

String en Python (parte 2)

Rodrigo Toro Icarte

Funciones sobre strings

`s[i:j:k]` (✓✓✓)

Para obtener un trozo de un string se utiliza `s[i:j]`, que retorna el substring de `s` entre `i` y `j`. Notar que si no damos un valor a `i`, comienza desde el inicio, y si no damos valor a `j`, se considera hasta el final.

y	o		s	o	y		t	u	
0	1	2	3	4	5	6	7	8	9

```
1 a = "yo soy tu padre"
2 print(a[3:6])           # >>> soy
3 print(a[:6])           # >>> yo soy
4 print(a[3:])           # >>> soy tu padre
```

Adicionalmente, se puede definir un valor `k` que indique de cuánto en cuánto avanzamos por el string.

```
1 s = "que la curiosidad me mate"
2
3 # mostrar caracteres pares
4 print(s[::2])
5 # mostrar caracteres impares
6 print(s[1::2])
7 # invertir el string
8 print(s[::-1])
```

`s.find(c)` (✓✓✓) y `s.rfind(c)` (✓)

Ambos métodos retornan el índice de la primera ocurrencia de `c` en `s` (-1 si no existe). `find()` busca de izquierda a derecha y `rfind()` de derecha a izquierda.

```
1 s = "el mejor placer de la vida es hacer las cosas que la gente
    dice que no podemos hacer"
2
3 print(s.find("d"))           # >>> 16
4 print(s.rfind("d"))         # >>> 73
5 print(s.find("vida"))       # >>> 22
6 print(s.find("paz"))        # >>> -1
7 print(s[s.find("d"):s.rfind("d")]) # ???
```

`s.lower()` (✓✓✓), `s.upper()` (-) y `s.capitalize()` (X)

Estos métodos permiten pasar un string a mayúscula (`upper()`) y minúscula (`lower()`). Además `capitalize()` deja todo en minúscula salvo la primera letra.

```
1 s = "La perseverancia puede cambiar un fracaso en un extraordinario
    logro"
2
3 print(s.lower())           # >>> la perseverancia puede...
4 print(s.upper())          # >>> LA PERSEVERANCIA PUEDE...
5 print(s.capitalize())     # >>> La perseverancia puede...
```

`s.strip(c)` (✓), `s.lstrip(c)` (X), `s.rstrip(c)` (X)

Estos métodos permiten eliminar caracteres extras a los extremos del string (por ejemplo, eliminar los espacios sobrantes a los costados). `s.strip(c)` elimina caracteres `c` a ambos extremos, mientras que `s.lstrip(c)` y `s.rstrip(c)` eliminan sólo a la izquierda y a la derecha, respectivamente.

```
1 s = "\n\t haz de tu vida un sueño, y de tu sueño una realidad.
    :):):)"
2
3 # quita los ':):)' de la derecha
4 print(s.rstrip(':):)'))
5 # quita espacios, \n y \t de la izquierda
6 print(s.lstrip())
```

`s.startswith(c)` (✓✓) y `s.endswith(c)` (✓✓✓)

Estos métodos retornan `True` si el string `s` comienza con `c` (`s.startswith(c)`), o termina con `c` (`s.endswith(c)`).

```
1 s = "si estás corriendo, no importa qué tan rápido o lento eres, tú
   eres un corredor"
2
3 print(s.startswith('si estás c')) # >>> True
4 print(s.endswith('juego'))       # >>> False
```

`s.isalpha()` (✓) y `s.isdigit()` (✓✓✓)

Estos métodos retornan `True` si `s` está compuesto sólo de caracteres alfabéticos (`s.isalpha()`) o numéricos (`s.isdigit()`). Notar que `isdigit()` sirve para chequear si un string se puede convertir a `int` o `float`.

```
1 s = "no he fracasado. Sólo acabo de encontrar 10.000 maneras de no
   lograrlo."
2
3 print(s.isalpha())                # >>> False
4 print(s[6:15].isalpha())          # >>> True  (fracasado)
5 print(s[41:47].isdigit())         # >>> False (10.000)
6 print(s[41:43].isdigit())         # >>> True  (10)
```

