Concatenación de Cadenas

Cuando el operador + es aplicado a una cadena, significa “concatenación”

```
>>> a = 'Hola'
>>> b = a + 'Ahí'
>>> print(b)
HolaAhí
>>> c = a + ' ' + 'Ahí'
>>> print(c)
Hola Ahí
>>>
```

Utilizando in como Operador Lógico

- La palabra in puede ser utilizada para revisar si una cadena se encuentra “en (in)” otra cadena
- La expresión in es una expresión lógica que retorna True o False y puede ser utilizada una sentencia if

```
>>> fruta = 'banana'
>>> 'n' in fruta
True
>>> 'm' in fruta
False
>>> 'nan' in fruta
True
>>> if 'a' in fruta :
...     print('Encontrada!')
...
Encontrada!
>>>
```

Comparación de Cadenas

```
if palabra == 'banana':
    print('Muy bien, bananas.')

if palabra < 'banana':
    print('Tu palabra,' + palabra + ', está antes de banana.')
elif palabra > 'banana':
    print('Tu palabra,' + palabra + ', está después de banana.')
else:
    print('Muy bien, bananas.')
```

Librería String

- Python tiene un número de funciones de cadenas que están en la librería string (cadena)
- Esas funciones ya están previamente construidas dentro de cada cadena – las invocamos al agregar la función a la variable de la cadena
- Esas funciones no modifican la cadena original, sino que retornan una nueva cadena que ha sido modificada

```
>>> saludo = 'Hola Bob'
>>> zap = saludo.lower()
>>> print(zap)
hola bob
>>> print(saludo)
Hola Bob
>>> print('Hola Ahí'.lower())
hola ahí
>>>
```

```
>>> cosa = 'Hola mundo'
>>> type(cosa)
<class 'str'>
>>> dir(cosa)
['capitalize', 'casefold', 'center', 'count', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'format_map',
'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper',
'zfill']
```

https://docs.python.org/3/library/stdtypes.html#string-methods

`str.`**`replace`**(*old*, *new*[, *count*])

Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

`str.`**`rfind`**(*sub*[, *start*[, *end*]])

Return the highest index in the string where substring *sub* is found, such that *sub* is contained within `s[start:end]`. Optional arguments *start* and *end* are interpreted as in slice notation. Return `-1` on failure.

`str.`**`rindex`**(*sub*[, *start*[, *end*]])

Like `rfind()` but raises `ValueError` when the substring *sub* is not found.

`str.`**`rjust`**(*width*[, *fillchar*])

Return the string right justified in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to `len(s)`.

`str.`**`rpartition`**(*sep*)

Split the string at the last occurrence of *sep*, and return a 3-tuple containing the part before the separator, the separator itself, and the part after the separator. If the separator is not found, return a 3-tuple containing two empty strings, followed by the string itself.

`str.`**`rsplit`**(*sep=None*, *maxsplit=-1*)

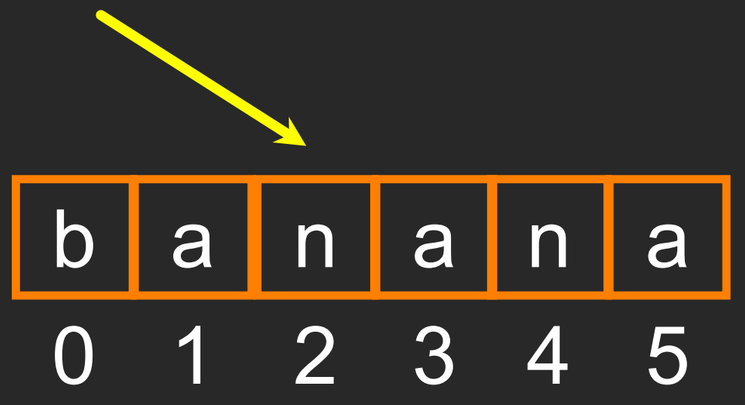
Return a list of the words in the string, using *sep* as the delimiter string. If *maxsplit* is given, at most *maxsplit* splits are done, the *rightmost* ones. If *sep* is not specified or `None`, any whitespace string is a separator. Except for splitting from the right, `rsplit()` behaves like `split()` which is described in detail below.

