Desconocido

### Problemas con las cadenas

Considera ahora este tonto ejemplo de una cadena, que hace que Python muestre un mensaje de error:

>>> silly\_string = 'He said, "Aren't can't shouldn't wouldn't."'  
SyntaxError: invalid syntax

En la primera línea, intentamos crear una cadena (definida como la variable silly\_string ) encerrada entre comillas simples, pero que también contiene una mezcla de comillas simples en las palabras can’t , shouldn’t , y wouldn’t , así como comillas dobles. ¡Menudo lío!

Recuerda que Python no es tan inteligente como un ser humano, así que todo lo que ve es una cadena que contiene He said, "Aren , seguida de un montón de otros caracteres que no espera. Cuando Python ve una comilla (ya sea simple o doble), espera que una cadena comience tras la primera comilla y que la cadena termine tras la siguiente comilla que coincida  (ya sea simple o doble) en esa línea. En este caso, el inicio de la cadena es la comilla simple antes de He , y el final de la cadena, en lo que respecta a Python, es la comilla simple después de n en Aren .

En la última línea, Python nos dice qué tipo de error se ha producido, en este ejemplo, un error de sintaxis.

Utilizar comillas dobles en lugar de simples también produce un error:

>>> silly\_string = "He said, "Aren't can't shouldn't wouldn't.""  
SyntaxError: invalid syntax

Aquí, Python ve una cadena entre comillas dobles, que contiene las letras He said, (y un espacio). Todo lo que sigue a esa cadena (a partir de Aren’t ) provoca el error.

Esto se debe a que, desde la perspectiva de Python, todo ese material extra no debería estar ahí. Python busca la siguiente comilla coincidente y no sabe qué quieres que haga con todo lo que sigue a esa comilla en la misma línea.

La solución a este problema es una cadena multilínea, de la que aprendimos antes, utilizando *tres* comillas simples ( ''' ). Esto nos permite combinar comillas dobles y simples en nuestra cadena sin provocar errores. De hecho, si utilizamos tres comillas simples, podemos poner cualquier combinación de comillas simples y dobles dentro de la cadena (siempre que no intentemos poner tres comillas simples). La versión sin errores de nuestra cadena tiene este aspecto:



silly\_string = '''He said, "Aren't can't shouldn't wouldn't."'''

Pero espera, hay más. Si realmente quieres utilizar comillas simples o dobles para rodear una cadena en Python, en lugar de tres comillas simples, puedes añadir una barra invertida (∖) antes de cada comilla dentro de la cadena. Esto se llama *escape .* Es una forma de decirle a Python: "Sí, sé que tengo comillas dentro de mi cadena, y quiero que las ignores hasta que veas la comilla final".

Escapar cadenas puede dificultar su lectura, por lo que se considera mejor práctica utilizar cadenas multilínea. Aun así, es posible que en  te encuentres con código que utiliza escapes, por lo que es útil saber por qué están ahí las barras invertidas.

Aquí tienes algunos ejemplos de cómo funciona el escape:

➊ >>> single\_quote\_str = 'He said, "Aren\'t can\'t shouldn\'t  
 wouldn\'t."'  
➋ >>> double\_quote\_str = "He said, \"Aren't can't shouldn't   
 wouldn't.\""  
 >>> print(single\_quote\_str)  
 He said, "Aren't can't shouldn't wouldn't."  
 >>> print(double\_quote\_str)  
 He said, "Aren't can't shouldn't wouldn't."

Primero, en ➊ , creamos una cadena con comillas simples, utilizando la barra invertida delante de las comillas simples dentro de esa cadena. En ➋ , creamos una cadena con comillas dobles, y utilizamos la barra invertida delante de esas comillas en la cadena. En las líneas siguientes, imprimimos las variables que acabamos de crear. Observa que el carácter barra invertida no aparece en las cadenas cuando las imprimimos.

[anterior](ch03_3.html)[Subtema 4 de 14: (Ver todo)](ch03.html)[siguiente](ch03_5.html)