En borrador permanente
el blog de Juanjo Conti – abstracto, lúdico y digital

Decoradores en Python (II) – Decoradores con parámetros

Publicado el 16 de julio, 2009 por Juanjo

Un año después del primer artículo, llega el segundo. ¿Por qué tardó tanto? Por lo general mis artículos técnicos surgen de algún problema que se me presenta y para el cual necesito investigar antes de poder solucionarlo. El artículo anterior cubría todo lo que necesité hacer con decoradores en Python hasta el mes pasado, cuando necesité decoradores con parámetros.

Si no leíste el artículo anterior, te recomiendo que lo hagas antes de seguir: Decoradores en Python (I).

Decoradores con Parámetros

Cuando quise escribir un decorador con un parámetro me encontré con errores que ni siquiera entendía. No solo que los estaba escribiendo mal, sino que también los estaba usando mal. Te voy a evitar el sufrimiento.

Un decorador con parámetro se aplica así (siendo deco un decorador y 1 el argumento utilizado):

```
@deco(1)
def funcion_a_decorar(a, b, c):
    pass
```

Creo que la raíz de mi confusión fue el azúcar sintáctica (si, el @). Así que vamos a sacarlo y ver cómo se usaría

este decorador en una versión de Python más vieja:

```
def funcion_a_decorar(a, b, c):
    pass
```

```
funcion_a_decorar = deco(1)(funcion_a_decorar)
```

Esto luce más claro para mi: deco es llamado con un argumento y el resultado tiene que ser algún objeto que pueda ser llamado con una función como parámetro para... decorarla. ¿Se entiende la idea? Vamos a definir deco, va a recibir un parámetro y utilizarlo para crear un decorador como los del artículo anterior. Finalmente retorna este decorador interno.

Agreguemos semántica al ejemplo. Mi decorador con parámetro recibirá un número, este número se usará para indicar cuantas veces queremos ejecutar la función decorada.

```
def deco(i):
    def _deco(f):
        def inner(*args, **kwargs):
            for n in range(i):
                r = f(*args, **kwargs)
            return r
        return inner
    return _deco
```

Como una convención personal, uso para el nombre de la segunda función _{nombre de la primer funcion}. Notemos entonces que _deco es un decorador dinámico, dependiendo del parámetro i, la función inner se compilará de una forma o de otra. Apliquemos el decorador:

```
@deco(2)
def saluda(nombre):
    print "hola", nombre
```

```
>>> saluda("juanjo")
hola juanjo
hola juanjo
```

```
@deco(3)
def suma1():
    global n
    n += 1
```

```
>>> n = 0
>>> suma1()
>>> n
3
```

Cuando aplicamos deco, se ejecuta deco, se compila _deco, se aplica _deco a la función que definimos y se compila inner utilizando un valor dado para i. Cuando llamamos a nuestra función (saluda, o suma1, en los ejemplos) se ejecuta inner.

¡Espero que se haya entendido!

Si no...

Si en lo anterior no fui lo suficientemente claro (por favor quejate en un comentario), no todo está perdido. Te puedo entregar un decorador para decoradores que convierte a tu decorador en un decorador con parámetros. ¿Qué tal?

```
def decorador_con_parametros(d):
    def decorador(*args, **kwargs):
        def inner(func):
            return d(func, *args, **kwargs)
        return inner
    return decorador
```

Original usando lambda en http://pre.activestate.com/recipes/465427/

Se usa así:

```
@decorador_con_parametros
def deco(func, i):
    def inner(*args, **kwargs):
        for n in range(i):
            r = func(*args, **kwargs)
        return r
    return inner
```

```
@deco(2)
def saludar(nombre):
    print "chau", nombre
```

```
>>> saludar("juanjo")
chau juanjo
chau juanjo
```

Para la próxima

Para el próximo artículo voy a explorar utilizar clases decoradoras en lugar de funciones decoradoras. Si bien todavía no lo terminé de investigar, me parece un enfoque que permite escribir código más organizado. Veremos! **update:** aquí está.

Acerca de Juanjo

Mi nombre es Juanjo Conti, vivo en Santa Fe y soy Ingeniero en Sistemas de Información. Mi lenguaje de programación de cabecera es Python; lo uso para trabajar, estudiar y jugar. Como hobby escribí algunos libros.

Ver todas las entradas por Juanjo →

Esta entrada fue publicada en Aprendiendo Python y etiquetada decoradores, Python. Guarda el enlace permanente.

5 Comentarios En borador permanente Iniciar sesión

Sort by Oldest Compartir Favorite

Join the discussion…

Nacho • hace 5 años

¿Clases decoradoras? ¿O métodos decoradores?
En fin, espero leer eso (prometo no googlear para leer tu artículo).

Muy bueno esto de los decoradores, recuerdo que los utilicé un tiempo cuando estaba estudiando Turbogears. Me recuerdan aires de lenguajes funcionales :-D

• Responder • Compartir ›

Facundo Batista • hace 5 años

Asumo que quiso decir "clases decoradas".

Los decoradores de clases son un feature de Python 3000.

• Responder • Compartir ›

jjconti Moderador • hace 5 años

Me refería a utilizar clases para decorar en lugar de utilizar funciones para decorar.

• Responder • Compartir ›

Alan Cristhian • hace un año

Hola, muy buenos tus artículos.

Tengo una duda: digamos que defino una variable dentro de la función decoradora. Y luego quiero usar esa variable dentro de la función decorada. ¿Como se hace?

Eso es todo. Saludos.

• Responder • Compartir ›

jjconti Moderador ↗ Alan Cristhian • hace un año

Podrías pasarla como parámetro.

• Responder • Compartir ›

En borrador permanente

Funciona con WordPress.