Programação Orientada a Objetos

Descritores

Descritores

Descritor (*descriptor*) é uma forma mais geral de definir como os atributos são calculados por meio de um protocolo. Com isso é possível obter um código mais fácil de reaproveitar do que com propriedades (*properties*) e os mesmos podem ser compartilhados entre as classes.

Um descritor permite que você personalize o que deve ser feito quando você se refere a um atributo em um objeto.

Descritores fornecem ao desenvolvedor a capacidade de adicionar atributos gerenciados aos objetos. Atributos gerenciados são usados para proteger um atributo contra alterações ou para atualizar automaticamente os valores de um atributo dependente.

Necessitamos de descritores quando queremos, por exemplo, validar a informação antes de designar a mesma ao atributo. Imagine que queira obrigar a informação de um número inteiro para idade, texto para nome e avaliar o formato de uma *string* de e-mail.

Descritores

O protocolo para criar um descritor é realmente muito simples. Você somente precisa definir um ou mais dos métodos a seguir: *__get__*, *__set__* e *__delete__*.

Onde:

__get__(self, obj, type = None) => Retorna o valor do atributo.

__set__(self, obj, value) => Informar um valor ao atributo (sem retorno).

__delete__(self, obj) => Controla operação de exclusão (sem retorno).

Ao definir pelo menos um destes métodos, você está criando um descritor.

Um descritor que implementa __get __ () e __set __ () é chamado de descritor de dados.

Se ele apenas implementar __get __ (), então ele é chamado de um descritor “sem dados” (non-data descriptor).

Descritores

get

```
print(usuario.nome)
```

set

```
usuario.nome = “Fulano de tal”
```

delete

```
del(usuario.nome)
```

Descritores

Veja um exemplo da documentação oficial do Python:

```
# descriptor_exemplo.py
class RevelarAcesso(object):
    """Um descriptor de dados que define e retorno
    valores normalmente e imprime uma mensagem
    registrando seu acesso."""
    def __init__(self, initval=None, name='var'):
        self.val = initval
        self.name = name

    def __get__(self, obj, objtype):
        print('Recuperando', self.name)
        return self.val

    def __set__(self, obj, val):
        print('Atualizando', self.name)
        self.val = val
```

```
class MinhaClasse(object):
    x = RevelarAcesso(10, 'var "x"')
    y = 5

m = MinhaClasse()

print("x: ", m.x)
m.x = 20
print("x: ", m.x)

print("y: ", m.y)
m.y = 8
print("y: ", m.y)
```

```
Recuperando var "x"
x:  10
Atualizando var "x"
Recuperando var "x"
x:  20
y:  5
y:  8
```

Descritores

O exemplo anterior mostra claramente que, se uma classe tiver o descritor de dados para o atributo dado, o método *__get__ ()* do descritor é chamado para retornar o valor sempre que o atributo da instância é recuperado e *__set__ ()* é chamado sempre que um valor é atribuído a esse atributo.

No caso do método *__del__*, ele é chamado sempre que um atributo de instância é excluído com a declaração del instance.atribute ou a chamada delattr(instância, 'atributo').

Descritores

Veja outro exemplo:

```
class MeuDescritor(object):
    """
    Um exemplo simples de descritor
    """
    def __init__(self, valor_inicial=None, nome='my_var'):
        self.valor = valor_inicial
        self.nome = nome

    def __get__(self, instance, owner):
        print("Obtendo: ", self.nome)
        return self.valor

    def __set__(self, instance, valor):
        print(f"Atribuindo {valor} a {self.nome}")
        self.valor = valor

class MinhaClasse(object):
    descritor = MeuDescritor(valor_inicial='10', nome='dinheiro')
    normal = 20

classe = MinhaClasse()
print(classe.descritor) # Executa o get para imprimir o valor, então imprime a linha do "Obtendo"
print(classe.normal) # Não é um descritor
classe.descritor = 200 # Executa o __set__
print(classe.descritor) # Imprime agora 200
```

Descritores

Observe o código a seguir. Veja como estou tratando para não permitir veículos com valor negativo.

```
class Carro(object):
    def __init__(self, marca, modelo, valor):
        self.marca = marca
        self.modelo = modelo
        if valor<0:
            raise ValueError("O valor do carro não pode ser negativo.")
        else:
            self.valor = valor

    def __str__(self):
        return f"Marca: {self.marca}, Modelo: {self.modelo}, Valor: R$ {self.valor:.2f}"

fusquinha = Carro("VW", "Fusca", 8500)
print(fusquinha)
fusquinha = Carro("VW", "Fusca", -1)
print(fusquinha)
```

