

NIVEL 2

OPERACIONES SOBRE CADENAS DE
CARACTERES O STRINGS



COMPARACIÓN DE CADENAS



Los operadores relacionales funcionan sobre cadenas

```
EjemploComparacionCadenas.py ✖  
1 def comparar_cadenas(palabra1: str, palabra2: str) -> None:  
2     if (palabra1 == palabra2):  
3         print("Las palabras son iguales")  
4     elif (palabra1 < palabra2):  
5         print("La palabra", palabra1, "es menor que la palabra", palabra2)  
6     else:  
7         print("La palabra", palabra1, "es mayor que la palabra", palabra2)
```

COMPARACIÓN DE CADENAS

Python utiliza un criterio de comparación de cadenas similar al orden alfabético

- ✓ La comparación se hace carácter a carácter, usando la codificación ASCII

```
Terminal 2/A x
In [3]: comparar_cadenas("alma","blanca")
La palabra alma es menor que la palabra blanca

In [4]: comparar_cadenas("salvaje","potro")
La palabra salvaje es mayor que la palabra potro

In [5]: comparar_cadenas("hola","Hola")
La palabra hola es mayor que la palabra Hola

In [6]: comparar_cadenas("Hola","Hola")
Las palabras son iguales
```

```
Terminal 2/A x
In [19]: ord("a")
Out[19]: 97

In [20]: ord("á")
Out[20]: 225
```

```
Terminal 2/A x
In [12]: "Barco" < "ancla"
Out[12]: True

In [13]: "ábaco" < "ajo"
Out[13]: False

In [14]: "a" < "aa"
Out[14]: True

In [15]: "aa" > "ab"
Out[15]: False

In [16]: "abajo" < "arriba"
Out[16]: True
```

El primer carácter de las dos cadenas es igual, pero la primera no tiene más caracteres, la segunda es mayor

OPERADORES

El operador **in** comprueba si una cadena forma parte o no de otra cadena



Una cadena es subcadena de ella misma

La cadena vacía es subcadena de cualquier cadena

```
Terminal 2/A x

In [37]: "n" in "manzana"
Out[37]: True

In [38]: "p" in "manzana"
Out[38]: False

In [39]: "za" in "manzana"
Out[39]: True

In [40]: "pa" in "manzana"
Out[40]: False

In [41]: "a" in "a"
Out[41]: True

In [42]: "manzana" in "manzana"
Out[42]: True

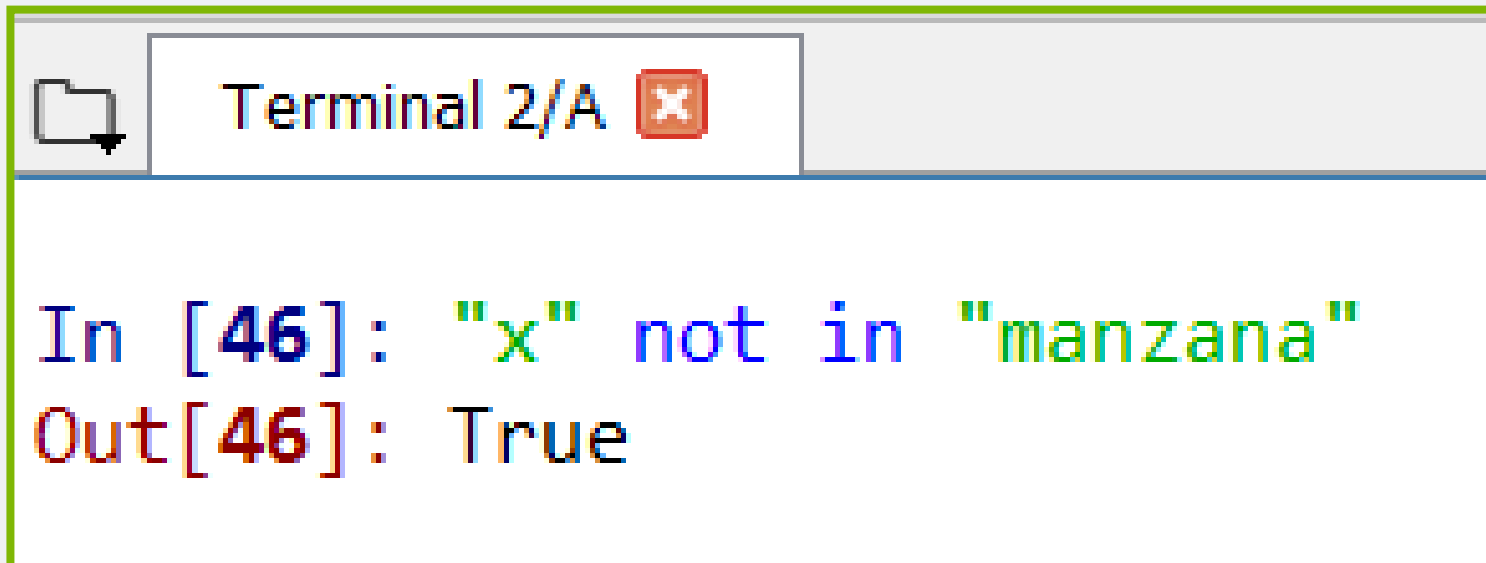
In [43]: "" in "a"
Out[43]: True

In [44]: "" in "manzana"
Out[44]: True
```