Librería String

```
str.capitalize()
str.center(width[, fillchar])
str.endswith(suffix[, start[, end]])
str.find(sub[, start[, end]])
str.lstrip([chars])
str.replace(old, new[, count])
str.lower()
str.rstrip([chars])
str.strip([chars])
str.upper()
```

Buscando una Cadena

- Utilizamos la función find() para buscar una subcadena dentro de otra cadena
- find() encuentra la primer ocurrencia de la subcadena
- Si la subcadena no se encuentra, find() regresa -1
- Recuerda que las posiciones de una cadena comienzan en cero.




```
>>> fruta = 'banana'
>>> pos = fruta.find('na')
>>> print(pos)
2
>>> aa = fruta.find('z')
>>> print(aa)
-1
```

Convirtiéndo Todo a MAYÚSCULAS

- Puedes crear una copia de una cadena en minúsculas o mayúsculas
- Frecuentemente cuando estamos buscando una cadena utilizando find() primero convertimos la cadena a minúsculas, de modo que podemos buscar una cadena sin importar si está en mayúsculas o minúsculas

```
>>> saludo = 'Hola Bob'
>>> nnn = saludo.upper()
>>> print(nnn)
HOLA BOB
>>> www = saludo.lower()
>>> print(www)
hola bob
>>>
```

Buscar y Reemplazar

- La función replace() es como una operación "buscar y reemplazar" en un editor de texto
- Esta función reemplaza todas las ocurrencias de una cadena de búsqueda con una cadena de reemplazo

```
>>> saludo = 'Hola Bob'
>>> ncad = saludo.replace('Bob','Jane')
>>> print(ncad)
Hola Jane
>>> ncad = saludo.replace('o','X')
>>> print(ncad)
HXla BXb
>>>
```

Removiendo Espacios en Blanco

- A veces queremos tomar una cadena y remover los espacios en blanco al inicio y/o al final
- lstrip() y rstrip() remueven los espacios en blanco a la izquierda o a la derecha
- strip() remueve espacios en blanco tanto al inicio como al final de la cadena

```
>>> saludo = '   Hola Bob  '
>>> saludo.lstrip()
'Hola Bob  '
>>> saludo.rstrip()
'   Hola Bob'
>>> saludo.strip()
'Hola Bob'
>>>
```

Prefijos

```
>>> linea = 'Que tengas un buen día'
>>> linea.startswith('Que')
True
>>> linea.startswith('q')
False
```

Análisis y Extracción

21 31

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> datos = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> arrpos = datos.find('@')
>>> print(arrpos)
21
>>> esppos = datos.find(' ', arrpos)
>>> print(esppos)
31
>>> direccion = datos[arrpos+1 : esppos]
>>> print(direccion)
uct.ac.za
```

Cadenas y Conjuntos de Caracteres

```
Python 2.7.10
>>> x = '이광춘'
>>> type(x)
<type 'str'>
>>> x = u'이광춘'
>>> type(x)
<type 'unicode'>
>>>
```

```
Python 3.5.1
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> x = u'이광춘'
>>> type(x)
<class 'str'>
>>>
```

En Python 3, todas las cadenas son Unicode

Resumen

- Tipo Cadena (String)
- Leer/Convertir
- Indexando cadenas []
- Rebanando cadenas [2:4]
- Atravesando cadenas con for y while
- Concatenando cadenas con +
- Operaciones de Cadenas
- Librería String
- Comparación de Cadenas
- Búsqueda de Cadenas
- Reemplazando texto
- Removiendo espacios

Agradecimientos / Contribuciones

Desarrollo inicial: Charles Severance, Escuela de Informática de la Universidad de Michigan.

Traducción al Español por Juan Carlos Pérez Castellanos - 2020-04-10