

#### Concepto

- Una cadena de caracteres (string) es un conjunto de 0 o más caracteres.
- Python 3 utiliza codificación UTF-8 para representar caracteres.
- UTF-8 soporta caracteres regionales como la ñ, las vocales con tilde, etc.



 Las constantes de cadena de caracteres pueden encerrarse indistintamente entre comillas simples o dobles.

cad1 = "Lunes"

cad2 = 'Martes'

@ Lic. Ricarde Thempsen

#### Constantes de cadena

 Esto permite utilizar el otro tipo de comillas dentro de la cadena.

marca = "Mc. Donald's"

frase = 'Dijo "Ya voy" y partió'



 Una constante de cadena extensa puede ser distribuida en varias líneas, usando una barra invertida como continuación.

zen = "Hermoso es mejor que feo. \
Explícito es mejor que implícito."
print(zen)

@ Lic. Ricarde Thempsen

#### Constantes de cadena

 Si se utilizan tres juegos de comillas la cadena retiene el formato dado por el programador y puede extenderse por múltiples líneas.



despacito = '''Tu, tú eres el imán y yo soy el metal Me voy acercando y voy armando el plan Sólo con pensarlo se acelera el pulso''' print(despacito)

Tu, tú eres el imán y yo soy el metal Me voy acercando y voy armando el plan Sólo con pensarlo se acelera el pulso

@ Lie. Ricarde Thempsen

# Impresión de cadenas

 Para imprimir cadenas de caracteres mediante el operador % se utiliza el especificador de conversión %s.

nombre = input("Ingrese su nombre: ")
edad = int(input("Ingrese su edad: "))
print("%s tiene %d años" %(nombre, edad))

También puede escribirse el ancho deseado entre el signo % y la letra s.

# Secuencias de escape

- Las secuencias de escape se utilizan para representar caracteres con significados especiales.
- Toda secuencia de escape comienza con una barra invertida: \
- El carácter que se escriba después de ella establece el significado de la secuencia de escape.

@ Lie. Ricarde Thempsen

# Secuencias de escape

#### **Ejemplos:**

\n: Salto de línea

**\b: Retroceso** 

**\t: Tabulador** 

\": Comilla doble

\': Comilla simple

\a: Campanilla (Sólo

(Sólo en algunos Python, por ej. Anaconda)



Una cadena de caracteres puede contener secuencias de escape, las que serán interpretadas según su significado.

print("Lunes\nMartes\nMiércoles")

Lunes Martes Miércoles

@ Lie. Ricarde Thempsen

#### **Cadenas crudas**

 Una cadena cruda (raw) se crea escribiendo la letra r antes de abrir las comillas.

ruta = r"c:\nuevo\datos.txt"

En una cadena cruda <u>no se</u>
<u>interpretan</u> las secuencias de escape.



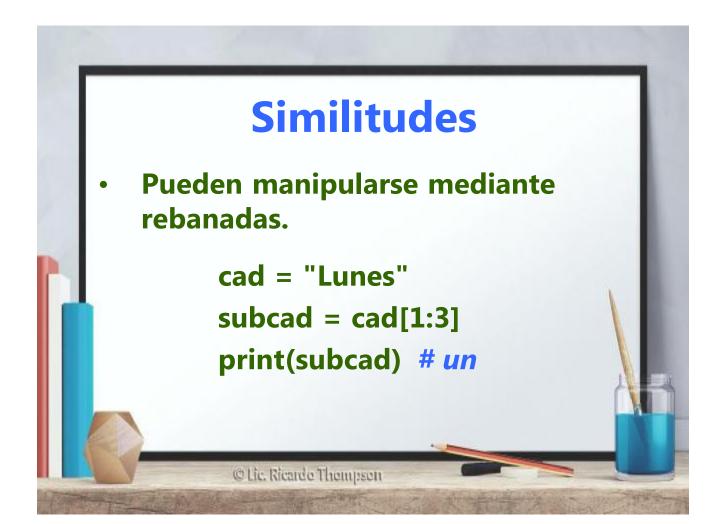
- Las cadenas de caracteres y las listas tienen muchas cosas en común, pero también algunas diferencias.
- Ambas son secuencias de elementos, es decir iterables.

