### El método \_check\_events()

Trasladaremos el código que gestiona los eventos a un método independiente llamado \_check\_events(). Esto simplificará run\_game() y aislará el bucle de gestión de eventos. Aislar el bucle de eventos te permite gestionar los eventos por separado de otros aspectos del juego, como la actualización de la pantalla.

Aquí tienes la clase AlienInvasion con el nuevo método \_check\_events(), que sólo afecta al código de run\_game():

**alien\_invasion.py**

def run\_game(self):  
 """Start the main loop for the game."""  
 while True:  
❶ self.\_check\_events()  
  
 # Redraw the screen during each pass through the loop.  
 --snip--  
  
❷ def \_check\_events(self):  
 """Respond to keypresses and mouse events."""  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 sys.exit()

Creamos un nuevo método \_check\_events() ❷ y trasladamos a este nuevo método las líneas que comprueban si el jugador ha hecho clic para cerrar la ventana.

Para llamar a un método desde dentro de una clase, utiliza la notación con puntos con la variable self y el nombre del método ❶. Llamamos al método desde dentro del bucle while en run\_game().

[anterior](c12_15.html)[Subtema 16 de 43: (Ver todo)](c12.html)[siguiente](c12_17.html)