Strings

Algoritmos y Programación Básica (CC2005) - 2026

Strings

Semestre 02, 2026

Strings como tipo de dato

En Python, un string es una secuencia ordenada de caracteres.

Cada carácter ocupa una posición numerada llamada índice.

Esa estructura los convierte en un tipo de dato poderoso para trabajar con texto.

Lo que ya se saben

```
nombre = "María"
mensaje = 'Hola, mundo'
```

Un string se puede almacenar en una variable como cualquier otro dato.

Qué se puede hacer con un string

- Obtener su longitud
- Acceder a caracteres individuales
- Extraer partes del texto

- Recorrerlo con un ciclo
- Transformarlo con métodos

Longitud

La función `len()` devuelve el número de caracteres de un string.

Ejemplo

```
texto = "Python"
print(len(texto))
```

Resultado:

```
6
```

Espacios y caracteres especiales cuentan

```
print(len("Hola mundo"))  # 10 — el espacio cuenta
print(len(""))             # 0 — string vacío
print(len("¡Hola!"))      # 6
```

Uso práctico

```
contrasena = input("Escribe tu contraseña: ")

if len(contrasena) < 8:
    print("La contraseña es demasiado corta.")
else:
    print("Contraseña aceptada.")
```

Índices

Cada carácter de un string tiene una posición.

Los índices empiezan en `0`.

Ejemplo visual

```
texto = "Python"

 P  y  t  h  o  n
 0  1  2  3  4  5
```

Acceder a un carácter

```
texto = "Python"

print(texto[0])   # P
print(texto[1])   # y
print(texto[5])   # n
```

Índices negativos

Los índices negativos cuentan desde el final.

```
 P  y  t  h  o  n
-6 -5 -4 -3 -2 -1
```

```
texto = "Python"

print(texto[-1])  # n
print(texto[-2])  # o
print(texto[-6])  # P
```

Error común

```
texto = "Python"
print(texto[6])  # ERROR: IndexError
```

El índice máximo es `len(texto) - 1`.

Slicing

Un slice extrae una porción del string.

Sintaxis: `texto[inicio:fin]`

El carácter en la posición `fin` no se incluye.

Ejemplo básico

```
texto = "Python"

print(texto[0:3])   # Pyt
print(texto[2:5])   # tho
print(texto[1:4])   # yth
```

Omitir inicio o fin

```
texto = "Python"

print(texto[:3])    # Pyt  — desde el inicio hasta índice 2
print(texto[3:])    # hon  — desde índice 3 hasta el final
print(texto[:])     # Python — copia completa
```

Slice con paso

Sintaxis: `texto[inicio:fin:paso]`

```
texto = "Python"

print(texto[::2])    # Pto   — cada 2 caracteres
print(texto[::-1])   # nohtyP — al revés
```

Aplicaciones del slice

```
email = "usuario@uvg.edu.gt"

# Extraer el dominio
print(email[8:])         # uvg.edu.gt

# Extraer el usuario
print(email[:7])         # usuario

# Invertir un texto
texto = "reconocer"
print(texto[::-1])       # reconocer — ¡es un palíndromo!
```

Iterar sobre un string

Un string es una secuencia, por lo que se puede recorrer con `for`.

Recorrer carácter por carácter

```
nombre = "Ana"

for letra in nombre:
    print(letra)
```

Resultado:

```
A
n
a
```

Contar caracteres específicos

```
texto = "murciélago"
vocales = "aeiouáéíóú"
contador = 0

for letra in texto:
    if letra in vocales:
        contador = contador + 1

print("Vocales:", contador)
```

Resultado:

```
Vocales: 5
```

Iterar con índice usando range y len

```
texto = "Python"

for i in range(len(texto)):
    print(i, texto[i])
```

Resultado:

```
0 P
1 y
2 t
3 h
4 o
```

```
5 n
```

Verificar si un string contiene algo

El operador `in` funciona igual que con caracteres.

```
frase = "El cielo es azul"

if "azul" in frase:
    print("Encontrado")

if "verde" not in frase:
    print("No está")
```

Resultado:

```
Encontrado
No está
```

Inmutabilidad

Un string no puede modificarse después de crearse.