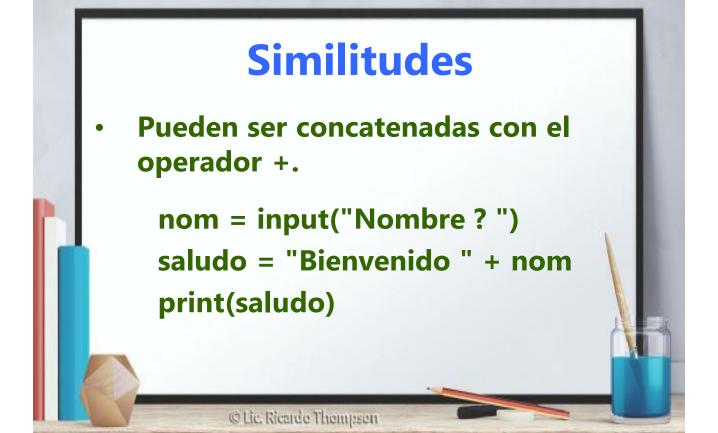
@ Lie. Ricarde Thempsen

#### **Similitudes**

 Se puede acceder a cada elemento a través de un subíndice (base 0).

cad = "Lunes"
print(cad[2]) # n





#### **Similitudes**

 Pueden replicarse mediante el operador \*.

```
risa = "ja"
print(risa * 5) # jajajaja
separador = "-" * 80
```

@ Lie. Ricarde Thempsen

#### **Similitudes**

Comparten muchas funciones y métodos:

cad = "Lunes"

len( ): print(len(cad)) # 5

in: if "es" in cad:

not in: if "es" not in cad:

# cad = "Lunes" max(): print(max(cad)) # u min(): print(min(cad)) # L index(): print(cad.index("es")) # 3 count(): print(cad.count("es")) # 1

#### **Diferencias**

- Las listas son mutables, pero las cadenas son inmutables.
- Esto significa que <u>no pueden ser</u> modificadas.
- Las funciones y métodos para manipularlas devuelven una nueva cadena, sin alterar la original.

#### **Diferencias**

- Debido a que son inmutables, las cadenas carecen del método reverse de las listas.
- Para invertir una cadena se puede utilizar la técnica de las rebanadas.

original = "Hola"

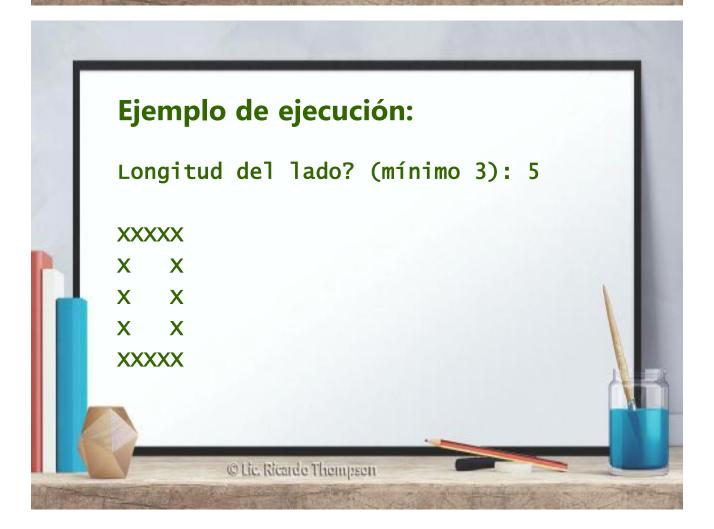
invertida = original[::-1] # aloH

@ Lie. Ricardo Thempsen

# **Ejemplo 1**

Dibujar el borde de un cuadrado con letras X, ingresando por teclado la longitud de cada lado.

```
lado = int(input("Longitud del lado? (mínimo 3): "))
while lado < 3:
    print("Lado inválido. Intente nuevamente")
    lado = int(input("Longitud del lado? (mínimo 3): "))
print()
print("X" * lado)  # borde superior
relleno = "X" + " " * (lado-2) + "X"
for i in range(lado-2): # laterales
    print(relleno)
print("X" * lado)  # borde inferior
```



# Comparación

- Las cadenas de caracteres pueden ser comparadas entre sí igual que cualquier otra variable.
- Se utiliza comparación alfabética (como en un diccionario).

@ Lic. Ricarde Thempsen

#### Conversión

- Un número no puede ser concatenado a una cadena en forma directa.
- Antes es necesario convertirlo a cadena, lo que se logra con la función str().

c = 75 mensaje = "Precio: " + str(c) + " dólares" print(mensaje) # Precio: 75 dólares



- Así como existe la función str() para convertir números en cadenas, existen las funciones int() y float() que permiten efectuar la conversión en sentido contrario.
- Si la cadena no contiene un número válido se producirá un error.

