### Convertir un argumento en opcional

A veces tiene sentido hacer que un argumento sea opcional, para que las personas que utilicen la función puedan elegir proporcionar información extra sólo si lo desean. Puedes utilizar valores por defecto para hacer que un argumento sea opcional.

Por ejemplo, supongamos que queremos ampliar get\_formatted\_name() para que también incluya los segundos nombres. Un primer intento de incluir los segundos nombres podría tener este aspecto:

def get\_formatted\_name(first\_name, middle\_name, last\_name):  
 """Return a full name, neatly formatted."""  
 full\_name = f"{first\_name} {middle\_name} {last\_name}"  
 return full\_name.title()  
  
musician = get\_formatted\_name('john', 'lee', 'hooker')  
print(musician)

Esta función funciona cuando se le da un nombre, un segundo nombre y un apellido. La función toma las tres partes del nombre y construye una cadena a partir de ellas. La función añade espacios donde corresponda y convierte el nombre completo a mayúsculas y minúsculas:

John Lee Hooker

Pero los segundos nombres no siempre son necesarios, y esta función, tal como está escrita, no funcionaría si intentaras llamarla sólo con un nombre y un apellido. Para que el segundo nombre sea opcional, podemos dar al argumento middle\_name un valor por defecto vacío e ignorar el argumento a menos que el usuario proporcione un valor. Para que get\_formatted\_name() funcione sin segundo nombre, establecemos el valor por defecto de middle\_name en una cadena vacía y lo movemos al final de la lista de parámetros:

def get\_formatted\_name(first\_name, last\_name, middle\_name=''):  
 """Return a full name, neatly formatted."""  
❶ if middle\_name:  
 full\_name = f"{first\_name} {middle\_name} {last\_name}"  
❷ else:  
 full\_name = f"{first\_name} {last\_name}"  
 return full\_name.title()  
  
musician = get\_formatted\_name('jimi', 'hendrix')  
print(musician)  
  
❸ musician = get\_formatted\_name('john', 'hooker', 'lee')  
print(musician)

En este ejemplo, el nombre se construye a partir de tres partes posibles. Como siempre hay un nombre y un apellido, estos parámetros aparecen primero en la definición de la función. El segundo nombre es opcional, por lo que aparece el último en la definición, y su valor por defecto es una cadena vacía.

En el cuerpo de la función, comprobamos si se ha proporcionado un segundo nombre. Python interpreta las cadenas no vacías como True, por lo que la prueba condicional if middle\_name se evalúa como True si hay un argumento de segundo nombre en la llamada a la función ❶. Si se proporciona un segundo nombre, el nombre, el segundo nombre y los apellidos se combinan para formar un nombre completo. Este nombre se cambia a mayúsculas y se devuelve a la línea de llamada a la función, donde se asigna a la variable musician y se imprime. Si no se proporciona el segundo nombre, la cadena vacía falla la prueba if y el bloque else se ejecuta ❷. El nombre completo se hace sólo con el nombre y los apellidos, y el nombre formateado se devuelve a la línea de llamada, donde se asigna a musician y se imprime.

Llamar a esta función con un nombre y apellidos es sencillo. Sin embargo, si utilizamos un segundo nombre, tenemos que asegurarnos de que el segundo nombre es el último argumento pasado para que Python haga coincidir correctamente los argumentos posicionales ❸.

Esta versión modificada de nuestra función funciona para las personas que sólo tienen nombre y apellidos, y también para las que tienen segundo nombre:

Jimi Hendrix  
John Lee Hooker

Los valores opcionales permiten a las funciones manejar una amplia gama de casos de uso, al tiempo que permiten que las llamadas a la función sigan siendo lo más sencillas posible.

[anterior](c08_17.html)[Subtema 18 de 40: (Ver todo)](c08.html)[siguiente](c08_19.html)