### Utilizar un indicador

En el ejemplo anterior, hicimos que el programa realizara determinadas tareas mientras una condición dada fuera cierta. Pero, ¿qué ocurre con programas más complicados en los que muchos sucesos diferentes pueden hacer que el programa deje de ejecutarse?

Por ejemplo, en un juego, varios sucesos diferentes pueden poner fin a la partida. Cuando el jugador se queda sin naves, se le acaba el tiempo o las ciudades que debía proteger son destruidas, el juego debe terminar. Debe terminar si ocurre cualquiera de estos acontecimientos. Si pueden ocurrir muchos eventos posibles para detener el programa, intentar probar todas estas condiciones en una sentencia while se vuelve complicado y difícil.

Para un programa que debe ejecutarse sólo mientras muchas condiciones sean verdaderas, puedes definir una variable que determine si todo el programa está activo o no. Esta variable, llamada *flag*, actúa como una señal para el programa. Podemos escribir nuestros programas de modo que se ejecuten mientras la bandera esté establecida en True y dejen de ejecutarse cuando alguno de varios eventos establezca el valor de la bandera en False. Como resultado, nuestra declaración global while sólo necesita comprobar una condición: si la bandera está actualmente en True. Entonces, todas nuestras demás pruebas (para ver si se ha producido un evento que debería poner la bandera en False) pueden organizarse ordenadamente en el resto del programa.

Añadamos una bandera a *parrot.py* de la sección anterior. Esta bandera, que llamaremos active (aunque puedes llamarla como quieras), controlará si el programa debe seguir ejecutándose o no:

prompt = "\nTell me something, and I will repeat it back to you:"  
prompt += "\nEnter 'quit' to end the program. "  
  
active = True  
❶ while active:  
 message = input(prompt)  
  
 if message == 'quit':  
 active = False  
 else:  
 print(message)

Establecemos la variable active en True para que el programa comience en un estado activo. Hacer esto simplifica la sentencia while porque no se realiza ninguna comparación en la propia sentencia while; de la lógica se encargan otras partes del programa. Mientras la variable active siga siendo True, el bucle seguirá ejecutándose ❶.

En la sentencia if dentro del bucle while, comprobamos el valor de message una vez que el usuario introduce su entrada. Si el usuario introduce 'quit', establecemos active en False, y el bucle while se detiene. Si el usuario introduce algo distinto de 'quit', imprimimos su entrada como un mensaje.

Este programa tiene la misma salida que el ejemplo anterior, en el que colocamos la prueba condicional directamente en la sentencia while. Pero ahora que tiene una bandera para indicar si el programa en general está en estado activo, sería fácil añadir más pruebas (como las sentencias elif ) para los eventos que deberían hacer que active se convirtiera en False. Esto es útil en programas complicados como los juegos, en los que puede haber muchos eventos que deberían hacer que el programa dejara de ejecutarse. Cuando alguno de estos eventos haga que la bandera activa se convierta en False, el bucle principal del juego saldrá, se podrá mostrar un mensaje *Game Over* y se podrá dar al jugador la opción de volver a jugar.

[anterior](c07_10.html)[Subtema 11 de 23: (Ver todo)](c07.html)[siguiente](c07_12.html)