### Utilizar dispositivos

En *test\_survey.py*, creamos una nueva instancia de AnonymousSurvey en cada función de prueba. Esto está bien en el breve ejemplo con el que estamos trabajando, pero en un proyecto del mundo real con decenas o cientos de pruebas, esto sería problemático.

En las pruebas, un *fixture* ayuda a configurar un entorno de pruebas. A menudo, esto significa crear un recurso que es utilizado por más de una prueba. Creamos un fixture en pytest escribiendo una función con el decorador @pytest.fixture. Un *decorator* es una directiva que se coloca justo antes de la definición de una función; Python aplica esta directiva a la función antes de que se ejecute, para alterar cómo se comporta el código de la función. No te preocupes si esto te parece complicado; puedes empezar a utilizar decoradores de paquetes de terceros antes de aprender a escribirlos tú mismo.

Utilicemos un fixture para crear una única instancia de encuesta que pueda utilizarse en ambas funciones de prueba en *test\_survey.py*:

import pytest  
from survey import AnonymousSurvey  
  
❶ @pytest.fixture  
❷ def language\_survey():  
 """A survey that will be available to all test functions."""  
 question = "What language did you first learn to speak?"  
 language\_survey = AnonymousSurvey(question)  
 return language\_survey  
  
❸ def test\_store\_single\_response(language\_survey):  
 """Test that a single response is stored properly."""  
❹ language\_survey.store\_response('English')  
 assert 'English' in language\_survey.responses  
  
❺ def test\_store\_three\_responses(language\_survey):  
 """Test that three individual responses are stored properly."""  
 responses = ['English', 'Spanish', 'Mandarin']  
 for response in responses:  
❻ language\_survey.store\_response(response)  
  
 for response in responses:  
 assert response in language\_survey.responses

Ahora necesitamos importar pytest, porque estamos utilizando un decorador que está definido en pytest. Aplicamos el decorador @pytest.fixture ❶ a la nueva función language\_survey() ❷. Esta función construye un objeto AnonymousSurvey y devuelve la nueva encuesta.

Observa que las definiciones de ambas funciones de prueba han cambiado ❸❺; cada función de prueba tiene ahora un parámetro llamado language\_survey. Cuando un parámetro de una función de prueba coincida con el nombre de una función con el decorador @pytest.fixture, el accesorio se ejecutará automáticamente y el valor devuelto se pasará a la función de prueba. En este ejemplo, la función language\_survey() suministra tanto a test\_store\_single\_response() como a test\_store\_three\_responses() una instancia de language\_survey.

No hay código nuevo en ninguna de las dos funciones de prueba, pero fíjate en que se han eliminado dos líneas de cada función ❹❻: la línea que definía una pregunta y la que creaba un objeto AnonymousSurvey.

Cuando volvemos a ejecutar el archivo de prueba, ambas pruebas siguen pasando. Estas pruebas serían especialmente útiles si intentáramos ampliar AnonymousSurvey para gestionar múltiples respuestas para cada persona. Después de modificar el código para que acepte múltiples respuestas, podrías ejecutar estas pruebas y asegurarte de que no has afectado a la capacidad de almacenar una sola respuesta o una serie de respuestas individuales.

Es casi seguro que la estructura anterior te parecerá complicada; contiene parte del código más abstracto que has visto hasta ahora. No es necesario que utilices fixtures de inmediato; es mejor escribir pruebas con mucho código repetitivo que no escribir ninguna prueba. Sólo tienes que saber que cuando hayas escrito suficientes pruebas como para que la repetición se interponga en tu camino, existe una forma bien establecida de tratar la repetición. Además, en ejemplos sencillos como éste, las fijaciones no hacen que el código sea más corto ni más fácil de seguir. Pero en proyectos con muchas pruebas, o en situaciones en las que se necesitan muchas líneas para construir un recurso que se utiliza en varias pruebas, las fijaciones pueden mejorar drásticamente tu código de prueba.

Cuando quieras escribir un fixture, escribe una función que genere el recurso que utilizan varias funciones de prueba. Añade el decorador @pytest.fixture  a la nueva función, y añade el nombre de esta función como parámetro para cada función de prueba que utilice este recurso. A partir de ese momento, tus pruebas serán más cortas y más fáciles de escribir y mantener.

[anterior](c11_19.html)[Subtema 20 de 22: (Ver todo)](c11.html)[siguiente](c11_21.html)