CADENAS DE CARACTERES

Además de números, Python puede manipular cadenas de texto, las cuales pueden ser expresadas de distintas formas. Pueden estar encerradas en comillas simples (`'...'`) o dobles (`"..."`) con el mismo resultado. `\` puede ser usado para escapar comillas:

```
>>> 'huevos y pan' # comillas simples
'huevos y pan'
>>> 'doesn\'t' # usa \' para escapar comillas simples...
"doesn't"
>>> "doesn't" # ...o de lo contrario usa comillas dobles
"doesn't"
>>> '"Si," le dijo.'
'"Si," le dijo.'
>>> "\"Si,\" le dijo."
'"Si," le dijo.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

En el intérprete interactivo, la salida de caracteres está encerrada en comillas y los caracteres especiales se escapan con barras invertidas. Aunque esto a veces se vea diferente de la entrada (las comillas que encierran pueden cambiar), las dos cadenas son equivalentes. La cadena se encierra en comillas dobles si la cadena contiene una comilla simple y ninguna doble, de lo contrario es encerrada en comillas simples. La función `print()` produce una salida más legible, omitiendo las comillas que la encierran e imprimiendo caracteres especiales y escapados:

```
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
>>> print('"Isn\'t," she said.')
"Isn't," she said.
>>> s = 'Primerea línea.\nSegunda línea.' # \n significa nueva línea
>>> s # sin print(), \n es incluido en la salida
'Primera línea.\nSegunda línea.'
>>> print(s) # con print(), \n produce una nueva línea
Primera línea.
Segunda línea.
```

Si no quieres que los caracteres precedidos por `\` se interpreten como caracteres especiales, puedes usar *cadenas sin formato* agregando una `r` antes de la primera comilla:

```
>>> print('C:\algun\nombre') # aquí \n significa nueva línea!
C:\algun
ombre
>>> print(r'C:\algun\nombre') # nota la r antes de la comilla
```

```
C:\algun\nombre
```

Hay un aspecto sutil en las cadenas sin formato: una cadena sin formato no puede terminar en un número impar de \ caracteres; consulte la entrada de preguntas frecuentes (FAQ) para obtener más información y soluciones alternativas

Las cadenas de texto literales pueden contener múltiples líneas. Una forma es usar triples comillas: `"""..."""` o `'''...'''`. Los *fin de línea* son incluidos automáticamente, pero es posible prevenir esto agregando una \ al final de la línea. Por ejemplo:

```
print("""\
Uso: algo [OPTIONS]
    -h Muestra el mensaje de uso
    -H nombrehost Nombre del host al cual conectarse
""")
```

produce la siguiente salida (tener en cuenta que la línea inicial no está incluida):

```
Uso: algo [OPTIONS]
    -h Muestra el mensaje de uso
    -H nombrehost Nombre del host al cual conectarse
```

Las cadenas se pueden concatenar (pegar juntas) con el operador + y se pueden repetir con *:

```
>>> # 3 veces 'un', seguido de 'ium'
>>> 3 * 'un' + 'ium'
'ununununium'
```

Dos o más *cadenas literales* (es decir, las encerradas entre comillas) una al lado de la otra se concatenan automáticamente.

```
>>> 'Py' 'thon'
'Python'
```

Esta característica es particularmente útil cuando quieres dividir cadenas largas:

```
>>> texto = ('Poné muchas cadenas dentro de paréntesis '
'para que ellas sean unidas juntas.')
>>> texto
'Poné muchas cadenas dentro de paréntesis para que ellas sean unidas juntas.'
```

Esto solo funciona con dos literales, no con variables ni expresiones:

```
>>> prefix = 'Py'
>>> prefix 'thon'  # no se puede concatenar una variable y una cadena literal
  File "<stdin>", line 1
    prefix 'thon'
           ^^^^^^
SyntaxError: invalid syntax
>>> ('un' * 3) 'ium'
  File "<stdin>", line 1
```

```
    ('un' * 3) 'ium'
               ^^^^^
SyntaxError: invalid syntax
```

Si quieres concatenar variables o una variable y un literal, usa +:

```
>>> prefix + 'thon'
'Python'
```

Las cadenas de texto se pueden *indexar* (subíndices), el primer carácter de la cadena tiene el índice 0. No hay un tipo de dato diferente para los caracteres; un carácter es simplemente una cadena de longitud uno:

