## Pasar un número arbitrario de argumentos

A veces no sabrás de antemano cuántos argumentos debe aceptar una función. Afortunadamente, Python permite que una función recoja un número arbitrario de argumentos de la sentencia que la llama.

Por ejemplo, considera una función que construye una pizza. Necesita aceptar un número de ingredientes, pero no puedes saber de antemano cuántos ingredientes querrá una persona. La función del siguiente ejemplo tiene un parámetro, \*toppings, pero este parámetro recoge tantos argumentos como proporcione la línea de llamada:

**pizza.py**

def make\_pizza(\*toppings):  
 """Print the list of toppings that have been requested."""  
 print(toppings)  
  
make\_pizza('pepperoni')  
make\_pizza('mushrooms', 'green peppers', 'extra cheese')

El asterisco en el nombre del parámetro \*toppings indica a Python que cree una tupla llamada toppings, que contenga todos los valores que recibe esta función. La llamada print() en el cuerpo de la función produce una salida que muestra que Python puede manejar una llamada a la función con un valor y una llamada con tres valores. Trata las distintas llamadas de forma similar. Observa que Python empaqueta los argumentos en una tupla, aunque la función sólo reciba un valor:

('pepperoni',)  
('mushrooms', 'green peppers', 'extra cheese')

Ahora podemos sustituir la llamada a print() por un bucle que recorra la lista de ingredientes y describa la pizza que se pide:

def make\_pizza(\*toppings):  
 """Summarize the pizza we are about to make."""  
 print("\nMaking a pizza with the following toppings:")  
 for topping in toppings:  
 print(f"- {topping}")  
  
make\_pizza('pepperoni')  
make\_pizza('mushrooms', 'green peppers', 'extra cheese')

La función responde adecuadamente, tanto si recibe un valor como tres:

Making a pizza with the following toppings:  
- pepperoni  
  
Making a pizza with the following toppings:  
- mushrooms  
- green peppers  
- extra cheese

Esta sintaxis funciona independientemente del número de argumentos que reciba la función.

[anterior](c08_25.html)[Subtema 26 de 40: (Ver todo)](c08.html)[siguiente](c08_27.html)