

Clase 4: Strings

Prof. Nicolás Torres

nicolas.torresr@usm.cl

Ingeniería Civil Telemática

Departamento de Electrónica

Universidad Técnica Federico Santa María

Strings

Cadenas de texto que se definen entre comillas simples `' '` o dobles `" "`.

```
In [1]: mensaje = 'Hola Mundo'
saludo = "Hello World"
```

Caracteres especiales

Python posee caracteres especiales que interpreta dentro de los strings. El carácter de barra invertida o *backslash* (`\`), se usa como carácter de escape para producir efectos especiales en los textos como agregar una nueva línea o una tabulación.

```
In [2]: x = 'blanco\nazul\nrojo'
print(x)
```

```
blanco
azul
rojo
```

```
In [3]: print("blanco\tazul\trojo\nverde\tlila\tgris")
```

```
blanco azul rojo
verde lila gris
```

Indexación

Una cadena de caracteres tiene elementos ordenados posicionalmente. Por lo tanto, cada carácter dentro del texto puede ser enumerado individualmente.

| | | | | | | | | | | |
|-----|-----|----|----|----|----|----|----|----|----|----|
| -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |
| H | e | l | l | o | | W | o | r | l | d |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Acceso

Es posible acceder a cualquier caracter de la secuencia a través de su índice usando el operador corchete `[]`.

```
In [4]: vocales = "aeiou"
```

```
In [5]: vocales[0]
```

```
Out[5]: 'a'
```

```
In [6]: vocales[-1]
```

```
Out[6]: 'u'
```

Inmutabilidad

Los strings son secuencias **inmutables** de caracteres, es decir, sus elementos no pueden ser modificados.

```
In [7]: vocales[0] = "f"
```

```
-----
TypeError                                Traceback (most recent call last)
Input In [7], in <cell line: 1>()
----> 1 vocales[0] = "f"

TypeError: 'str' object does not support item assignment
```

Rebanada (Slicing)

`s[i:j]`, entrega los caracteres comenzando en la posición `i` hasta el carácter en la posición `j-1` del string `s`.

```
In [8]: vocales[1:4]
```

```
Out[8]: 'eio'
```

`s[i:]` entrega los elementos desde el índice `i` hasta el final.

```
In [9]: vocales[2:]
```

```
Out[9]: 'iou'
```

`s[:j]` entrega los elementos desde el inicio hasta el índice `j-1`.

```
In [10]: vocales[:3]
```

```
Out[10]: 'aei'
```

`s[i:j:k]`, entrega los caracteres comenzando en la posición `i` hasta el carácter en la posición `j-1` con salto o incremento de `k`.

```
In [11]: vocales[1:4:2]
```

```
Out[11]: 'eo'
```

Operaciones elementales sobre strings

Concatenación

El operador concatenador (`+`) retorna la unión entre secuencias.

```
In [12]: 'a' + 'b' + 'c'
```

```
Out[12]: 'abc'
```

```
In [13]: '1' + '2'
```

```
Out[13]: '12'
```

Repetición

El operador repetidor (`*`) retorna los elementos de la secuencia repetidos.

```
In [14]: 'na ' * 5
```

```
Out[14]: 'na na na na na '
```

Membresía

Los operadores de **membresía** o **pertenencia** (`in` y `not in`) evalúan si un valor es miembro de otro. En otras palabras, permiten saber si un elemento está contenido en una secuencia. En los strings, `x in y` es `True` si y solo si `x` es un substring de `y` .

```
In [15]: 'pollo' in 'repollo'
```

```
Out[15]: True
```

```
In [16]: 'pollo' in 'rechicken'
```

```
Out[16]: False
```

Funciones básicas sobre strings

Longitud de un string

La función `len(s)` retorna el número de caracteres en el string `s` .

```
In [17]: len("hola mundo")
```

```
Out[17]: 10
```

```
In [18]: len('')
```

```
Out[18]: 0
```

```
In [19]: len('a\nb')
```

```
Out[19]: 3
```

Mínimo y Máximo

La función `min(s)` retorna el menor carácter en el string `s`.