Intentar cambiar un carácter individual produce un error.

Ejemplo

```
texto = "Python"
texto[0] = "p"   # ERROR: TypeError
```

La solución: crear un nuevo string

```
texto = "Python"
nuevo = "p" + texto[1:]

print(nuevo)   # python
```

Se construye un string nuevo combinando partes del original.

Utilidad

```
nombre = "Carlos"
nombre.upper()   # esto NO cambia nombre

print(nombre)    # Carlos — sigue igual
```

Los métodos de strings no modifican el original.

Siempre devuelven un nuevo string.

```
nombre = "Carlos"
resultado = nombre.upper()

print(resultado)  # CARLOS
print(nombre)     # Carlos
```

Concatenación y repetición

Los operadores `+` y `*` funcionan con strings.

Concatenación con `+`

```
saludo = "Hola"
nombre = "Luis"
```

```
mensaje = saludo + ", " + nombre + "!"
print(mensaje)
```

Resultado:

```
Hola, Luis!
```

Construir strings en un ciclo

```
resultado = ""

for i in range(1, 6):
    resultado = resultado + str(i) + " "

print(resultado)
```

Resultado:

```
1 2 3 4 5
```

Repetición con `*`

```
linea = "-" * 20
print(linea)

separador = "=-" * 10
print(separador)
```

Resultado:

```
--------------------
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Métodos de strings

Un método es una función que pertenece a un tipo de dato.

Se llama con un punto: `string.metodo()`

Mayúsculas y minúsculas

```
texto = "hola Mundo"

print(texto.upper())       # HOLA MUNDO
print(texto.lower())       # hola mundo
print(texto.capitalize())  # Hola mundo
print(texto.title())       # Hola Mundo
```

Eliminar espacios: strip

```
entrada = "   Python   "

print(entrada.strip())    # "Python"  — elimina ambos lados
print(entrada.lstrip())   # "Python   "  — solo izquierda
print(entrada.rstrip())   # "   Python"  — solo derecha
```

Muy útil al limpiar datos ingresados por el usuario.

```
nombre = input("Tu nombre: ").strip()
```

Reemplazar texto: replace

```
frase = "Me gusta el café con café"

nueva = frase.replace("café", "té")
print(nueva)
```

Resultado:

```
Me gusta el té con té
```

Reemplaza todas las apariciones por defecto.

Buscar texto: find

```
texto = "El gato está en el tejado"

pos = texto.find("gato")
print(pos)   # 3 — índice donde empieza

pos = texto.find("perro")
print(pos)   # -1 — no se encontró
```

`find` devuelve el índice de la primera aparición, o `-1` si no existe.

Contar apariciones: count

```
frase = "ana banana"

print(frase.count("a"))    # 6
print(frase.count("ana"))  # 2
print(frase.count("z"))    # 0
```

Verificar contenido: startswith y endswith

```
archivo = "informe_final.pdf"

if archivo.endswith(".pdf"):
    print("Es un PDF")

correo = "jose@uvg.edu.gt"
```

```
if correo.startswith("jose"):
    print("Correo de José")
```

Dividir un string: split

`split` separa un string en una lista de partes.

```
frase = "manzana naranja uva"

frutas = frase.split()
print(frutas)   # ['manzana', 'naranja', 'uva']
```

split con separador

```
fecha = "2026-04-15"

partes = fecha.split("-")
print(partes[0])   # 2026
print(partes[1])   # 04
print(partes[2])   # 15
```

```
csv = "Pedro,25,Guatemala"

datos = csv.split(",")
print(datos[0])   # Pedro
print(datos[1])   # 25
print(datos[2])   # Guatemala
```

Unir strings: join

`join` une una lista de strings con un separador.

Es el opuesto de `split`.

