Properties

Repaso - Jueves 30 de abril 2020

¿Es necesario siempre ocuparlas con atributos encapsulados?

No. Siempre radica en en una decisión de asignación de responsabilidades entre entidades. ¿Quién se encarga de modificar el valor de un atributo? ¿Cualquiera puede? ¿Hay algo que impida dejarle a cualquiera cambiar un valor?

Ejemplo

```
class Fracción:

    def __init__(self, numerador, denominador):
        self.numerador = numerador
        self._denominador = 1
        self.denominador = denominador

    @property
    def denominador(self):
        return self._denominador

    @denominador.setter
    def denominador(self, valor):
        if valor != 0:
            self._denominador = valor
```

$$\text{fracción} = \frac{\text{numerador}}{\text{denominador}}$$

¿Cuál es la diferencia de programar sin ellas? ¿Cúal es su utilidad si podemos cambiar valores dentro de métodos?

El comportamiento de *properties* se puede imitar mediante métodos, pero proveen una interfaz más conveniente y semánticamente más correcta.

Ejemplo

```
class Fracción:

    def __init__(self, numerador, denominador):
        self.numerador = numerador
        self._denominador = 1
        self.denominador = denominador

    @property
    def denominador(self):
        return self._denominador

    @denominador.setter
    def denominador(self, valor):
        if valor != 0:
            self._denominador = valor

    @property
    def valor(self):
        return self.numerador / self.denominador
```

$$\text{fracción} = \frac{\text{numerador}}{\text{denominador}}$$

Ejemplo (adaptado de AC03 2015-1)

Se busca modelar una clase `Carrito` que contiene instancias de `Producto`. Cada carrito es capaz de agregar y remover productos de si mismo. Se busca que se pueda extraer de cada carrito la siguiente información:

- Los tipos de producto que contiene actualmente (sin repeticiones)
- La cantidad de productos por tipo que contiene actualmente.

Ejemplo (adaptado de AC03 2015-1)

```
class Carrito:

    def __init__(self):
        self.productos = []

    def agregar(self, producto):
        self.productos.append(producto)

    def remover(self, producto):
        self.productos.remove(producto)

    ....
```

```
class Producto:

    def __init__(self, tipo, precio):
        self.tipo = tipo
        self.precio = precio
```

Notar que `tipos_productos` y `cantidad_por_tipos` a fin de cuentas, serán características de `Carrito`, tal como lo era `valor` en el ejemplo de `Fraccion`

Otros ejercicios

¡Si tienen dudas resolviendolos, no duden en hacer una *issue* con su duda!

- Ejercicios propuestos sobre *properties*. (Contenidos: semana 2 - parte 2)
- Actividad 1 en 2017-1 (Actividad sobre OOP, con uso de *properties*)
- Actividad 1 en 2018-1 (Actividad sobre OOP, con uso de *properties*)
- Actividad 4 en 2018-2 (Actividad sobre OOP, con uso de *properties*)
- Actividad 10 en 2019-1 (Actividad Recuperativa, donde la parte 2 es sobre OOP)
- Actividad 1 en 2019-2 (Actividad sobre OOP, con uso de *properties*)