### Argumentos posicionales

Cuando llamas a una función, Python debe hacer coincidir cada argumento de la llamada a la función con un parámetro de la definición de la función. La forma más sencilla de hacerlo es basándose en el orden de los argumentos proporcionados. Los valores emparejados de esta forma se denominan *positional arguments*.

Para ver cómo funciona esto, considera una función que muestra información sobre mascotas. La función nos dice qué tipo de animal es cada mascota y el nombre de la mascota, como se muestra aquí:

**pets.py**

❶ def describe\_pet(animal\_type, pet\_name):  
 """Display information about a pet."""  
 print(f"\nI have a {animal\_type}.")  
 print(f"My {animal\_type}'s name is {pet\_name.title()}.")  
  
❷ describe\_pet('hamster', 'harry')

La definición muestra que esta función necesita un tipo de animal y el nombre del animal ❶. Cuando llamamos a describe\_pet(), necesitamos proporcionar un tipo de animal y un nombre, en ese orden. Por ejemplo, en la llamada a la función, el argumento 'hamster' se asigna al parámetro animal\_type y el argumento 'harry' se asigna al parámetro pet\_name ❷. En el cuerpo de la función, estos dos parámetros se utilizan para mostrar información sobre la mascota que se está describiendo.

La salida describe un hámster llamado Harry:

I have a hamster.  
My hamster's name is Harry.

#### Llamadas múltiples a una función

Puedes llamar a una función tantas veces como sea necesario. La descripción de una segunda mascota distinta sólo requiere una llamada más a describe\_pet():

def describe\_pet(animal\_type, pet\_name):  
 """Display information about a pet."""  
 print(f"\nI have a {animal\_type}.")  
 print(f"My {animal\_type}'s name is {pet\_name.title()}.")  
  
describe\_pet('hamster', 'harry')  
describe\_pet('dog', 'willie')

En esta segunda llamada a la función, pasamos a describe\_pet() los argumentos 'dog' y 'willie'. Al igual que con el anterior conjunto de argumentos que utilizamos, Python hace coincidir 'dog' con el parámetro animal\_type y 'willie' con el parámetro pet\_name. Como antes, la función hace su trabajo, pero esta vez imprime los valores de un perro llamado Willie. Ahora tenemos un hámster llamado Harry y un perro llamado Willie:

I have a hamster.  
My hamster's name is Harry.  
  
I have a dog.  
My dog's name is Willie.

Llamar a una función varias veces es una forma muy eficaz de trabajar. El código que describe una mascota se escribe una vez en la función. Luego, cada vez que quieras describir una nueva mascota, llamas a la función con la información de la nueva mascota. Aunque el código para describir una mascota se ampliara a 10 líneas, podrías describir una nueva mascota en una sola línea llamando de nuevo a la función.

#### El orden importa en los argumentos posicionales

Puedes obtener resultados inesperados si confundes el orden de los argumentos en una llamada a función cuando utilizas argumentos posicionales:

def describe\_pet(animal\_type, pet\_name):  
 """Display information about a pet."""  
 print(f"\nI have a {animal\_type}.")  
 print(f"My {animal\_type}'s name is {pet\_name.title()}.")  
  
describe\_pet('harry', 'hamster')

En esta llamada a la función, enumeramos primero el nombre y después el tipo de animal. Como el argumento 'harry' aparece primero esta vez, ese valor se asigna al parámetro animal\_type. Del mismo modo, 'hamster' se asigna a pet\_name. Ahora tenemos un "harry" llamado "Hamster":

I have a harry.  
My harry's name is Hamster.

Si obtienes resultados raros como éste, comprueba que el orden de los argumentos en tu llamada a la función coincide con el orden de los parámetros en la definición de la función.

[anterior](c08_7.html)[Subtema 8 de 40: (Ver todo)](c08.html)[siguiente](c08_9.html)