### Copiar una lista

A menudo, querrás empezar con una lista existente y hacer una lista completamente nueva basada en la primera. Exploremos cómo funciona copiar una lista y examinemos una situación en la que copiar una lista resulta útil.

Para copiar una lista, puedes hacer un corte que incluya toda la lista original omitiendo el primer índice y el segundo índice ([:]). Esto le dice a Python que haga un corte que empiece en el primer elemento y termine en el último, produciendo una copia de toda la lista.

Por ejemplo, imagina que tenemos una lista de nuestras comidas favoritas y queremos hacer otra lista de las comidas que le gustan a un amigo. A este amigo le gusta todo lo que hay en nuestra lista hasta ahora, así que podemos crear su lista copiando la nuestra:

**foods.py**

my\_foods = ['pizza', 'falafel', 'carrot cake']  
❶ friend\_foods = my\_foods[:]  
  
print("My favorite foods are:")  
print(my\_foods)  
  
print("\nMy friend's favorite foods are:")  
print(friend\_foods)

Primero, hacemos una lista de los alimentos que nos gustan llamada my\_foods. Luego hacemos una nueva lista llamada friend\_foods. Hacemos una copia de my\_foods pidiendo un trozo de my\_foods sin especificar ningún índice ❶, y asignamos la copia a friend\_foods. Cuando imprimimos cada lista, vemos que ambas contienen los mismos alimentos:

My favorite foods are:  
['pizza', 'falafel', 'carrot cake']  
  
My friend's favorite foods are:  
['pizza', 'falafel', 'carrot cake']

Para demostrar que en realidad tenemos dos listas separadas, añadiremos un nuevo alimento a cada lista y demostraremos que cada lista lleva la cuenta de los alimentos favoritos de la persona correspondiente:

my\_foods = ['pizza', 'falafel', 'carrot cake']  
❶ friend\_foods = my\_foods[:]  
  
❷ my\_foods.append('cannoli')  
❸ friend\_foods.append('ice cream')  
  
print("My favorite foods are:")  
print(my\_foods)  
  
print("\nMy friend's favorite foods are:")  
print(friend\_foods)

Copiamos los elementos originales de my\_foods en la nueva lista friend\_foods, como hicimos en el ejemplo anterior ❶. A continuación, añadimos un nuevo alimento a cada lista: añadimos 'cannoli' a my\_foods ❷, y añadimos 'ice cream' a friend\_foods ❸. A continuación, imprimimos las dos listas para ver si cada uno de estos alimentos está en la lista adecuada:

My favorite foods are:  
['pizza', 'falafel', 'carrot cake', 'cannoli']  
  
My friend's favorite foods are:  
['pizza', 'falafel', 'carrot cake', 'ice cream']

La salida muestra que 'cannoli' aparece ahora en nuestra lista de alimentos favoritos, pero 'ice cream' no. Podemos ver que 'ice cream' aparece ahora en la lista de nuestro amigo pero 'cannoli' no. Si simplemente hubiéramos establecido friend\_foods igual a my\_foods, no produciríamos dos listas separadas. Por ejemplo, esto es lo que ocurre cuando intentas copiar una lista sin utilizar un corte:

my\_foods = ['pizza', 'falafel', 'carrot cake']  
  
# This doesn't work:  
friend\_foods = my\_foods  
  
my\_foods.append('cannoli')  
friend\_foods.append('ice cream')  
  
print("My favorite foods are:")  
print(my\_foods)  
  
print("\nMy friend's favorite foods are:")  
print(friend\_foods)

En lugar de asignar una copia de my\_foods a friend\_foods, establecemos friend\_foods igual a my\_foods. Esta sintaxis en realidad le dice a Python que asocie la nueva variable friend\_foods a la lista que ya está asociada a my\_foods, de modo que ahora ambas variables apuntan a la misma lista. Como resultado, cuando añadamos 'cannoli' a my\_foods, también aparecerá en friend\_foods. Del mismo modo, 'ice cream' aparecerá en ambas listas, aunque parezca que sólo se ha añadido a friend\_foods.

La salida muestra que ahora ambas listas son iguales, que no es lo que queríamos:

My favorite foods are:  
['pizza', 'falafel', 'carrot cake', 'cannoli', 'ice cream']  
  
My friend's favorite foods are:  
['pizza', 'falafel', 'carrot cake', 'cannoli', 'ice cream']

[anterior](c04_23.html)[Subtema 24 de 41: (Ver todo)](c04.html)[siguiente](c04_25.html)