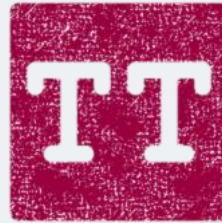


NIVEL 3

MÁS SOBRE STRINGS





RECORDAMOS...

OPERADORES Y FUNCIONES DE PHYTON

- Operador `+` (concatenación de cadenas)
- Operador `*` (repetición de cadena)
- Operadores `in` y `not in` (comprueban si una cadena forma parte o no de otra cadena)
- `int` (conversión de cadena a entero)
- `float` (conversión de cadena a flotante)
- `str` (conversión de entero o flotante a cadena)
- `ord` (conversión de una cadena compuesta de un único carácter a código ascii -entero)
- `chr` (conversión de un entero a una cadena con el carácter que tiene a dicho entero como código ascii)
- `len` (longitud de una cadena)

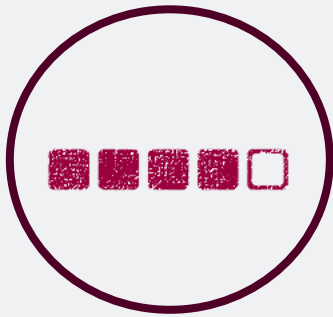


FUNCIONES PROPIAS DE LA CLASE STRING (MÉTODOS)



- Si `a` es una cadena
 - `a.lower()` devuelve una nueva cadena con los caracteres de `a` convertidos en minúscula
 - `a.upper()` devuelve una nueva cadena con los caracteres de `a` convertidos en mayúscula
 - `a.title()` devuelve una cadena en la que toda palabra de `a` empieza por mayúscula
 - `a.replace(str1, str2)` reemplaza las ocurrencias de `str1` en `a` por `str2`
 - `a.format(expr1, expr2, ...)` devuelve una cadena en la que las marcas de formato de `a` se sustituyen por el resultado de evaluar las expresiones dadas

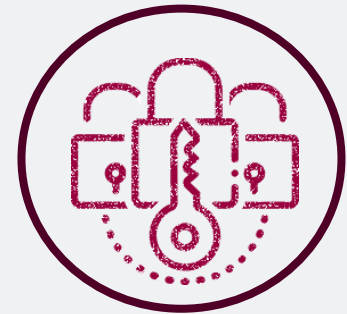
LO NUEVO SOBRE STRINGS



Operador de
indexación



Subcadenas



Inmutabilidad

INDEXACIÓN

0	1	2	3	4	5	6	7	8	9	10	11
H	o	l	a	,		m	u	n	d	o	.

- ✓ Podemos usar variables como índices, para acceder a los caracteres de una cadena
- ✓ El último carácter de la cadena almacenada en la variable `a` no es `a[len(a)]`, sino `a[len(a)-1]`
- ✓ Si intentamos acceder al elemento `a[len(a)]` Python protesta. El error es de tipo `IndexError` y el mensaje indica que el índice de la cadena está fuera del rango de valores válidos

```

Terminal 1/A x
In [2]: "Hola, mundo."[0]
Out[2]: 'H'

In [3]: "Hola, mundo."[1]
Out[3]: 'o'

In [4]: a = "Hola, mundo."

In [5]: a[2]
Out[5]: 'l'

In [6]: a[1]
Out[6]: 'o'

In [7]: i = 3

In [8]: a[i]
Out[8]: 'a'

In [9]: a[len(a)-1]
Out[9]: '.'

In [10]: a[len(a)]
Traceback (most recent call last):
  File "<ipython-input-10-836f840eaf9f>", line 1, in <module>
    a[len(a)]
IndexError: string index out of range

```

INDEXACIÓN

0	1	2	3	4	5	6	7	8	9	10	11
H	o	l	a	,	\n	m	u	n	d	o	.
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Los caracteres de control como el `\n` ocupan una sola casilla

Podemos utilizar índices negativos. Los valores negativos acceden a los caracteres de derecha a izquierda. El último carácter de una cadena tiene índice `-1`, el penúltimo, `-2`, y así sucesivamente.

```

Terminal 1/A ✕

In [15]: a = "Hola,\nmundo."

In [16]: print(a)
Hola,
mundo.

In [17]: a[-1]
Out[17]: '.'

In [18]: a[len(a)-1]
Out[18]: '.'

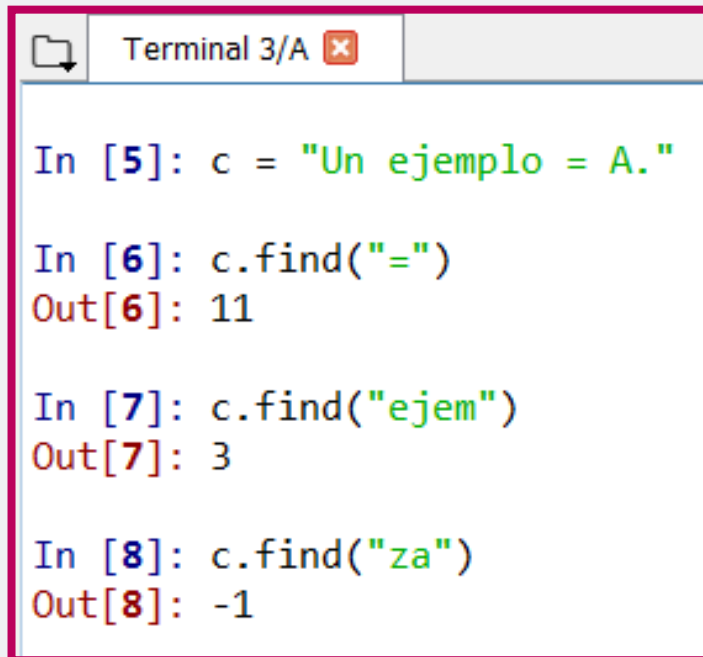
In [19]: a[-3]
Out[19]: 'd'

In [20]: a[-len(a)]
Out[20]: 'H'