OPERADORES

El operador **not in** es el opuesto lógico de **in**

not in



```
Terminal 2/A [X]  
  
In [46]: "x" not in "manzana"  
Out[46]: True
```

CADENAS DE CARACTERES – REPASO Y OTRAS FUNCIONES



- ✓ Operador **+** (concatenación de cadenas)
- ✓ Operador ***** (repetición de 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)

FUNCIONES PROPIAS

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.swapcase()** devuelve una nueva cadena con los caracteres en minúscula convertidos a mayúscula y viceversa

MÉTODOS PARA MANIPULAR CADENAS



- Los datos de tipo String permiten invocar unas funciones especiales: los denominados «**métodos**». Los métodos son funciones especiales, pues se invocan del siguiente modo:
- String.método(argumento1, argumento2...)

upper: devuelve una nueva cadena con los caracteres de a convertidos en mayúscula

```
Terminal 1/A ✕  
  
In [22]: ss = "Hola, mundo!"  
  
In [23]: tt = ss.upper()  
  
In [24]: tt  
Out[24]: 'HOLA, MUNDO!'  
  
In [25]: ll = tt.lower()  
  
In [26]: ll  
Out[26]: 'hola, mundo!'
```

lower: devuelve una nueva cadena con los caracteres de a convertidos en minúscula

MÉTODOS PARA MANIPULAR CADENAS

```
Terminal 1/A ✕  
  
In [30]: ss = "Hola, mundo!"  
  
In [31]: mm = ss.title()  
  
In [32]: mm  
Out[32]: 'Hola, Mundo!'  
  
In [33]: xx = ss.swapcase()  
  
In [34]: xx  
Out[34]: 'hOLA, MUNDO!'
```

• **title**: devuelve una nueva cadena en la que toda palabra empiece por mayúscula

• **swapcase**: devuelve una nueva cadena con los caracteres en minúscula convertidos a mayúscula y viceversa

INMUTABILIDAD

- ✓ Cuando en Python se construye una cadena no puede cambiar su contenido
- ✓ Si queremos hacerle un cambio, lo único que Python puede hacer es construir una nueva cadena con los cambios

```
Console 1/A
Python 3.7.6 (default, Jan  8 2020, 13:42:34)
Type "copyright", "credits" or "license" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: original = "texto original"
...: minusculas = original.lower()
...: print("Son iguales:", original == minusculas)
...: print("Son el mismo:", original is minusculas)
Son iguales: True
Son el mismo: False
```

MÉTODOS PARA MANIPULAR CADENAS: REPLACE

El método **replace**, recibe como parámetro dos cadenas: un patrón y un reemplazo. El método busca el patrón en la cadena sobre la que se invoca el método y sustituye todas sus apariciones por la cadena de reemplazo

Terminal de IPython



Terminal 2/A



```
In [8]: aa = "Un pequeño ejemplo"
```

```
In [9]: bb = aa.replace("pequeño", "gran")
```

```
In [10]: bb
```

```
Out[10]: 'Un gran ejemplo'
```

MÉTODO FIND

```
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
```

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

MÉTODO COUNT

```
In [3]: "Hola, Mundo!".count("o")  
Out[3]: 2
```

```
In [4]: ejemplo = "Hola, Mundo!"
```

```
In [5]: ejemplo.count("o")  
Out[5]: 2
```

```
In [6]: str.count(ejemplo, "o")  
Out[6]: 2
```

El método **count** cuenta cuántas veces está la cadena indicada en otra cadena. Si no la encuentra devuelve cero

TODOS LOS MÉTODOS PARA MANIPULAR CADENAS

```
Terminal 3/A
In [35]: help('str')
Help on class str in module builtins:

class str(object)
|   str(object='') -> str
|   str(bytes_or_buffer[, encoding[, errors]]) -> str
|
|   Create a new string object from the given object. If encoding or
|   errors is specified, then the object must expose a data buffer
|   that will be decoded using the given encoding and error handler.
|   Otherwise, returns the result of object.__str__() (if defined)
|   or repr(object).
|   encoding defaults to sys.getdefaultencoding().
|   errors defaults to 'strict'.
|
|   Methods defined here:
|
|   __add__(self, value, /)
|       Return self+value.
|
|   __contains__(self, key, /)
|       Return key in self.
|
|   __eq__(self, value, /)
|       Return self==value.
```