```
>>> palabra = 'Python'
>>> palabra[0] # caracter en la posición 0
'P'
>>> palabra[5] # caracter en la posición 5
'n'
```

Los índices también pueden ser números negativos, para empezar a contar desde la derecha:

```
>>> palabra[-1] # último caracter
'n'
>>> palabra[-2] # ante último caracter
'o'
>>> palabra[-6]
'P'
```

Nótese que -0 es lo mismo que 0, los índice negativos comienzan desde -1.

Además de los índices, las *rebanadas* también están soportadas. Mientras que los índices se utilizar para obtener caracteres individuales, las *rebanadas* te permiten obtener partes de las cadenas de texto:

```
>>> palabra[0:2] # caracteres desde la posición 0 (incluída) hasta la 2
(excluída)
'Py'
>>> palabra[2:5] # caracteres desde la posición 2 (incluída) hasta la 5
(excluída)
'tho'
```

Los índices de las rebanadas tienen valores por defecto útiles; el valor por defecto para el primer índice es cero, el valor por defecto para el segundo índice es la longitud de la cadena a rebanar.

```
>>> palabra[:2] # caracteres desde el principio hasta la posición 2
(excluída)
'Py'
11
>>> palabra[4:] # caracterrs desde la posición 4 (incluída) hasta el final
'on'
```

```
>>> palabra[-2:] # caracteres desde la ante-última (incluída) hasta el final
'on'
```

Nótese cómo el inicio siempre se incluye y el final siempre se excluye. Esto asegura que `s[:i] + s[i:]` siempre sea igual a `s`:

```
>>> palabra[:2] + palabra[2:]
'Python'
>>> palabra[:4] + palabra[4:]
'Python'
```

Una forma de recordar cómo funcionan las rebanadas es pensar que los índices apuntan *entre* caracteres, con el borde izquierdo del primer carácter numerado 0. Luego, el punto derecho del último carácter de una cadena de *n* caracteres tiene un índice *n*, por ejemplo

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

La primera fila de números da la posición de los índices 0…6 en la cadena; La segunda fila da los correspondientes índices negativos. La rebanada desde *i* hasta *j* consta de todos los caracteres entre los bordes etiquetados *i* y *j*, respectivamente.

Para índices no negativos, la longitud de la rebanada es la diferencia de los índices, si ambos están dentro de los límites. Por ejemplo, la longitud de `palabra[1:3]` es 2.

Intentar usar un índice que es muy grande resultará en un error:

```
>>> palabra[42] # la palabra solo tiene 6 caracteres
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Sin embargo, los índices de rebanadas fuera de rango se manejan satisfactoriamente cuando se usan para rebanar:

```
>>> palabra[4:42]
'on'
>>> palabra[42:]
''
```

Las cadenas de Python no se pueden modificar, son immutable. Por eso, asignar a una posición indexada de la cadena resulta en un error:

```
>>> palabra[0] = 'J'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> palabra[2:] = 'py'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Si necesitas una cadena diferente, deberías crear una nueva:

```
>>> 'J' + palabra[1:]
'Jython'
>>> palabra[:2] + 'py'
'Pypy'
```

La función incorporada `len()` retorna la longitud de una cadena:

```
>>> s = 'supercalifrastilisticoespialidoso'
>>> len(s)
33
```

Material readaptado para **ISPC** por el prof. Augusto Schaumburg, tema "*Strings o manejo de cadenas en Python*", tomando como base y guía "*El tutorial de Python*" de **Guido Van Rossum** (ref. bibliográficas debajo)

BIBLIOGRAFÍA

"El Tutorial de Python", Guido Van Rossum. Editor original: Fred L. Drake, Jr.

Se tomó como referencia la versión traducida (junio 2016) para python 3.3.0, realizada por voluntarios del grupo de usuarios de Python de Argentina. Se encuentran disponibles versiones actualizadas, pero no difieren en forma significativa de la utilizada como referencia para la creación de este material.

Los mismos se pueden encontrar en los siguientes enlaces:

Grupo Python Argentina: https://wiki.python.org.ar/aprendiendopython/

Material actulizado: http://tutorial.python.org.ar/

Python Org. (internacional) : https://docs.python.org/es/3/tutorial/index.html

Enlace del tema tratado (strings / cadenas) :
https://docs.python.org/es/3/tutorial/introduction.html#strings

LICENCIAS