**Cadenas de texto**

En esta lección se tratan el tipo de datos cadena de texto (string) en Python.

# Cadenas de texto

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(), que se comenta con más detalle en la lecció[n Salida por pantalla](https://www.mclibre.org/consultar/python/lecciones/python-salida-pantalla.html), 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. Aunque en la documentación de Python se utilizan casi siempre comillas simples, en estos apuntes se utilizan casi siempre las comillas dobles.

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

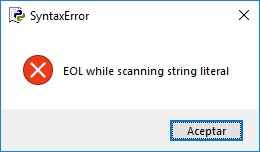
|  |
| --- |
| >>> print("Esto es una cadena')  SyntaxError: EOL while scanning string literal  >>> |

En un programa, al intentar ejecutar el programa se abriría una ventana de error:

print("Esto es una cadena')

|  |
| --- |
| File "prueba.py", line 1  print("Hola')  ^  SyntaxError: EOL while scanning string literal |

En IDLE se muestra una ventana indicando el error:



Sin necesidad de ejecutar la orden o el programa, la forma en que IDLE colorea la orden nos ayuda a identificar que hay algo incorrecto en el programa. En el ejemplo anterior, el paréntesis de cierre se muestra de color verde, lo que indica que IDLE no identifica la comilla simple como cierre de la cadena abierta con comilla doble.

## 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*, definidas en el [PEP 257](https://www.python.org/dev/peps/pep-0257/). 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. Sin necesidad de ejecutar la orden o el programa, la forma en que IDLE colorea la orden así nos lo indica.

En un programa, obtendríamos también una ventana de error de sintaxis. Dependiendo del número y la posición de las cadenas, el mensaje de error puede ser distinto:

print("Las comillas dobles " delimitan cadenas")

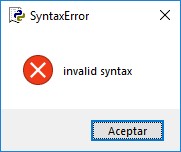
File "prueba.py", line 1

print("Las comillas dobles " delimitan cadenas")

^

SyntaxError: invalid syntaxFile "prueba.py", line 1

En IDLE se muestra una ventana indicando el error:



print('Un tipo le dice a otro: '¿Cómo estás?'')

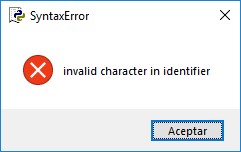
File "prueba.py", line 1

print('Un tipo le dice a otro: '¿Cómo estás?'')

^

SyntaxError: invalid character in identifier

En IDLE se muestra una ventana indicando el error:



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 en el prompt de IDLE

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

En la mayoría de los casos IDLE 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, IDLE 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, IDLE 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.'

# Cadenas largas

De acuerdo con la [guía de estilo oficial de Python](https://www.mclibre.org/consultar/python/otros/libro-de-estilo.html), las líneas de código no deben contener más de 79 caracteres, para facilitar la legibilidad.

Si un programa contiene cadenas muy largas, las cadenas se pueden simplemente partir en varias cadenas.

>>> print("Esta línea está cortada en dos líneas de menos de 79 caracteres "

"partiendo la cadena en dos")

Esta línea está cortada en dos líneas de menos de 79 caracteres partiendo la cadena en dos

También se puede escribir el carácter contrabarra (\) para partir una cadena en varias líneas.

|  |
| --- |
| >>> print("Esta línea está cortada en dos líneas de menos de 79 caracteres \ partiendo la cadena en dos")  Esta línea está cortada en dos líneas de menos de 79 caracteres partiendo la cadena en dos |

En los programas en los que hay bloques, la segunda notación hace más difícil la lectura del programa.

for i in range(3): print("a b c " "d e f ")

a b c d e f a b c d e f a b c d e f

for i in range(3): print("a b c \ d e f ")

a b c d e f a b c d e f a b c d e f

## Cadenas "f"

En Python 3.6 se añadió ([PEP 498](https://www.python.org/dev/peps/pep-0498/)) una nueva notación para cadenas llamada cadenas "f", que simplifica la inserción de variables y expresiones en las cadenas. Una cadena "f" contiene variables y expresiones entre llaves ({}) que se sustituyen directamente por su valor. Las cadenas "f" se reconocen porque comienzan por una letra f antes de las comillas de apertura.

nombre = "Pepe" edad = 25

print(f"Me llamo {nombre} y tengo {edad} años.")

Me llamo Pepe y tengo 25 años.

semanas = 4

print(f"En {semanas} semanas hay {7 \* semanas} días.")

En 4 semanas hay 28 días.

Esta notación no añade espacios que en la notación "clásica" aparecían al incluir varios argumentos separados por comas:

fecha = 2020

print("¡Feliz", fecha, "!") print(f"¡Feliz {fecha}!")

