

strings: métodos

métodos de análisis

count() : retorna el número de veces que se repite un conjunto de caracteres especificado.

```
"Hola mundo".count("Hola")  
>> 1
```

find() e **index()** retornan la ubicación (comenzando desde el cero) en la que se encuentra el argumento indicado. Difieren en que **index** lanza **ValueError** cuando el argumento no es encontrado, mientras **find** retorna **-1**.

```
"Hola mundo".find("world")  
>> -1
```

rfind() y **rindex()**. Para buscar un conjunto de caracteres pero desde el final.

```
"C:/python36/python.exe".rfind("/")  
>> 11
```

startswith() y **endswith()** indican si la cadena en cuestión comienza o termina con el conjunto de caracteres pasados como argumento, y retornan **True** o **False** en función de ello.

```
"Hola mundo".startswith("Hola")  
>> True
```

strings: métodos

métodos de análisis

isdigit(): determina si todos los caracteres de la cadena son **dígitos**, o **pueden formar números**, incluidos aquellos correspondientes a lenguas orientales.

```
"abc123".isdigit()  
>> False
```

isnumeric(): determina si todos los caracteres de la cadena son **números**, incluye **también caracteres de connotación numérica** que no necesariamente son dígitos (por ejemplo, una fracción).

```
"1234".isnumeric()  
>> True
```

isdecimal(): determina si todos los caracteres de la cadena son **decimales**; esto es, formados por dígitos **del 0 al 9**.

```
"1234".isdecimal()  
>> True
```

isalnum(): determina si todos los caracteres de la cadena son **alfanuméricos**.

```
"abc123".isalnum()  
>> True
```

isalpha(): determina si todos los caracteres de la cadena son **alfabéticos**.

```
"abc123".isalpha()  
>> False
```

strings: métodos

métodos de análisis

islower(): determina si todos los caracteres de la cadena son **minúsculas**.

```
"abcdef".islower()  
>> True
```

isupper(): determina si todos los caracteres de la cadena son **mayúsculas**.

```
"ABCDEF".isupper()  
>> True
```

isprintable(): determina si todos los caracteres de la cadena son **imprimibles** (es decir, no son caracteres especiales indicados por \...).

```
"Hola \t mundo!".isprintable()  
>> False
```

isspace(): determina si todos los caracteres de la cadena son **espacios**.

```
"Hola mundo".isspace()  
>> False
```

strings: métodos

métodos de transformación

En realidad los strings son inmutables; por ende, todos los métodos a continuación no actúan sobre el objeto original sino que retornan uno nuevo.

capitalize() retorna la cadena con su primera letra en mayúscula.

```
"hola mundo".capitalize()  
>> 'Hola mundo'
```

encode() codifica la cadena con el mapa de caracteres especificado y retorna una instancia del tipo bytes.

```
"Hola mundo".encode("utf-8")  
>> b'Hola mundo'
```

replace() reemplaza una cadena por otra.

```
"Hola mundo".replace("mundo", "world")  
>> 'Hola world'
```

lower() retorna una copia de la cadena con todas sus letras en minúsculas.

```
"Hola Mundo!".lower()  
>> 'hola mundo!'
```

upper() retorna una copia de la cadena con todas sus letras en mayúsculas.

```
"Hola Mundo!".upper()  
>> 'HOLA MUNDO!'
```


strings: métodos

métodos de transformación

swapcase() cambia las mayúsculas por minúsculas y viceversa.

```
"Hola Mundo!".swapcase()  
>> 'hOLA mUNDO!'
```

strip(), **lstrip()** y **rstrip()** remueven los espacios en blanco que preceden y/o suceden a la cadena.

```
"  Hola mundo!  ".strip()  
>> 'Hola mundo!'
```

Los métodos **center()**, **ljust()** y **rjust()** alinean una cadena en el centro, la izquierda o la derecha. Un segundo argumento indica con qué carácter se deben llenar los espacios vacíos (por defecto un espacio en blanco).

```
"Hola".center(10, "*")  
>> '***Hola***'
```

métodos de separación y unión

split() divide una cadena según un carácter separador (por defecto son espacios en blanco). Un segundo argumento en **split()** indica cuál es el máximo de divisiones que puede tener lugar (-1 por defecto para representar una cantidad ilimitada).

```
"Hola mundo!\nHello world!".split()  
>> ['Hola', 'mundo!', 'Hello', 'world!']
```

strings: métodos

métodos de separación y unión

splitlines() divide una cadena con cada aparición de un salto de línea.

```
"Hola mundo!\nHello world!".splitlines()  
>> ['Hola mundo!', 'Hello world!']
```

partition() retorna una tupla de tres elementos: el bloque de caracteres anterior a la primera ocurrencia del separador, el separador mismo, y el bloque posterior.

```
"Hola mundo. Hello world!".partition(" ")  
>> ('Hola', ' ', 'mundo. Hello world!')
```

rpartition() opera del mismo modo que el anterior, pero partiendo de derecha a izquierda.

```
"Hola mundo. Hello world!".rpartition(" ")  
>> ('Hola mundo. Hello', ' ', 'world!')
```

join() debe ser llamado desde una cadena que actúa como separador para unir dentro de una misma cadena resultante los elementos de una lista.

```
", ".join(["C", "C++", "Python", "Java"])  
>> 'C, C++, Python, Java'
```