`s.count(c)` (✓)

Este método retorna el número de ocurrencias de `c` en `s`.

```
1 s = "debes ser el cambio que quieras ver en el mundo."
2
3 print(s.count('e'))               # >>> 9
4 print(s.count('el'))              # >>> 2
5 print(s.count('mundo'))           # >>> 1
6 print(s.count('que, '))           # >>> 0
```

`s.replace(c1,c2)` (✓✓✓)

Este método retorna un string igual a `s`, pero reemplazando cada ocurrencia de `c1` por `c2`. Generalmente se usa para quitar espacios y signos de puntuación.

```
1 s = "el éxito llega cuando tus sueños superan tus excusas."
2
3 # cambio una frase por otra
4 print(s.replace("tus sueños superan", "superas"))
5
6 # quito los espacios
7 print(s.replace(" ", ""))
8
9 # quito signos de puntuación
10 s = s.replace(".", "").replace(",","").replace(":", "")
11 s = s.replace(";","").replace("!", "").replace("?", "")
12 print(s)
```

`s.split(c)` (✓✓✓) y `c.join(l)` (✓✓)

Estos métodos permiten separar un string en una *lista* de componentes (`split()`), y unir una *lista* de elementos en un string (`join()`). Si bien los strings son *listas de caracteres*, también podemos crear listas genéricas de números o de strings. En el caso de `split()` y `join()`, trabajan con listas de strings. Para obtener los elementos de una lista genérica se usa `s[id]`, al igual que para los strings.

`s.split(c)` divide `s` según las ocurrencias de `c` y retorna sus partes en una *lista*. Mientras que `c.join(l)` une una *lista l* mediante `c`, y retorna el string formado.

```
1 s = "debes permanecer enfocado en tu camino hacia la grandeza."
2
3 l = s.split(" ") # separo s en los " "
4 print(l)
5 # >>> ['debes', 'permanecer', 'enfocado', 'en', ...]
6 print(l[3]) # >>> en
7 print(l[-5]) # >>> tu
8 s2 = "\n".join(l) # unimos l mediante "\n"
9 print(s2) # ???
```

Ejercicios propuestos

1. Cree una función que reciba un string y lo retorne en minúscula y sin signos de puntuación (.,;-/!¿?) ni caracteres espaciales (`\n`, `\t`, `\r`, `\'`, `\'`, `\\`).
2. Cree un reconocedor de palíndromos que no considere ni espacios ni signos de puntuación.
3. Implemente la función **mayor_que(s,k)** tal que retorne True ssi **s** tiene una palabra de tamaño mayor o igual a **k**. Notar que el método creado en **1)** le será útil.
4. Cree una función que reciba un string **s** y retorne True ssi **s** es un correo electrónico.
5. Cree una función que reciba un string **s** y retorne True ssi **s** puede ser convertido a float.
6. Cree una función que reciba un string **s** y retorne la cantidad de números enteros que en él aparecen.
7. Cree una función que reciba un string **s** y retorne su traducción a jerigonzo (ej: rodrigo \rightarrow ropodripigopo).
8. Cree una función que reciba **s** y retorne True ssi **s** es un pangram (contiene todas las letras del alfabeto sin considerar eñes ni tildes).
9. Cree el codificador ROT-*n*. Este codificador mapea cada letra del string a la letra que está *n* posiciones delante de ella en el alfabeto (sin considerar la eñe). Ej: En ROT-13: a \rightarrow n; b \rightarrow o, ..., m \rightarrow z, n \rightarrow a, ..., z \rightarrow m. Para esto, use las propiedades del formato ASCII y las funciones **ord()** y **chr()**.