Descritores

Podemos resolver criando um descritor para validar valores negativos que pode ser reaproveitado.

```python
class DescritorValor(object):
    def __init__(self):
        self.valor = 0

    def __get__(self, instance, owner):
        return self.valor

    def __set__(self, instance, value):
        if value<0:
            raise ValueError("O valor do carro não pode ser negativo.")
        else:
            self.valor = value

    def __delete__(self, instance):
        del self.valor
```

```python
class Carro(object):
    valor = DescritorValor()
    def __init__(self, marca, modelo, valor):
        self.marca = marca
        self.modelo = modelo
        self.valor = valor

    def __str__(self):
        return f"Marca: {self.marca}, Modelo: {self.modelo}, " \
               f"Valor: R$ {self.valor:.2f}"

fusquinha = Carro("VW", "Fusca", 8500)
print(fusquinha)

fusquinha2 = Carro("VW", "Fusca", -1)
```

```
Marca: VW, Modelo: Fusca, Valor: R$ 8500.00
Traceback (most recent call last):
  File .../carro_descritores.py", line 32, in <module>
    fusquinha2 = Carro("VW", "Fusca", -1)
  File ".../carro_descritores.py", line 22, in __init__
    self.valor = valor
  File ".../carro_descritores.py", line 10, in __set__
    raise ValueError("O valor do carro não pode ser negativo.")
ValueError: O valor do carro não pode ser negativo.
```

Descritores

Neste exemplo ainda temos um problema devido ao valor ser uma propriedade da classe Carro.

```
class DescritorValor(object):
    def __init__(self):
        self.valor = 0

    def __get__(self, instance, owner):
        return self.valor

    def __set__(self, instance, value):
        if value<0:
            raise ValueError("O valor do carro não pode ser negativo.")
        else:
            self.valor = value

    def __delete__(self, instance):
        del self.valor
```

```
class Carro(object):
    valor = DescritorValor()
    def __init__(self, marca, modelo, valor):
        self.marca = marca
        self.modelo = modelo
        self.valor = valor

    def __str__(self):
        return f"Marca: {self.marca}, Modelo: {self.modelo}, " \
               f"Valor: R$ {self.valor:.2f}"
```

```
fusquinha = Carro("VW", "Fusca", 8500)
print(fusquinha)

gol = Carro("VW", "Gol", 25000)
print(gol)
print(fusquinha)
```

```
Marca: VW, Modelo: Fusca, Valor: R$ 8500.00
Marca: VW, Modelo: Gol, Valor: R$ 25000.00
Marca: VW, Modelo: Fusca, Valor: R$ 25000.00
```

Descritores

Podemos resolver utilizando um dicionário na propriedade "valor" e passar a instância do objeto no *get*, *set* e *delete*.

```
class DescritorValor(object):
    def __init__(self):
        self.valor = {}

    def __get__(self, instance, owner):
        return self.valor[instance]

    def __set__(self, instance, value):
        if value<0:
            raise ValueError("O valor do carro não "
                             "pode ser negativo.")
        else:
            self.valor[instance] = value

    def __delete__(self, instance):
        del self.valor[instance]
```

```
class Carro(object):
    valor = DescritorValor()
    def __init__(self, marca, modelo, valor):
        self.marca = marca
        self.modelo = modelo
        self.valor = valor

    def __str__(self):
        return f"Marca: {self.marca}, Modelo: {self.modelo}, " \
               f"Valor: R$ {self.valor:.2f}"

fusquinha = Carro("VW", "Fusca", 8500)
gol = Carro("VW", "GOL", 20000)
print(gol)
print(fusquinha)
fusquinha2 = Carro("VW", "Fusca 2", 700)
print(fusquinha2)
print(gol)
print(fusquinha)
```

```
Marca: VW, Modelo: GOL, Valor: R$ 20000.00
Marca: VW, Modelo: Fusca, Valor: R$ 8500.00
Marca: VW, Modelo: Fusca 2, Valor: R$ 700.00
Marca: VW, Modelo: GOL, Valor: R$ 20000.00
Marca: VW, Modelo: Fusca, Valor: R$ 8500.00
```

CONTINUA...