



Metaprogramación

Decoradores

Índice

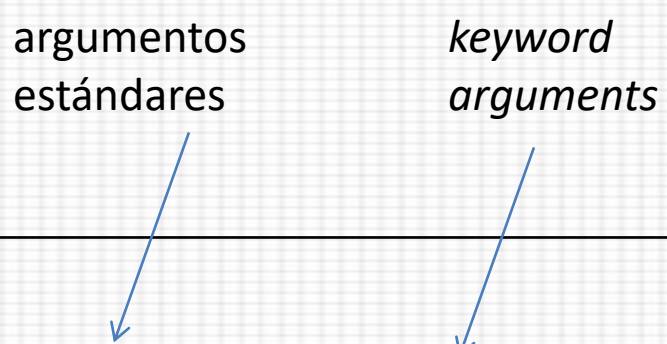
- Fundamentos sobre decoradores
- Creación de un decorador
- Aplicación de un decorador

Fundamentos sobre decoradores

- Permiten aplicar un código sin modificar el comportamiento de clases o funciones ya existentes
- Tareas de tipo transversal en una aplicación
- Técnicamente, son funciones que se aplican sobre una función y devuelven otra función
- Se aplica mediante `@nombre_decorador`

Creación de un decorador

➤ Se define como una función dentro de otra función:



```
def decorador(f):  
    def nueva_funcion(*args, **kwargs):  
        #código con la tarea a realizar por decorador  
        return f (*args, **kwargs) #devuelve la función recibida  
    return nueva_funcion #devuelve la función decorador
```

The diagram shows two blue arrows pointing from annotations to the function signature in the code block. The first arrow points from the text "argumentos estándares" to the "*args" parameter. The second arrow points from the text "keyword arguments" to the "**kwargs" parameter.

Aplicación de un decorador

- Se aplican sobre la función mediante la sintaxis @nombre_decorador.
- Cada vez que se llame a la función el decorador se ejecuta

```
@decorador  
def mifuncion(a, b, c=10):  
    pass
```

```
@decorador  
def otra_funcion(x,y)  
    pass
```

```
mifuncion(1,4)  
mifuncion(2,8,c=4)  
otra_funcion(10,7)
```

El decorador se aplica
con todas estas llamadas