Introducción a Cadena de caracteres

Versión ‘24

Cadena de Caracteres-Str

Las cadenas en Python o strings son un tipo inmutable que permite almacenar secuencias de caracteres.

Para crear una, es necesario incluir el texto entre comillas dobles “ o simples ‘.

Ej:
```
s = "Esto es una cadena"
print(s)                    #Esto es una cadena
print(type(s))              #<class 'str'>
```

Las cadenas no están limitadas en tamaño, por lo que el único límite es la memoria del procesador. Una cadena puede estar también vacía. s = “ ”

Cadena de Caracteres-Secuencias de escape

Cuando queremos introducir una comilla, sea simple ' o doble " dentro de una cadena debemos recurrir a las secuencias de escape.

-Incluir comillas dentro de una cadena:

\"

```
s = "Esto es una comilla doble \" de ejemplo"
print(s) #Esto es una comilla doble " de ejemplo
```

Cadena de Caracteres-Secuencias de escape

Cuando queremos introducir un salto de línea dentro de una cadena debemos recurrir a otra secuencia de escape.

-Incluir un salto de línea dentro de una cadena, lo que significa que lo que esté después del salto, se imprimirá en una nueva línea.

\n

```
s = "Primer linea\nSegunda linea"
print(s)
#Primer linea
#Segunda linea
```

Cadena de Caracteres-Asignar a variables

Puedes asignar una cadena de caracteres a una variable

```
x = "Este texto de ejemplo es muy largo, resultando en un código significativamente más largo."
        print ( x )
#Este texto de ejemplo es muy largo, resultando en un código significativamente más largo.
```

Cadena de Caracteres-Asignar a variables

Para generar y definir una cadena con varias líneas puedes utilizar tres comillas simples o dobles.

Los saltos de línea también se tienen en cuenta a la hora de emitir la cadena.

```
x = """Este texto de ejemplo es muy largo, resultando en un código significativamente más largo.
Continúa en una nueva línea,
se amplía con una tercera y cuarta línea
y finalmente termina con un punto y aparte."""
        print ( x )
```

Cadena de Caracteres-Asignar a variables

El resultado correspondiente es el siguiente:

Este texto de ejemplo es muy largo, resultando en un código significativamente más largo.

Continúa en una nueva línea,

se amplía con una tercera y cuarta línea

y finalmente termina con un punto y aparte.

Cadena de Caracteres-Formateo de cadenas

Para declarar una cadena que contenga variables en su interior, como números o incluso otras cadenas se concatenan usando el operador +.

Nótese que str() convierte (castea) en string lo que se pasa como parámetro.

```
x = 5
s = "El número es: " + str(x)
print(s) #El número es: 5
```

Cadena de Caracteres-Formateo de cadenas

Otra forma es usando %.

Por un lado tenemos %s que indica el tipo que se quiere imprimir, y por otro a la derecha del % tenemos la variable a imprimir.

Para imprimir una cadena se usaría %s o %f para un valor en coma flotante.

```
x = 5
s = "El número es: %d" %x
print(s) #El número es: 5
```

Cadena de Caracteres-Formateo de cadenas

Si tenemos más de una variable, también se puede hacer pasando los parámetros dentro de ().

Muy parecido a lenguajes como C.

No obstante, esta no es la forma preferida de hacerlo, ahora que tenemos nuevas versiones de Python.

```
s = "Los números son %d y %d." % (5, 10)
print(s) #Los números son 5 y 10.
```

Cadena de Caracteres-Formateo de cadenas

Una forma un poco más moderna de realizar lo mismo, es haciendo uso de format().

```
s = "Los números son {} y {}".format(5, 10)
print(s) #Los números son {} y {}".format(5, 10)
```

Es posible también darle nombre a cada elemento, y format() se encargará de reemplazar todo.

```
s = "Los números son {a} y {b}".format(a=5, b=10)
print(s) #Los números son 5 y 10
```

Cadena de Caracteres-Formateo de cadenas

Por si no fueran pocas ya, existe una tercera forma de hacerlo introducida en la versión 3.6 de Python.

Reciben el nombre de cadenas literales o f-strings.