© Lie. Ricarde Thempsen

#### Conversión

int(): Convierte una cadena en entero

float(): Convierte una cadena en real

# Ejemplo 2

Eliminar los signos de puntuación de una cadena de caracteres

```
signos = ".,:;?!"
frase = input("Ingrese una frase: ")
nueva = ""
for caracter in frase:
    if caracter not in signos:
        nueva = nueva + caracter
print(nueva)
```

# Métodos para cadenas

<str>.split([<sep>]): Divide <str> en una lista de cadenas, buscando <sep> como separador. Si <sep> se omite se asumen los espacios.

```
a = "Hola Mundo"
```

```
b = a.split() # ['Hola', 'Mundo']
```

c, d = a.split() # 'Hola' y 'Mundo'

@ Lie. Ricarde Thempsen

# Métodos para cadenas

#### **Ejemplos:**

```
cad = "Hola Mundo" # 3 espacios
lista1 = cad.split()
print(lista1) # ['Hola', 'Mundo']
lista2 = cad.split(' ')
print(lista2) # ['Hola', '', '', 'Mundo']
```

# Métodos para cadenas

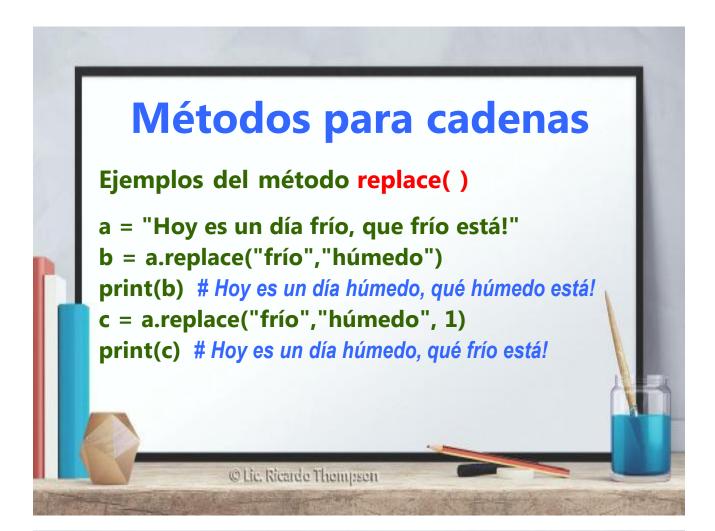
<sep>.join(<secuencia>): Devuelve una cadena con los elementos de <secuencia> separados por <sep>. <secuencia> puede ser una cadena o una lista de cadenas.

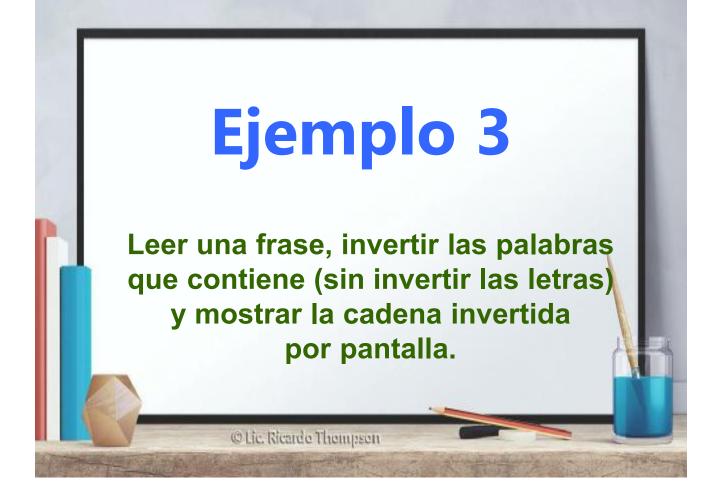
cad = ', '.join('abc')
print(cad) # a, b, c

@ Lie. Ricarde Thempsen

# Métodos para cadenas

<str>.replace(<vie>,<nue>[,<max>]):
 Devuelve una cadena nueva
 reemplazando todas las apariciones de
 <vie> por <nue>, hasta un máximo de
 <max> reemplazos. Este último
 parámetro es opcional. Si se omite se
 reemplazan todas las apariciones. No da
 error si <vie> no existe.





```
frase = input("Ingrese una frase: ")
# Dividimos la frase en una lista de palabras
palabras = frase.split()
# Invertimos la lista con reverse()
palabras.reverse()
# Construimos la cadena a partir de la lista
frase = " ".join(palabras)
print(frase)
# También se puede usar un ciclo para concatenar
frase2 = ""
for palabra in palabras:
    frase2 = frase2 + palabra + " "
print(frase2)
```

# Copia de cadenas

Copiar una cadena siempre copia su referencia -es decir su dirección de memoria- sin importar la técnica que se utilice.

# Copia de cadenas

```
a = "Hola"
b = a
c = a[:]
d = str(a)
e = a + ""
for i in map(id, [a, b, c, d, e]):
    print(i)

162615112
162615112
162615112
```

@ Lie. Ricarde Thempsen

# Copia de cadenas

- Esto se debe a que las cadenas son inmutables, por lo que crear copias idénticas sólo desperdiciaría memoria.
- Sin embargo, con algo de ingenio es posible romper esta regla.



 Por ejemplo, utilizando el método join().

```
a = "Hola"
b = "".join(a)
print(id(a), id(b)) # 162615112 162612480
```

De este modo se puede crear un clon de una cadena.

@ Lic. Ricarde Thempsen

# **Ejercitación**

Práctica 4: Ejercicios 1 a 9