### Utilizar excepciones para evitar fallos

Manejar los errores correctamente es especialmente importante cuando el programa tiene más trabajo que hacer después de que se produzca el error. Esto ocurre a menudo en los programas que piden a los usuarios que introduzcan datos. Si el programa responde adecuadamente a las entradas no válidas, puede pedir más entradas válidas en lugar de bloquearse.

Creemos una calculadora sencilla que sólo haga divisiones:

**division\_calculator.py**

print("Give me two numbers, and I'll divide them.")  
print("Enter 'q' to quit.")  
  
while True:  
❶ first\_number = input("\nFirst number: ")  
 if first\_number == 'q':  
 break  
❷ second\_number = input("Second number: ")  
 if second\_number == 'q':  
 break  
❸ answer = int(first\_number) / int(second\_number)  
 print(answer)

Este programa pide al usuario que introduzca un first\_number ❶ y, si no introduce q para salir, un second\_number ❷. A continuación, dividimos estos dos números para obtener un answer ❸. Este programa no hace nada para manejar los errores, por lo que pedirle que divida por cero hace que se bloquee:

Give me two numbers, and I'll divide them.  
Enter 'q' to quit.  
  
First number: 5  
Second number: 0  
Traceback (most recent call last):  
 File "division\_calculator.py", line 11, in <module>  
 answer = int(first\_number) / int(second\_number)  
 ~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~  
ZeroDivisionError: division by zero

Es malo que el programa se bloquee, pero tampoco es buena idea dejar que los usuarios vean las trazas. Los usuarios no técnicos se sentirán confundidos por ellos, y en un entorno malicioso, los atacantes aprenderán más de lo que tú quieres. Por ejemplo, sabrán el nombre del archivo de tu programa y verán una parte de tu código que no funciona correctamente. A veces, un atacante hábil puede utilizar esta información para determinar qué tipo de ataques utilizar contra tu código.

[anterior](c10_22.html)[Subtema 23 de 37: (Ver todo)](c10.html)[siguiente](c10_24.html)