi definido aind
    def __getattr__(self, name_attr):
        print('chamando getattr')
        self.__dict__[name_attr] = 0  # cria o atributo e retorna 0
        return 0


    # metodo chamado qnd queremos alterar o valor do atributo (ja existente ou nao).
```

```
    # como não é usado para leitura, só alteração, não retorna nada
    def __setattr__(self, name_attr: str, novo_val: any) -> None:
        print('chamando setattr')
        self.__dict__[name_attr] = novo_val
```

```
objeto1 = AcessosAtributos()
```

```
chamando setattr
chamando setattr
```

```
objeto1.b
```

```
chamando getattr

0
```

Lembre-se que quando executamos `objeto1.a` o **getattr** é chamado e retorna zero pois deixamos o acesso privado ao atributo a. Logo, ele considera como inexistente.

```
objeto1._b
```

```
2
```

Indicando que está privado ele retorna o valor que atribuimos no início (lembrando que a melhor forma de acessar um atributo privado é através de métodos get que vão manter a encapsulação e ocultar a implementação do usuário).

Acessar atributos que ainda não definimos:

```
objeto1.c
```

```
chamando getattr

0
```

Alterando valores:

```
objeto1.c = 15
objeto1.c
```

```
chamando setattr
```

```
15
```

Acessando o dicionário de objeto1:

```
objeto1.__dict__
```

```
{'_a': 1, '_b': 2, 'b': 0, 'c': 15}
```

Método getattribute

Os métodos **getattr** e **setattr** são implementados para ler attr que ainda não definimos e modificar seus valores. O método **getatribute** segue o mesmo raciocínio, mas é implementado para que a leitura de qualquer atributo (definido ou não) passe por ele. Isso é útil apenas em situações muito especiais...

```
class AcessoAtributos2:
    def __init__(self):
        self._a = 1
        self._b = 2

    def __getattr__(self, name_attr):
        # devemos chamar super() quando definimos __getattribute__ para evitar recursao
        super().__getattribute__('__dict__')[name_attr] = 0
        return 0

    def __getattribute__(self, nome_attr):
        return super().__getattribute__(name)

    def __setattr__(self,name, new_val):
        self.__dict__[name] = new_val
```