### Una prueba que pasa

Con pytest, escribir tu primera prueba unitaria es bastante sencillo. Escribiremos una única función de prueba. La función de prueba llamará a la función que estamos probando y haremos una afirmación sobre el valor devuelto. Si nuestra afirmación es correcta, la prueba pasará; si la afirmación es incorrecta, la prueba fallará.

Aquí tienes la primera prueba de la función get\_formatted\_name():

**test\_name\_function.py**

from name\_function import get\_formatted\_name  
  
❶ def test\_first\_last\_name():  
 """Do names like 'Janis Joplin' work?"""  
❷ formatted\_name = get\_formatted\_name('janis', 'joplin')  
❸ assert formatted\_name == 'Janis Joplin'

Antes de ejecutar la prueba, echemos un vistazo más de cerca a esta función. El nombre del archivo de prueba es importante; debe empezar por *test\_*. Cuando le pidamos a pytest que ejecute las pruebas que hemos escrito, buscará cualquier archivo que empiece por *test\_*, y ejecutará todas las pruebas que encuentre en ese archivo.

En el archivo de prueba, primero importamos la función que queremos probar: get\_formatted\_name(). Después definimos una función de prueba: en este caso, test\_first\_last\_name() ❶. Se trata de un nombre de función más largo que el que hemos estado utilizando, por una buena razón. En primer lugar, las funciones de prueba deben empezar por la palabra *test*, seguida de un guión bajo. Cualquier función que empiece por test\_ será *discovered* por pytest, y se ejecutará como parte del proceso de prueba.

Además, los nombres de las pruebas deben ser más largos y descriptivos que el nombre típico de una función. Nunca llamarás a la función tú mismo; pytest encontrará la función y la ejecutará por ti. Los nombres de las funciones de prueba deben ser lo suficientemente largos como para que, si ves el nombre de la función en un informe de prueba, tengas una buena idea de qué comportamiento se estaba probando.

A continuación, llamamos a la función que estamos probando ❷. Aquí llamamos a get\_formatted\_name() con los argumentos 'janis' y 'joplin', igual que utilizamos cuando ejecutamos *names.py*. Asignamos el valor de retorno de esta función a formatted\_name.

Por último, hacemos una afirmación ❸. Una *assertion* es una afirmación sobre una condición. Aquí afirmamos que el valor de formatted\_name debe ser 'Janis Joplin'.

[anterior](c11_8.html)[Subtema 9 de 22: (Ver todo)](c11.html)[siguiente](c11_10.html)