### Modificar una lista en una función

Cuando pasas una lista a una función, la función puede modificar la lista. Cualquier cambio realizado en la lista dentro del cuerpo de la función es permanente, lo que te permite trabajar con eficacia incluso cuando manejas grandes cantidades de datos.

Piensa en una empresa que crea modelos impresos en 3D de diseños que envían los usuarios. Los diseños que necesitan ser impresos se almacenan en una lista, y después de ser impresos se mueven a otra lista. El siguiente código hace esto sin utilizar funciones:

**printing\_models.py**

# Start with some designs that need to be printed.  
unprinted\_designs = ['phone case', 'robot pendant', 'dodecahedron']  
completed\_models = []  
  
# Simulate printing each design, until none are left.  
# Move each design to completed\_models after printing.  
while unprinted\_designs:  
 current\_design = unprinted\_designs.pop()  
 print(f"Printing model: {current\_design}")  
 completed\_models.append(current\_design)  
  
# Display all completed models.  
print("\nThe following models have been printed:")  
for completed\_model in completed\_models:  
 print(completed\_model)

Este programa comienza con una lista de diseños que necesitan ser impresos y una lista vacía llamada completed\_models a la que se moverá cada diseño después de ser impreso. Mientras los diseños permanezcan en unprinted\_designs, el bucle while simula la impresión de cada diseño eliminando un diseño del final de la lista, almacenándolo en current\_design, y mostrando un mensaje de que se está imprimiendo el diseño actual. A continuación, añade el diseño a la lista de modelos completados. Cuando el bucle termina de ejecutarse, se muestra una lista de los diseños que se han impreso:

Printing model: dodecahedron  
Printing model: robot pendant  
Printing model: phone case  
  
The following models have been printed:  
dodecahedron  
robot pendant  
phone case

Podemos reorganizar este código escribiendo dos funciones, cada una de las cuales realiza un trabajo específico. La mayor parte del código no cambiará; sólo lo estamos estructurando con más cuidado. La primera función se encargará de imprimir los diseños, y la segunda resumirá las impresiones realizadas:

❶ def print\_models(unprinted\_designs, completed\_models):  
 """  
 Simulate printing each design, until none are left.  
 Move each design to completed\_models after printing.  
 """  
 while unprinted\_designs:  
 current\_design = unprinted\_designs.pop()  
 print(f"Printing model: {current\_design}")  
 completed\_models.append(current\_design)  
  
❷ def show\_completed\_models(completed\_models):  
 """Show all the models that were printed."""  
 print("\nThe following models have been printed:")  
 for completed\_model in completed\_models:  
 print(completed\_model)  
  
unprinted\_designs = ['phone case', 'robot pendant', 'dodecahedron']  
completed\_models = []  
  
print\_models(unprinted\_designs, completed\_models)  
show\_completed\_models(completed\_models)

Definimos la función print\_models() con dos parámetros: una lista de diseños que hay que imprimir y una lista de modelos completados ❶. Dadas estas dos listas, la función simula la impresión de cada diseño vaciando la lista de diseños sin imprimir y rellenando la lista de modelos completados. A continuación, definimos la función show\_completed\_models() con un parámetro: la lista de modelos completados ❷. Dada esta lista, show\_completed\_models() muestra el nombre de cada modelo que se imprimió.

Este programa tiene la misma salida que la versión sin funciones, pero el código está mucho más organizado. El código que hace la mayor parte del trabajo se ha trasladado a dos funciones separadas, lo que hace que la parte principal del programa sea más fácil de entender. Mira el cuerpo del programa y fíjate en lo fácil que es seguir lo que ocurre:

unprinted\_designs = ['phone case', 'robot pendant', 'dodecahedron']  
completed\_models = []  
  
print\_models(unprinted\_designs, completed\_models)  
show\_completed\_models(completed\_models)

Creamos una lista de diseños sin imprimir y una lista vacía que contendrá los modelos terminados. Luego, como ya hemos definido nuestras dos funciones, todo lo que tenemos que hacer es llamarlas y pasarles los argumentos adecuados. Llamamos a print\_models() y le pasamos las dos listas que necesita; como era de esperar, print\_models() simula la impresión de los diseños. Luego llamamos a show\_completed\_models() y le pasamos la lista de modelos completados para que nos informe de los modelos que se han impreso. Los nombres descriptivos de las funciones permiten que otros lean este código y lo entiendan, incluso sin comentarios.

Este programa es más fácil de ampliar y mantener que la versión sin funciones. Si más adelante necesitamos imprimir más diseños, basta con volver a llamar a print\_models(). Si nos damos cuenta de que hay que modificar el código de impresión, podemos cambiar el código una vez, y nuestros cambios tendrán lugar en todos los lugares donde se llame a la función. Esta técnica es más eficaz que tener que actualizar el código por separado en varios lugares del programa.

Este ejemplo también demuestra la idea de que cada función debe tener un trabajo específico. La primera función imprime cada diseño, y la segunda muestra los modelos completados. Esto es más beneficioso que utilizar una función para hacer ambos trabajos. Si estás escribiendo una función y notas que realiza demasiadas tareas diferentes, intenta dividir el código en dos funciones. Recuerda que siempre puedes llamar a una función desde otra función, lo que puede ser útil al dividir una tarea compleja en una serie de pasos.

[anterior](c08_22.html)[Subtema 23 de 40: (Ver todo)](c08.html)[siguiente](c08_24.html)