Decoradores en Python

by kotov - Miércoles, noviembre 28, 2012

http://rooteando.com/decoradores-en-python/

Un decorador nos permite realizar modificación simples en objetos llamables como son las funciones ,métodos o clases.,estas modificaciones se realizan en tiempo de ejecución. Una definición de los decoradores sería; es una función que recibe como parámetro otra función y devuelve una función,con lo que tenemos:

- El decorador que es una función.
- La función a decorar que se recibe como un parámetro.
- La función decorada que es la función que devuelve el decorador.

Con un decorador podemos cambiar el comportamiento de un función, método o clase sin modificar su código, esa es la ventaja de utilizar un decorador. Se crea un “envoltura” alrededor de la función a decorar donde se encuentra el código que deseamos añadir, esta envoltura y la función original, es la nueva función que obtenemos.

El decorador mas simple seria un decorador sin argumentos y la función a decorar sin argumentos.

1- Se define el decorador ,que como vemos es una función con nombre *decorador*, que recibe como argumento otra función.

2- Se define la función de envoltura donde se escribe el código del decorador**.**

3- Antes de ejecutar la función a decorar, mostrara el mensaje descrito, aquí podemos escribir cualquier código que se ejecutara antes de llamar a la función.

4- Llamamos a la función que se va a ejecutar.

5- Después de llamar a la función, podemos ejecutar código definido en el decorador. En este caso mostrara otro mensaje.

6- Devolvemos la función que sirve de envoltura, que incluye la función decorada.

Para utilizar el decorador utilizaremos el símbolo @ de la siguiente forma.

1- Para llamar a un decorador se utiliza el símbolo "@" y el nombre del decorador definido anteriormente.

2- Se define la función que se va a decorar, en este caso la función no tiene argumentos.

3- El código de la función, muestra un mensaje por pantalla.

6- Ejecutamos la función

El decorador añade código antes y después de ejecutar la función, primero muestra el mensaje definido en el decorador, se muestra el mensaje definido en la función y por último el mensaje definido en el decorador. Ahora nuestra función esta decorada, con lo que hemos modificado el comportamiento del función original sin introducir código dentro de la función. Si borramos la linea @decorador, la función solo ejecuta el código definido en la propia función.

. Utilizando decoradores generamos un código mejor organizado y mas legible.

Hemos visto el decorador mas simple, sin argumentos y la función sin argumentos, pero la cosas se pueden complicar, podemos tener decoradores para funciones con argumentos y decoradores son argumentos, también podemos definir clases decoradoras.

Se pueden aplicar varios decoradores a una misma función, no conviene abusar de esto, y el orden de ejecución de los decoradores es de abajo a arriba.Como desventaja, el uso de los decoradores añaden cierta complejidad al código.

Caso de ejemplo

Para ilustrar el uso de los decoradores se vera un ejemplo de una función decoradores en una función con argumentos y una clase decoradora en un método de una clase.

Tenemos un programa de gestión de un banco, que ofrece varios productos (cuentas bancarias, depósitos y tarjeta), para determinados tramites bancarios es necesario ser cliente del banco y para otras tareas no es necesario.Por ejemplo, para abrir una cuenta bancaria es obligatorio que sea un cliente, el programa pide el dni y busca en una base de datos si ese dni corresponde a un cliente del banco se crea la cuenta, si no es cliente la operación no se puede realizar hasta que se registre como cliente en la Base de datos.

En el programa se han definido diferentes clases para los clientes y los productos bancarios que se ofertan.Para la primera versión del programa , dentro de la clase Cliente se define una método que se encarga de comprobar si existe un cliente con el dni que se ha enviado, con esta versión me di cuenta que se repetía código en cada clase que define a los productos, con lo que decidí utilizar un decorador que me dijera si existe o no el cliente.

Para la segunda versión utilice una función decoradora que va a decorar una función con argumentos.

La función decoradora "*DecoradorClienteExiste*" recibe como parámetro la función que se va a decorar *"funcion"*. A continuación se define la envoltura "*wrapper"* que recibe como para el argumento "*arg",* que almacena el dni que se desea comprobar. Cuando se decora una función con argumentos, estos argumentos se definen en la función que va realizar la envoltura,en este caso la función *wrapper,* .

En la función *wrapper* se define el código del decorador, se realiza una consulta en una tabla Clientes para comprobar si el dni(arg) pertenece a un cliente,se esta utilizando el ORM SQLAlchemy para realizar la consulta. Mediante una excepción se controla el resultado de la consulta, si no existe cliente se muestra un mensaje y si el cliente existe se llama a la función para que ejecute su código.

El decorador añade código a la función original, si el cliente existe la función se ejecuta y si el cliente no existe la función no se ejecuta.

Por último, después de definir el decorador, se especifica donde se va usar el decorador.

Cuando se llame al método para crear una cuenta con un dni, primero se ejecuta el decorador que decidirá si se crea la cuenta o no.

Clase decoradora

Se he definido y utilizado una función decoradora ,pero a partir de la versión de 2.6 de *Python*,también se puede definir clases decoradoras.

Convertimos la función decoradora en una clase decoradora, aunque no cambiamos la funcionalidad. Como ventaja, la clase decoradora pueden incluir mas funcionalidad y el código esta mas organizado.

En este caso la clase decoradora se definen dos métodos; *init* y *call,* el primer método donde se recibe la función a decorar y el segundo método donde se encuentra el código que se añade.Como se puede ver, el método *call* contiene el mismo código definido en la función decoradora.

Enlaces

En este artículo solo se ha visto una parte pequeña de lo que se pueden hacer con los decoradores, a continuación especifico algunos que me han servidor para entender los decoradores.

http://www.python.org/dev/peps/pep-3129/ : Guia de estilo de desarrollo (PEP) sobre las clases decoradoras.

http://blog.apsl.net/weblog/2009/08/31/decoradores-en-python/ : Buen artículo sobre los decoradores en español.

http://www.artima.com/weblogs/viewpost.jsp?thread=240808 : Serie de dos artículos sobre decoradores en ingles.

http://stackoverflow.com/questions/739654/understanding-python-decorators : Excelente artículo explicando mediante ejemplo que se son los decoradores y que pueden hacer.
