Encapsulamento no Python

Encapsulamento sem private

Em python não podemos forçar o encapsulamento como em outras linguagens ue utilizam o termo Private. O que podemos realizar é a implementação de métodos como getters e setters ou mesmo decoradores. Uma importante convenção é utilizar o _ para indicar que um atributo é privado.

Getter ou property

Getter

Se quisermos expor o conteúdo de um atributo privado, podemos declarar um método que apenas retorna esse valor, evidenciando o que está contido no atributo mas sem deixar margem para alterá-lo. Esse método é o que chamamos de get.

```
class BancoFuncionarios:
    def __init__(self, name):
        self._name = name

    def getName(self):  # sua única função é retornar e mostrar p usuario o que tem dentro
        return self._name
```

```
funcionario = BancoFuncionarios('Juliana')
funcionario.getName()
```

```
'Juliana'
```

Property

Poderíamos também atingir o mesmo objetivo utilizando o decorador @property. A vantagem é que podemos acessar o atributo usando o nome do método declarado em @property, como se a classe BancoFuncionarios tivesse um atribto com esse nome.

```
class BancoFuncionarios:
    def __init__(self, name):
        self._name = name

    @property
    def name(self):
        return self._name
```

```
funcionario = BancoFuncionarios('Juliana')
funcionario.name  # aparece com aspas
print(funcionario.name)  # aparece sem aspas
```

```
Juliana
```