¡Feliz 2020 ! ¡Feliz 2020!

Si no se escribe la letra f antes de la cadena, Python no sustituye los valores de las variables ni calcula las expresiones.

nombre = "Pepe" edad = 25

print("Me llamo {nombre} y tengo {edad} años.")

Me llamo {nombre} y tengo {edad} años.

Si se quieren escribir los caracteres { o }, se deben escribir duplicados.

nombre = "Pepe" edad = 25 print(f"Si escribe {{nombre}} se escribirá el valor de la variable nombre, " f"en este caso {nombre}.")

Si escribe {nombre} se escribirá el valor de la variable nombre, en este caso Pepe.

Las cadenas "f" permiten dar formato a las variables, siguiendo la sintaxis d[e](https://docs.python.org/3/library/string.html#formatstrings) [método str.format()](https://docs.python.org/3/library/string.html#formatstrings)

## Operaciones

Las cadenas también admiten operadores aritméticos como los siguientes:

El operador *suma* para realizar concatenación de cadenas de caracteres:

**>>>** a, b = "uno", "dos"

**>>>** a + b

'unodos'

El operador *multiplicación* para repetir la cadena de caracteres por N veces definidas en la multiplicación:

**>>>** c = "tres"

**>>>** c \* 3

'trestrestres'

## 

## Comentarios

Son cadenas de caracteres las cuales constituyen una ayuda esencial tanto para quien está desarrollando el programa, como para otras personas que lean el código.

Los comentarios en el código tienen una vital importancia en el desarrollo de todo programa, algunas de las funciones más importantes que pueden cumplir los comentarios en un programa, son:

* Brindar información general sobre el programa.
* Explicar qué hace cada una de sus partes.
* Aclarar y/o fundamentar el funcionamiento de un bloque específico de código, que no sea evidente de su propia lectura.
* Indicar cosas pendientes para agregar o mejorar.

El signo para indicar el comienzo de un comentario en Python es el carácter numeral #, a partir del cual y hasta el fin de la línea, todo se considera un comentario y es ignorado por el intérprete Python.

**3.8. Tipo cadenas de caracteres**

**>>>** *# comentarios en linea* **...**

>>>

El carácter # puede estar al comienzo de línea (en cuyo caso toda la línea será ignorada), o después de finalizar una instrucción válida de código.

**>>>** *# Programa que calcula la sucesión*

**...** *# de números Fibonacci* **...**

**>>>** *# se definen las variables*

**...** a, b = 0, 1

**>>> while** b < 100: *# mientras b sea menor a 100 itere*

**... print** b,

**...** a, b = b, a + b *# se calcula la sucesión Fibonacci*

**...** 1 1 2 3 5 8 13 21 34 55 89

#### Comentarios multilínea

Python no dispone de un método para delimitar bloques de comentarios de varias líneas.

Al igual que los comentarios de un sola línea, son cadenas de caracteres, en este caso van entre triples comillas (simples o dobles), esto tiene el inconveniente que, aunque no genera código ejecutable, el bloque delimitado no es ignorado por el intérprete Python, que crea el correspondiente objeto de tipo *cadena de caracteres* (Página 45).

**>>>** *"""comentarios en varias lineas"""*

'comentarios en varias lineas'

**>>>** *'''comentarios en varias lineas'''*

'comentarios en varias lineas'

A continuación, una comparación entre comentarios multilínea y comentarios en solo una línea:

**>>>** *# Calcula la sucesión* **...** *# de números Fibonacci* **...**

**>>>** *"""Calcula la sucesión*

**...** *de números Fibonacci"""*

'Calcula la sucesi\xc3\xb3n \nde n\xc3\xbameros Fibonacci'

Entonces existen al menos dos (02) alternativas para introducir comentarios multilínea son:

* Comentar cada una de las líneas con el carácter #: en general todos los editores de programación y entornos de desarrollo (IDEs) disponen de mecanismos que permiten comentar y descomentar fácilmente un conjunto de líneas.
* Utilizar triple comillas (simples o dobles) para generar una cadena multilínea: si bien este método es aceptado.

A continuación, un ejemplo de Comentarios multilínea y de solo una linea:

**>>>** *u"""Calcula la sucesiónde números Fibonacci"""* u'Calcula la sucesi\xf3nde n\xfameros Fibonacci'

**>>>** *# se definen las variables*

**...** a, b = 0, 1

**>>> while** b < 100:

**... print** b,

**...** *# se calcula la sucesión Fibonacci*

**...** a, b = b, a + b

**...**

1 1 2 3 5 8 13 21 34 55 89

### Convertir a cadenas de caracteres

Para convertir a *tipos cadenas de caracteres* debe usar la función *str()* la cual *está integrada* en el intérprete Python.