Cadenas de texto

Introducción

Una cadena es una secuencia **inmutable** de caracteres Unicode, delimitada por comillas.

Comillas simples y dobles

Las cadenas de texto se pueden delimitar con comillas simples (') o con comillas dobles ("):

```
>>> print('Esto es una cadena')
Esto es una cadena
>>> print("Esto es una cadena")
Esto es una cadena
```

La función `print()` muestra por pantalla el contenido de la cadena, pero no las comillas delimitadoras de las cadenas.

En Python las comillas dobles y las comillas simples son completamente equivalentes, pero en otros
lenguajes de programación no lo son.

> En otros lenguajes de programación, por ejemplo en **PHP**, en las cadenas delimitadas con comillas dobles las variables se sustituyen por su valor y se pueden utilizar caracteres especiales,
> pero en las cadenas delimitadas con comillas simples, no.

Las cadenas se deben cerrar con las mismas comillas con las que se abrieron, de lo contrario estaremos cometiendo un error de sintaxis:

```
>>> print("Esto es una cadena')
SyntaxError: EOL while scanning string literal
>>>
```

Comillas triples

Las comillas triples permiten que las cadenas ocupen más de una línea:

```
>>> print("""Esto es una cadena
que ocupa
  varias líneas""")
Esto es una cadena
que ocupa
  varias líneas
```

Pero las comillas triples se utilizan sobre todo con una finalidad específica: la **documentación** de módulos, funciones, clases o métodos. Son las llamadas **docstrings**. Son cadenas que se escriben al principio del elemento describiendo lo que hace el elemento. No producen ningún resultado en el programa, pero las herramientas de documentación de Python pueden extraerlas para generar documentación automáticamente.

```
def licencia():
    """Escribe la licencia de estos apuntes"""
    print("Copyright 2013 Bartolomé Sintes Marco")
    print("Licencia CC-BY-SA 4.0")
    return
```

Comillas dentro de comillas

Se pueden escribir comillas simples en cadenas delimitadas con comillas dobles y viceversa:

```
>>> print("Las comillas simples ' delimitan cadenas.")
Las comillas simples ' delimitan cadenas.
>>> print('Las comillas dobles " delimitan cadenas.')
Las comillas dobles " delimitan cadenas.
```

Pero no se pueden escribir en el interior de una cadena comillas del mismo tipo que las comillas delimitadoras:

```
>>> print("Las comillas dobles " delimitan cadenas")
SyntaxError: invalid syntax
>>>
>>> print('Las comillas simples ' delimitan
cadenas')
SyntaxError: invalid syntax
>>>
```

El motivo es que Python entiende que la comilla escrita en medio de la cadena está cerrando la cadena y no puede interpretar lo que viene a continuación.

Otra forma de escribir comillas en una cadena es utilizar los caracteres especiales `\"` y `\'` que representan los caracteres comillas dobles y simples respectivamente y que Python no interpreta en ningún caso como delimitadores de cadena:

```
>>> print('Las comillas simples \' delimitan cadenas.')
Las comillas simples ' delimitan cadenas.
>>> print("Las comillas dobles \" delimitan cadenas.")
Las comillas dobles " delimitan cadenas.
```

Se pueden utilizar ambos caracteres especiales independientemente del delimitador utilizado

```
>>> print('Las comillas simples \' y las comillas dobles \" delimitan
cadenas.')
Las comillas simples ' y las comillas dobles " delimitan cadenas.
>>> print("Las comillas simples \' y las comillas dobles \" delimitan
cadenas.")
Las comillas simples ' y las comillas dobles " delimitan cadenas.
```

Caracteres especiales

Los caracteres especiales empiezan por una contrabarra ().

- Comilla doble: `\"`

```
>>> print("Las comillas dobles \" delimitan cadenas.")
Las comillas dobles " delimitan cadenas.
```

- Comilla simple: `\'`

```
>>> print('Las comillas simples \' delimitan cadenas.')
Las comillas simples ' delimitan cadenas.
```

- Salto de línea: `\n`

```
>>> print("Una línea\nOtra línea")
Una línea
Otra línea
```

- Tabulador: `\t`

```
>>> print("1\t2\t3")
1       2       3
```

Evaluación de cadenas desde el intérprete

En el prompt del intérprete se pueden escribir cadenas sueltas, sin necesidad de escribir la función `print()`. Al pulsar Intro, el intérprete evalúa la cadena y escribe el resultado, como hace cuando se escribe una expresión aritmética.

En la mayoría de los casos el intérprete escribe el resultado entre comillas simples, para indicar que se trata de una cadena.

```
>>> "Esto es una cadena"
'Esto es una cadena'
>>> 'Esto es una cadena'
'Esto es una cadena'
```

Si el resultado contiene únicamente comillas simples, el intérprete lo escribe entre comillas dobles.

```
>>> "Las comillas simples ' delimitan cadenas."
"Las comillas simples ' delimitan cadenas."
>>> 'Las comillas simples \' delimitan cadenas.'
"Las comillas dobles ' delimitan cadenas."
```

Pero si aparecen ambas comillas, el intérprete lo escribe entre comillas simples y las comillas simples se muestran como caracteres especiales.

```
>>> 'Las comillas simples \' y dobles \" delimitan cadenas.'
'Las comillas simples \' y dobles " delimitan cadenas.'
>>> "Las comillas simples \' y dobles \" delimitan cadenas."
'Las comillas simples \' y dobles " delimitan cadenas.'
```

Operaciones con cadenas

Concatenar

El operador `+` aplicado a cadenas de caracteres concatena las mismas.

```
>>> "uno" + "dos"
'unodos'
>>> 'uno' + 'dos'
'unodos'
>>> "uno" + " dos"
'uno dos'
>>> "uno," + " dos"
'uno, dos'
```

Se pueden concatenar varias cadenas en la misma expresión

```
>>> "uno," + " dos," + ' tres'
'uno, dos, tres'
```

Repetición

El operador de multiplicación `*` repite el número de veces que indiquemos la cadena.

```
>>> "cadena repetida " * 3
'cadena repetida cadena repetida cadena repetida '
>>> 2 * "cadena repetida "
'cadena repetida cadena repetida '
```

Cadenas en bruto (raw)

Una cadena en bruto es una cadena literal y que, por tanto, trata el caracter de escape `` \`` como un carácter literal y no aplica la operación asociada. Para que una cadena se muestre en bruto ponemos una `r` o una `R` antes de la primera comilla de la misma.

```
>>> print("1\t2\t3\n4\t5\t6")
1	2	3
4	5	6
>>> print(r"1\t2\t3\n4\t5\t6")
1\t2\t3\n4\t5\t6
```

Referencias

Apuntes generados a partir del curso Introducción a la programación con Python que se distribuye bajo una Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0).