Las strings en Python pueden ser definidas usando indistintamente las *single quotation marks* o las *double quotation marks*. Por ejemplo:

*String = “Hello” es equivalente a String = ‘Hello’*

Se pueden crear strings de varias líneas usando tres *single quotation marks*  o tres *double quotation marks*:

*String = “””*Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""

Los strings son considerados arrays de bytes representando caracteres Unicode. Un carácter se considera un string de tamaño 1.

Para acceder a un elemento de una string, se pueden usar corchetes. Cogiendo como ejemplo el string anterior:

String = '''Lorem ipsum dolor sit amet, consectetur adipiscing elit,sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua.'''

print(String[0]) => L

De igual manera, puedes seleccionar un rango de caracteres de la siguiente manera:

Nota: Cuando seleccionas un rango [start:end], la posición end no está incluida.

String = "Hello"

print(String[0:2]) => He

Los strings también se pueden indexar desde el final con números negativos:

Nota: Cuando seleccionas un rango [start:end], la posición end no está incluida. Para poder coger el último elemento, debes dejar end vacío.

String = "Hello"

print(String[-3:-1]) => ll

print(String[-3:]) => llo

Otras funcionalidades de Python con las strings serían:

Para ver todos los String Methods: <https://www.w3schools.com/python/python_strings.asp>

Para ver si una frase está contenida en un string, se puede usar el operador *in*:

txt = "The rain in Spain stays mainly in the plain"

x = "ain" in txt

print(x) = True

Para ver si una frase no se encuentra en un string, se puede usar el operador *not in*:

txt = "The rain in Spain stays mainly in the plain"

x = "ain" not in txt

print(x) = False

Para concatenar o combinar dos strings, se puede usar el operador +:

a = "Hello"

b = "World"

c = a + b

print(c) => HelloWorld

Como se puede ver, los strings se juntan sin dejar huecos. Para evitarlo, suma un “ “:

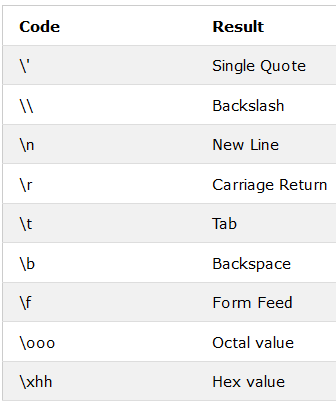
a = "Hello"

b = "World"

c = a + " " + b

print(c) => Hello World

Hay caracteres especiales que usados detrás de un \ cumplen funciones concretas:



Por ejemplo, \x48 = H

Por ejemplo, \110 = H