Para conocer TODOS los métodos para manipular cadenas, usando `help('str')` en la terminal de Spyder aparecerá toda la información

MÉTODO FORMAT

El método format es la forma más fácil y poderosa de formatear una cadena. Miremos cómo funciona con algunos ejemplos:

- ✓ La «cadena plantilla» contiene unas marcas de formato: {0} ... {1} ... {2} ... etc.
- ✓ El método `format` sustituye estas marcas por el valor de sus argumentos
- ✓ El número de una marca es un índice que determina cuál argumento va a sustituir la marca

```
In [39]: n1 = 4
```

```
In [40]: s3 = "2**10 = {0} y {1} * {2} = {3:.3f}".format(2**10, n1, n2, n1*n2)
```

```
In [41]: print(s3)
```

```
2**10 = 1024 y 4 * 5 = 20.000
```

MÉTODO FORMAT

Cada una de las marcas puede contener también una **especificación de formato**. Esta especificación determina cómo deben hacerse las sustituciones en la cadena plantilla

La **especificación de formato** también controla cosas como:

- ✓ Si el texto es alineado a la **izquierda** `<`, **centrado** `^`, o a la **derecha** `>`
- ✓ El ancho reservado en la cadena resultante (un número como 10 por ejemplo)
- ✓ El tipo de conversión (por ejemplo, `f` de float). Si el tipo de conversión es float, se pueden especificar la cantidad de decimales (ejemplo: `.3f` para 3 decimales)

```
In [39]: n1 = 4
```

```
In [40]: s3 = "2**10 = {0} y {1} * {2} = {3:.3f}".format(2**10, n1, n2, n1*n2)
```

```
In [41]: print(s3)
```

```
2**10 = 1024 y 4 * 5 = 20.000
```

MÁS EJEMPLOS

Los ejemplos que vamos a ver a continuación son suficientes para la mayoría de las necesidades de formateo. En caso de necesitar algo más sofisticado teclee `help('FORMATTING')` en la consola de Spyder y ahí encontrará toda la información



Formato flotante con 3 decimales

EjemploFormat.py

```

n1 = "Simon José Antonio"
n2 = "de la Santísima Trinidad"
n3 = "Bolívar y Palacios"

print("Pi con tres decimales es {0:.3f}".format(3.1415926))
print("123456789 123456789 123456789 123456789 123456789 123456789")
print("|||{0:<25}|||{1:^25}|||{2:>25}|||nació en {3}|||".format(n1,n2,n3,1783))

```

Alineación a la izquierda y
espacio reservado de 25
caracteres

Alineación al centro y
espacio reservado de 25
caracteres

Alineación a la derecha y
espacio reservado de 25
caracteres

Resultado de la ejecución

Terminal 4/A

```

In [7]: runfile('C:/Users/Mhernandez/Desktop/IP/N2-C3/EjemploFormat.py', wdir='C:/Users/Mhernandez/Desktop/N2-C3')
Pi con tres decimales es 3.142
123456789 123456789 123456789 123456789 123456789 123456789
|||Simon José Antonio      |||de la Santísima Trinidad|||      Bolívar y Palacios|||nació en 1783|||

```

Se pueden tener múltiples marcas para el mismo argumento, o argumentos que no son usados:

```
Ejemplo2Format.py ✕  
1 carta = ""  
2 Querido {0} {2}.  
3 {0}, Tengo una propuesta financiera muy interesante para hacerle!  
4 Si usted deposita $10 millones de dólares en mi cuenta bancaria, puedo  
5 duplicar su dinero ...  
6 ""  
7  
8 print(carta.format("Julio", "Mario", "Santodomingo"))  
9 print(carta.format("Luis Carlos", "Sarmiento", "Angulo"))
```

```
Terminal 1/A ✕  
  
In [1]: runfile('C:/Users/Mhernandez/Desktop/IP/N2-C3/Ejemplo2Format.py', w  
IP/N2-C3')  
  
Querido Julio Santodomingo.  
Julio, Tengo una propuesta financiera muy interesante para hacerle!  
Si usted deposita $10 millones de dólares en mi cuenta bancaria, puedo  
duplicar su dinero ...  
  
Querido Luis Carlos Angulo.  
Luis Carlos, Tengo una propuesta financiera muy interesante para hacerle!  
Si usted deposita $10 millones de dólares en mi cuenta bancaria, puedo  
duplicar su dinero ...
```

Resultado de
la ejecución