La función `max(s)` retorna el mayor carácter en el string `s`.

El criterio que utiliza para comparar caracteres es el **orden lexicográfico**.

```
In [20]: min('aeiou')
```

```
Out[20]: 'a'
```

```
In [21]: max('aeiou')
```

```
Out[21]: 'u'
```

Métodos

- Un método es una función que «pertenece a» un tipo de dato.
- Un método se llama por su nombre, pero está asociado a un tipo de dato y puede o no retornar un valor.
- Para utilizar los métodos en Python la sintaxis es: el dato, seguido de punto y el método que se desea utilizar.

Entonces, **¿Cuál es la diferencia entre métodos y funciones?**

- La principal diferencia es que un método es parte de las propiedades de un tipo de dato específico.
- Cada tipo de dato tiene sus propios métodos. Incluso aunque se llamen igual, pueden tener un comportamiento distinto.
- En cambio, las funciones son entidades independientes en un programa.

Métodos básicos de los strings

Los strings tienen muchos métodos, pero por ahora, nos enfocaremos en los más básicos para entender su funcionamiento.

Mayúsculas y Minúsculas

El método `str.lower()` retorna una copia del string `str` con todos sus caracteres en minúsculas.

El método `str.upper()` retorna una copia del string `str` con todos sus caracteres en mayúsculas.

```
In [22]: x = "Hola Mundo"
```

```
print(x.upper())  
print(x.lower())
```

```
HOLA MUNDO  
hola mundo
```

Ambos métodos retornan una copia del string, no modifican los caracteres del string original.

```
In [23]: mayusculas = x.upper()  
minuscultas = x.lower()
```

```
In [24]: print(mayusculas)  
print(minuscultas)  
print(x)
```

```
HOLA MUNDO  
hola mundo  
Hola Mundo
```

Método `find`

El método `str.find(s)` retorna el índice más bajo donde se encuentra el substring `s` en el string `str`. Si no se encuentra, retorna `-1`.

```
In [25]: "abracadabra".find("a")
```

```
Out[25]: 0
```

```
In [26]: "abracadabra".find("e")
```

```
Out[26]: -1
```

Método `count`

El método `str.count(s)` retorna el número de ocurrencias del substring `s` en el string `str`.

```
In [27]: "abracadabra".count("a")
```

```
Out[27]: 5
```

```
In [28]: "abracadabra".count("e")
```

```
Out[28]: 0
```

Iterable

Los strings son **iterables**, es decir, se pueden recorrer elemento por elemento, a través de un ciclo.

Estructura de Repetición `for`

- Funciona sobre tipos de datos **iterables**.
- Recorre secuencias finitas. Por lo tanto, itera una cantidad **fija** de veces.

Sintaxis genérica de un ciclo `for`

`for` variable `in` secuencia:
 sentencias para cada elemento de la secuencia

- Cada vez que se utiliza un ciclo `for`, se define automáticamente una `variable` cuyo valor será cada uno de los elementos de la `secuencia`, en orden, uno por uno.
- La `secuencia` debe ser un tipo de dato **iterable** (Por ejemplo, un string o una lista).
- No confundir el `" in "` que se utiliza en el `for` con el operador para verificar membresía.

Ejemplo

```
In [29]: vocales = "aeiou"
for letra in vocales:
    print(letra)
```

```
a
e
i
o
u
```

La variable `letra` de forma automática va iterando sobre cada caracter del texto.

Equivalencia entre ciclo `while` y ciclo `for`.

El ciclo `while` también se puede utilizar para recorrer una secuencia. Sin embargo, no itera sobre los elementos de forma automática como el `for`.

```
In [30]: vocales = "aeiou"
i = 0
while i < len(vocales):
    letra = vocales[i]
    print(letra)
    i += 1
```

```
a
e
i
o
u
```