Programação Orientada a Objetos

Decorador @property

Prof. Ivonei Marques

Exemplo @property

O decorador `@property` em Python é usado para dar aos métodos de uma classe a aparência de atributos.

Isso permite que você acesse um método como se fosse um atributo, sem precisar usar parênteses para chamá-lo.

Os decoradores permitem alterar o comportamento de um objeto sem a necessidade de alterar a sua estrutura.

`@property` também é uma maneira de implementar encapsulamento.

Neste caso, não precisamos entender como a @property é implementada.
Mas só para contextualizar. Podemos inclusive criar nossos próprios decoradores.

Exemplo @property

Exemplo 1 - usando get e set

```
class Pessoa:
    def __init__(self, nome):
        self.__nome = nome

    def get_nome(self):
        return self.__nome

    def set_nome(self, nome):
        if nome.isalpha():
            self.__nome = nome
        else:
            print("Nome inválido.")

ti = Pessoa("Ana")
ti.set_nome("Anna")
print(ti.get_nome())
```

instância

bucando o atributo

alterando o nome

instância
alterando o nome
imprimindo o nome

Exemplo @property

Exemplo 2 - usando a classe property

```
class Pessoa:
    def __init__(self, nome):
        self.__nome = nome

    def get_nome(self):
        return self.__nome

    def set_nome(self, nome):
        if len(nome) >= 2:
            self.__nome = nome
        else:
            print("Nome inválido.")

   nome = property(get_nome, set_nome)

ti = Pessoa("Ana")
ti.nome = "Anna"
print(ti)
```

instância

bucando o atributo

alterando o nome

criando o método especial nome

instância
alterando o nome
imprimindo o nome

Exemplo @property

Exemplo 3 - usando o decorador @

```
class Pessoa:
    def __init__(self, nome):
        self.__nome = nome

    @property
    def nome(self):
        return self.__nome

    @nome.setter
    def nome(self, nome):
        if nome.isalpha():
            self.__nome = nome
        else:
            print("Nome inválido.")

ti = Pessoa("Ana")
ti.nome = "Anna"
print(ti.nome)
```

instância

bucando o atributo

alterando o nome

instância

alterando o nome

imprimindo o nome

Permite acesso a um método como se fosse um atributo, sem precisar usar parênteses para chamá-lo.

Exemplo @property

Comparativo

exemplo 1

```
def get_nome(self):
    return self.__nome

def set_nome(self, nome):
    if nome.isalpha():
        self.__nome = nome
    else:
        print("Nome inválido.")
```

exemplo 2

```
def get_nome(self):
    return self.__nome

def set_nome(self, nome):
    if len(nome) >= 2:
        self.__nome = nome
    else:
        print("Nome inválido.")

nome = property(get_nome, set_nome)
```

exemplo 3

```
@property
def nome(self):
    return self.__nome

@nome.setter
def nome(self, nome):
    if nome.isalpha():
        self.__nome = nome
    else:
        print("Nome inválido.")
```

```
ti = Pessoa("Ana")
ti.set_nome("Anna")
print(ti.get_nome())
```

```
ti = Pessoa("Ana")
ti.nome = "Anna"
print(ti.nome)
```