```

EL MÉTODO FIND

El método `find` recibe una cadena y nos dice si esta aparece o no en la cadena sobre la que se invoca. Si está, nos devuelve el índice de su primera aparición. Si no está, devuelve el valor `-1`



```
Terminal 3/A ✕  
  
In [5]: c = "Un ejemplo = A."  
  
In [6]: c.find("=")  
Out[6]: 11  
  
In [7]: c.find("ejem")  
Out[7]: 3  
  
In [8]: c.find("za")  
Out[8]: -1
```


SUBCADENAS

```
Terminal 1/A x  
In [64]: s = "Piratas del Caribe"  
In [65]: x = s[0:7]  
In [66]: x  
Out[66]: 'Piratas'  
In [67]: y = s[8:11]  
In [68]: y  
Out[68]: 'del'  
In [69]: z = s[12:18]  
In [70]: z  
Out[70]: 'Caribe'
```

El operador `[n:m]` devuelve la parte de la cadena comprendida entre el n-ésimo carácter y el m-ésimo carácter, incluyendo el primero (`n`) pero excluyendo el último (`m`)

SUBCADENAS

m	a	n	z	a	n	a
0	1	2	3	4	5	6

```
Terminal 1/A [X]

In [1]: fruta = "manzana"
In [2]: inicio = fruta[:3]
In [3]: inicio
Out[3]: 'man'

In [4]: final = fruta[3:]
In [5]: final
Out[5]: 'zana'

In [6]: fruta[:]
Out[6]: 'manzana'
```

Si se omite el primer índice (antes de los dos puntos), la subcadena empieza al inicio de la cadena

Si se omite el segundo índice (después de los dos puntos), la subcadena empieza en el primer índice y va hasta el final de la cadena

LAS CADENAS SON INMUTABLES

Esto quiere decir que no podemos cambiar una cadena existente. Si queremos modificar una cadena, debemos crear una nueva copia que sea una modificación de la original

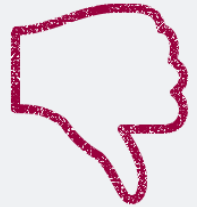
```
Terminal 2/A x

In [22]: saludo = "Hola mi jente"

In [23]: saludo[8] = "g"
Traceback (most recent call last):

  File "<ipython-input-23-24ebb290711c>", line 1, in <module>
    saludo[8] = "g"

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



LAS CADENAS SON INMUTABLES

Esto quiere decir que no podemos cambiar una cadena existente. Si queremos modificar una cadena, debemos crear una nueva copia que sea una modificación de la original

```
Terminal 2/A ✕  
  
In [27]: saludo = "Hola mi gente"  
  
In [28]: correccion = saludo[:8] + "g" + saludo[9:]  
  
In [29]: correccion  
Out[29]: 'Hola mi gente'
```



EJEMPLO: CONTAR LAS OCURRENCIAS DE UN CARÁCTER DENTRO DE UNA CADENA

```
EjemploOcurrenciasCaracterConWhile.py x
1 def ocurrencias_caracter(cadena: str, caracter: str) -> int:
2
3     ocurrencias = 0
4     i = 0
5
6     while ( i < len(cadena) ):
7         if (cadena[i] == caracter):
8             ocurrencias += 1
9             i += 1
10
11     return ocurrencias
```

Desde la posición cero

Mientras no se haya
llegado al final de la cadena

Resultado de la ejecución

Si el carácter en la posición *i*
es igual al carácter buscado,
se incrementa el contador

```
Terminal de IPython
Terminal 1/A x

In [7]: ocurrencias_caracter("La Casa Blanca","a")
Out[7]: 5

In [8]: ocurrencias_caracter("La Casa Blanca","B")
Out[8]: 1

In [9]: ocurrencias_caracter("La Casa Blanca","b")
Out[9]: 0

In [10]: ocurrencias_caracter("La Casa Blanca","A")
Out[10]: 0
```