Esta nueva característica, permite incrustar expresiones dentro de cadenas.

```
a = 5; b = 10
s = f"Los números son {a} y {b}"
print(s) #Los números son 5 y 10
```

Cadena de Caracteres-Formateo de cadenas

Se puede hacer operaciones dentro de la creación del string o incluso llamar a una función.

```
a = 5; b = 10
s = f"a + b = {a+b}"
print(s) #a + b = 15
```

```
def funcion():
    return 20
s = f"El resultado de la función es {funcion()}"
print(s) #El resultado de la funcion es 20
```

Cadena de Caracteres-Formateo de cadenas

Se puede multiplicar un string por un int. Su resultado es replicarlo tantas veces como el valor del entero.

```
s = "Hola "
print(s*3)          #Hola Hola Hola
```

Cadena de Caracteres-Formateo de cadenas

Podemos ver si una cadena esta contenida en otra con in.

```
print("cielo" in "Rascacielos")      #True
```

O asegurarse de que un término **no** aparece en la cadena

```
text = "Este texto de ejemplo es muy largo y resulta en un código significativamente más largo."
if "corto" not in text:
    print(" No, 'corto' NO aparece en este extracto.")
```

Cadena de Caracteres-Formateo de cadenas

Se puede convertir a string otras clases, como int o float.

```
x = str(10.4)
print(x)          #10.4
print(type(x))    #<class 'str'>
```

Cadena de Caracteres-Formateo de cadenas

Se pueden indexar las cadenas, como si fuera una lista.

```
x = "abcde"
print(x[0])  #a
print(x[-1]) #e
```

- hace que vuelva hacia atrás por los índices de los elementos

Cadena de Caracteres-Formateo de cadenas

Del mismo modo, se pueden crear cadenas más pequeñas partiendo de una grande, indicando la posición del primer elemento y el la posición de último que queremos tomar menos uno.

```
x = "abcde"
print(x[0:2])  #ab
```

Si no se indica ningún valor a la derecha de los : se llega hasta el final.

```
x = "abcde"
print(x[2:])    #cde
```

Cadena de Caracteres-Formateo de cadenas

Es posible crear subcadenas que contengan elementos salteados y no contiguos añadiendo un tercer elemento entre []. Indica los elementos que se saltan. En el siguiente ejemplo se toman elementos del 0 al 5 de dos en dos.

```
x = "abcde"
print(x[0:5:2])     #ace
```

Tampoco es necesario saber el tamaño de la cadena, y el segundo valor se podría omitir. El siguiente ejemplo es igual al anterior.

```
x = "abcde"
print(x[0::2])      #ace
```

Cadena de Caracteres-Formateo de cadenas

Se puede leer una cadena con un bucle for.

Una cadena se utiliza como un array/lista en Python, por lo que, con el bucle apropiado, se puede recorrer cualquier palabra.

```
for caracter in "python":
    print(caracter)
```

```
p
y
t
h
o
n
```

Cadena de Caracteres-Formateo de cadenas

La función “len” permite determinar la longitud de una cadena.

Es especialmente útil para extractos de código muy largos.

```
text = "Este texto de ejemplo es muy largo, resultando en un código significativamente más largo."
print(len(text))
        # 89
```

Cadena de Caracteres-Formateo de cadenas

Los caracteres del string se ennumeran comenzando en 0, es decir, el primer carácter del string está en la posición 0, el segundo en la posición 1, y así sucesivamente. Por ejemplo, si s = “abracadabra”,

s[0] será “a”

s[1] será “b”

s[2] será “r”

s[3] será “a”

s[4] será “c”

… y así sucesivamente.

Cadena de Caracteres-Formateo de cadenas

¿En que posición está el último carácter de un string de largo N?

Dado que los caracteres se enumeran desde 0, el último carácter estará en la posición N-1.

Por ejemplo, si s=”hola”, len(s) es 4, y:

s[0] es “h”

s[1] es “o”

s[2] es “l”

s[3] es “a” (si N = 4, N-1 es 3)

s[4] lanzaría un error, dado que no hay carácter en la posición 4.

Cadena de Caracteres-Modificar o eliminar cadenas

Una cadena no puede ser modificado a posteriori.

Tampoco se pueden borrar caracteres individuales de la cadena.

El programa bloquea cualquier intento de modificación y muestra un mensaje de error.

Por lo tanto, la única opción de eliminar una cadena defectuosa del código es borrándolos por completo.

Se borran mediante el comando del

```
text = "Este es un texto de ejemplo"
del text
```