

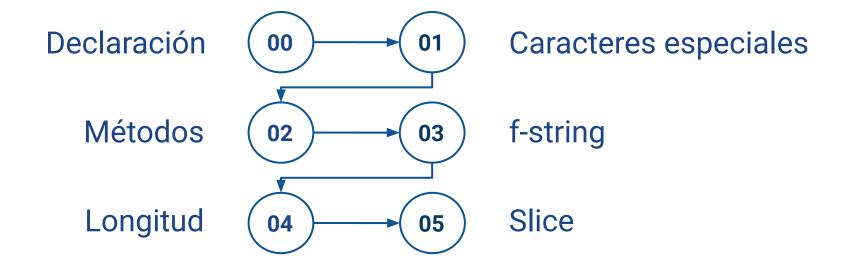


Strings

Programación y Laboratorio I



Strings





Declaración

Las cadenas en Python o strings son un tipo inmutable que permite almacenar secuencias de caracteres. Para crear una, es necesario incluir el texto entre comillas dobles

```
texto = "Hola Mundo"
print(texto) #Hola Mundo
print(type(texto)) #<class 'str'>
```



Declaración

También es válido declarar las cadenas con comillas simples simples

```
texto = 'Hola Mundo'
print(texto) #Hola Mundo
print(type(texto)) #<class 'str'>
```



Declaración

En el caso de requerir una comilla, bien sea simple ' o doble " dentro de una cadena, se puede...

```
texto = 'Hola " " Mundo'
print(texto) #Hola " " Mundo
print(type(texto)) #<class 'str'>
```



Caracteres especiales

Para incluir un salto de línea dentro de una cadena.

```
texto = 'Hola \n Mundo'
print(texto)
#Hola
# Mundo
```



Métodos de string

Un método es una función especial, que existe para un tipo de dato en particular.

Para trabajar con cadenas de texto en Python, se emplean una serie de métodos a las variables del tipo str.



Métodos de string

variable.strip(): El método strip eliminará todos los caracteres vacíos que pueda contener la variable

variable.lower(): El método lower convertirá a las letras en minúsculas.



Método: strip

El método **strip** eliminará todos los caracteres vacíos que pueda contener la variable

```
cadena = " Hola Mundo "
cadena = cadena.strip()
print(cadena) # Hola Mundo
```



Método: lower

El método **lower** convertirá a las letras en minúsculas.

```
cadena = "Hola Mundo"
cadena = cadena.lower()
print(cadena) # hola mundo
```



Método: upper

El método **upper** convertirá a las letras en mayúsculas.

```
cadena = "Hola Mundo"
cadena = cadena.upper()
print(cadena) # HOLA MUNDO
```



Método: capitalize

El método capitalize convertirá a la primera letra de la Strings en mayúscula y el resto en minúscula.

```
cadena = "hola Mundo"
cadena = cadena.capitalize()
print(cadena) # Hola mundo
```



Método: replace

El método replace remplazará un conjunto de caracteres por otro.

```
cadena = "Hola Mundo"
cadena = cadena.replace("la","@")
print(cadena) # Ho@ Mundo
```



Método: split

El método **split** divide una cadena en subcadenas y las devuelve almacenadas en una lista.

```
cadena = "Python, Java, C"
print(cadena.split(","))
#['Python', 'Java', 'C']
```



Método: join

El método join devuelve la primera cadena unida a cada uno de los elementos de la lista que se le pasa como parámetro.

```
cadena = "+"
cadena = cadena.join(["A", "B", "C"])
print(cadena) # A+B+C
```



Método: zfill

El método **zfill** rellena la cadena con ceros a la izquierda hasta llegar a la longitud pasada como parámetro.

```
cadena = "314"
print(cadena.zfill(6))
#000314
```



Método: isalpha

El método **isalpha** devuelve True si todos los caracteres son alfabéticos, False de lo contrario.

```
cadena = "Hola Mundo"
print(cadena.isalpha())
# False -> por el espacio

cadena = "HolaMundo"
print(cadena.isalpha())
# True
```



Método: isalnum

El método **isalnum** devuelve True si todos los caracteres son alfanuméricos, False de lo contrario.

```
cadena = "Hola Mundo 123"
print(cadena.isalnum())
# False -> por el espacio
cadena = "HolaMundo123"
print(cadena.isalnum())
# True
```



Método: count

El método count permite contar las veces que otra cadena se encuentra dentro de la primera.

```
cadena = "Hola Mundo Hola"
print(cadena.count("la")) # 2
```



Método: format

En el método format las llaves, llamadas campos de formato, son reemplazadas con los valores de las variables pasadas.

```
nombre_usuario="JUAN"
edad_usuario=35
cadena = "Nombre: {1}, Edad: {0}"
print(cadena.format(edad_usuario,nombre_usuario))
#Nombre: JUAN, Edad: 35
```



f-strings

Las cadenas literales o f-strings, permiten incrustar expresiones dentro de cadenas.

```
n_usuario="JUAN"
e_usuario=35
cadena = f"Nombre: {n_usuario}, Edad: {e_usuario}"
print(cadena)
#Nombre: JUAN, Edad: 35
```



Longitud: len

El método **len** indica la longitud de la cadena de texto dentro de la variable en ese momento.

```
cadena = "Hola Mundo"
print(len(cadena)) # 10
```



Slice

Cuando se crea una **slice** (rebanada), el primer número es donde comienza (inclusivo), y el segundo número de índice es donde termina (exclusivo).

```
cadena = "Hola Mundo"
print(cadena[5:10]) # Mundo
print(cadena[5:]) # Mundo
print(cadena[:5]) # Hola
```