La librería String

La librería *string* de la biblioteca estándar de Python define constantes y operaciones comunes de las cadenas de caracteres. Esta librería define constantes y operaciones útiles para trabajar con el tipo de dato *string*.

Constantes

A continuación, se listan las constantes que define la librería *string*:

Nombre	Descripción	Valor
ascii_letters	La concatenación de las letras minúsculas y letras mayúsculas.	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
ascii_lowercase	Las letras minúsculas.	abcdefghijklmnopqrstuvwxyz
ascii_uppercase	Las letras mayúsculas.	ABCDEFGHIJKLMNOPQRSTUVWXYZ
digits	Los dígitos del sistema decimal.	0123456789
hexdigits	Los dígitos del sistema hexadecimal.	0123456789abcdefABCDEF
octdigits	Los dígitos del sistema octal.	01234567
punctuation	Símbolos de puntuación.	!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~
printable	Todos los caracteres considerados imprimibles.	0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL

		MNOPQRSTUVWXYZ!"#$%& \'()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c
whitespace	Todos los caracteres considerados espacios en blanco.	\t\n\r\x0b\x0c

Formateo de strings

El tipo de dato *string* provee la función *format* para realizar una sustitución compleja de variables y formateo de valores. La librería *string* define la clase *Formatter* que te permite crear y personalizar el comportamiento del formateo de strings usando la misma implementación que la función *format* de los strings.

En esta sección veremos cómo formatear strings con la función *format* de los strings, que es el comportamiento default del método *format* de la clase *Formatter*.

Tener en cuenta que, si se quiere cambiar la forma de formatear los strings en nuestro programa, conviene utilizar la clase *Formatter* para formatear los strings, en caso contrario se puede utilizar directamente el método format de los strings.

En la documentación de Python se puede ver más información sobre la sintaxis[1] y la especificación[2] del formateo de los strings.

A continuación, veremos los usos más relevantes de la función format mostrando varios ejemplos.

```
>>> name = 'Agustín'

# Acceso a los argumentos por posición

>>> 'Hola {}'.format(name)
'Hola Agustín'

>>> '{} + {} = {}'.format(2, 5, 7)
```

[1] https://docs.python.org/3.1/library/string.html#format-string-syntax
[2] https://docs.python.org/3.1/library/string.html#format-specification-mini-language

```
‘2 + 5 = 7’


>>> '{1} + {2} = {0}'.format(7, 5, 2)
‘5 + 2 = 7’

# Acceso a los argumentos por nombre

>>> 'Hola {name}'.format(name=name)
‘Hola Agustín’

>>> '{0} + {1} = {result}'.format(2, 5, result=7)
‘2 + 5 = 7’



# Acceso a los elementos del argumento

>>> tupla = (4, 3)
>>> 'X: {0[0]};  Y: {0[1]}'.format(tupla)
‘X: 4; Y: 3’

# Aplicando formato a los elementos

>>> '{0:f} + {1:f} = {result:f}'.format(2, 5, result=7)
'2.000000 + 5.000000 = 7.000000'

>>> '{0:.3f} + {1:.3f} = {result:.3f}'.format(2, 5, result=7)
'2.000 + 5.000 = 7.000'

>>> '{:d}'.format(25)
‘25’
```

```
>>> '{:d}'.format(25.5)
ValueError: Unknown format code 'd' for object of type 'float'

>>> '{:.0f}'.format(25.50)
‘26’

# Alineación del texto reemplazado

>>> 'Hola {name:16}'.format(name=name)
'Hola Agustín        '

>>> 'Hola {name:<16}'.format(name=name)
'Hola Agustín        '

>>> 'Hola {name:>16}'.format(name=name)
'Hola         Agustín'

>>> 'Hola {name:^16}'.format(name=name)
'Hola {name:^16}'.format(name=name)

>>> 'Hola {name:*^16s}'.format(name=name)
'Hola {name:*^16s}'.format(name=name)
```