```
palabras = ["Hola", "desde", "Python"]

resultado = " ".join(palabras)
print(resultado)   # Hola desde Python
```

```
letras = ["P", "y", "t", "h", "o", "n"]

print("".join(letras))    # Python
print("-".join(letras))   # P-y-t-h-o-n
```

Verificar tipo de contenido

```
print("12345".isdigit())    # True  — solo dígitos
print("Hola".isalpha())     # True  — solo letras
print("abc123".isalnum())   # True  — letras y dígitos
print("   ".isspace())      # True  — solo espacios
```

Útil para validar entradas del usuario.

```
edad = input("Tu edad: ")

if edad.isdigit():
    print("Edad:", int(edad))
else:
    print("Eso no es un número.")
```

f-strings

Una f-string permite insertar variables dentro de un string de forma directa.

Se escribe poniendo `f` antes de las comillas.

Sintaxis

```
nombre = "Ana"
edad = 22

mensaje = f"Hola, {nombre}. Tienes {edad} años."
print(mensaje)
```

Resultado:

```
Hola, Ana. Tienes 22 años.
```

Expresiones dentro de f-strings

```
a = 5
b = 3

print(f"La suma de {a} y {b} es {a + b}")
print(f"El cuadrado de {a} es {a  2}")
```

Resultado:

```
La suma de 5 y 3 es 8
El cuadrado de 5 es 25
```

Comparación: con y sin f-string

Sin f-string:

```
print("El área es: " + str(base * altura) + " cm²")
```

Con f-string:

```
print(f"El área es: {base * altura} cm²")
```

Las f-strings son más legibles y directas.

Ejemplo 1: Análisis de texto

```
texto = input("Escribe una frase: ")

texto_limpio = texto.strip()

print(f"Caracteres: {len(texto_limpio)}")
print(f"Mayúsculas: {texto_limpio.upper()}")
print(f"Palabras: {len(texto_limpio.split())}")
print(f"¿Contiene 'Python'? {'Python' in texto_limpio}")
```

Entrada: `" Aprendo Python "`

Resultado:

```
Caracteres: 14
Mayúsculas: APRENDO PYTHON
Palabras: 2
¿Contiene 'Python'? True
```

Ejemplo 2: Validar correo

```
correo = input("Correo electrónico: ").strip().lower()

if "@" not in correo:
    print("Falta el símbolo @")
elif not correo.endswith(".com") and not correo.endswith(".gt"):
    print("Dominio no reconocido")
elif correo.startswith("@"):
    print("Falta el usuario")
else:
```

```
    partes = correo.split("@")
    usuario = partes[0]
    dominio = partes[1]
    print(f"Usuario: {usuario}")
    print(f"Dominio: {dominio}")
    print("Correo válido.")
```

Ejemplo 3: Contar vocales y consonantes

```
texto = input("Escribe una palabra: ").lower()
vocales = "aeiouáéíóú"
total_vocales = 0
total_consonantes = 0

for letra in texto:
    if letra in vocales:
        total_vocales = total_vocales + 1
    elif letra.isalpha():
        total_consonantes = total_consonantes + 1

print(f"Vocales: {total_vocales}")
print(f"Consonantes: {total_consonantes}")
```

Entrada: `"murciélago"`

Resultado:

```
Vocales: 5
Consonantes: 5
```

Errores comunes

Querer modificar un carácter directamente

```
texto = "Hola"
texto[0] = "h"   # ERROR: los strings son inmutables
```

Solución: construir un nuevo string.

```
texto = "h" + texto[1:]
```

Olvidar guardar el resultado del método

```
nombre = "  carlos  "
nombre.strip()   # strip se ejecuta pero el resultado se descarta

print(nombre)    # "  carlos  " — sin cambio
```

Corrección:

```
nombre = nombre.strip()
print(nombre)    # "carlos"
```

Índice fuera de rango

```
texto = "Hola"
print(texto[4])   # ERROR: el índice máximo es 3
```

Convertir número a string para concatenar

```
edad = 22
print("Tengo " + edad + " años")   # ERROR: tipos distintos
```

Corrección:

```
print("Tengo " + str(edad) + " años")   # opción 1
print(f"Tengo {edad} años")             # opción 2 — más limpia
```