

Modificación de cadenas

Edison Achalma

Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Este abstract será actualizado una vez que se complete el contenido final del artículo.

Tabla de contenidos

Introduction	2
0.1 Motivación	2
0.2 Intuición	2
0.3 Modelo mental	2
0.4 Desarrollo	2
0.4.1 1. Cambiar mayúsculas/minúsculas	2
0.4.2 2. Eliminar espacios (whitespace)	2
0.4.3 3. Reemplazar texto	3
0.4.4 4. Dividir texto (split)	3
0.4.5 5. Unir texto (join) - el inverso de split	4
0.5 Ejemplos	4
0.6 Errores comunes	5
0.7 Buenas prácticas	5
0.8 Ejercicio	6
0.9 Síntesis	7
0.10 Qué sigue	7
1 Publicaciones Similares	8

Edison Achalma  <https://orcid.org/0000-0001-6996-3364>

El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; <https://credit.niso.org/>) de la siguiente manera: Edison Achalma: conceptualización, metodología, análisis formal, investigación, recursos, curación de datos, redacción, visualización, supervisión, administración del proyecto

La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga, Portal Independencia N° 57, Ayacucho, AYA 5001, Perú, Email: elmer.achalma.09@unsch.edu.pe

Modificación de cadenas

0.1 Motivación

Los strings son uno de los tipos de datos más manipulados en programación. A menudo necesitas limpiar texto (eliminar espacios), cambiar formato (mayúsculas/minúsculas), reemplazar contenido o dividir texto en partes. Python ofrece métodos integrados que hacen estas operaciones simples y eficientes sin necesidad de escribir código complejo.

0.2 Intuición

Imagina que tienes un texto en un procesador de palabras y quieres aplicar diferentes transformaciones: poner todo en mayúsculas, eliminar espacios extra, o buscar y reemplazar palabras. Los métodos de string en Python son como los botones de formato en tu editor de texto: hacen exactamente eso, pero desde tu código.

0.3 Modelo mental

- **Strings son inmutables:** estos métodos **no modifican** el string original, devuelven uno nuevo.
- **Métodos encadenables:** puedes aplicar múltiples métodos en secuencia.
- **La mayoría devuelven un nuevo string** o una lista (en el caso de `split()`).
- **Sintaxis:** `string.metodo()` donde `string` es tu variable o literal.

0.4 Desarrollo

0.4.1 1. Cambiar mayúsculas/minúsculas

```
texto = "Hola Mundo"

# A mayúsculas
print(texto.upper())    # "HOLA MUNDO"

# A minúsculas
print(texto.lower())    # "hola mundo"

# Capitalizar (primera letra mayúscula)
print(texto.capitalize()) # "Hola mundo"

# Título (cada palabra con mayúscula inicial)
print(texto.title())    # "Hola Mundo"

# Swap case (invierte mayúsculas/minúsculas)
print("Hola Mundo".swapcase()) # "hOLA mUNDO"
```

0.4.2 2. Eliminar espacios (*whitespace*)

```
texto = "  Hola Mundo  "

# Eliminar espacios al inicio y final
print(texto.strip())      # "Hola Mundo"

# Solo al inicio (left)
print(texto.lstrip())     # "Hola Mundo  "

# Solo al final (right)
print(texto.rstrip())     # "  Hola Mundo"

# También funciona con caracteres específicos
texto2 = "***Hola***"
print(texto2.strip("*"))  # "Hola"
```

0.4.3 3. Reemplazar texto

```
texto = "Hola Mundo"

# Reemplazar una subcadena
print(texto.replace("Mundo", "Python")) # "Hola Python"

# Reemplazar un carácter
print(texto.replace("o", "0")) # "H0la Mund0"

# Reemplazar múltiples veces (con límite)
texto2 = "uno dos tres cuatro"
print(texto2.replace(" ", "-", 2)) # "uno-dos-tres cuatro"
```

0.4.4 4. Dividir texto (split)

```
texto = "Hola Mundo Python"

# Dividir por espacios (por defecto)
print(texto.split()) # ['Hola', 'Mundo', 'Python']

# Dividir por un separador específico
csv = "manzana,pera,banana"
print(csv.split(",")) # ['manzana', 'pera', 'banana']

# Dividir con límite
print(csv.split(",", 1)) # ['manzana', 'pera,banana']

# Dividir por saltos de línea
multilinea = "Línea 1\nLínea 2\nLínea 3"
print(multilinea.split("\n")) # ['Línea 1', 'Línea 2', 'Línea 3']
```

0.4.5 5. Unir texto (join) - el inverso de split

```
# Unir una lista en un string
palabras = ["Hola", "Mundo", "Python"]
print(" ".join(palabras))      # "Hola Mundo Python"
print("-".join(palabras))      # "Hola-Mundo-Python"
print(" | ".join(palabras))    # "Hola | Mundo | Python"
```

0.5 Ejemplos**Mínimo — Métodos básicos:**

```
texto = "  python es genial  "
print(texto.strip().upper())    # "PYTHON ES GENIAL"
print(texto.title())           # "  Python Es Genial  "
```

Progresivo — Limpieza de datos:

```
# Datos sucios de entrada
nombre = "  ana  "
email = "ANA@GMAIL.COM"
archivo = "imagen.JPG"
comentario = "  ¡Excelente!  "

# Limpieza
nombre_limpio = nombre.strip().title() # "Ana"
email_limpio = email.lower()           # "ana@gmail.com"
archivo_limpio = archivo.lower()       # "imagen.jpg"
comentario_limpio = comentario.strip() # "¡Excelente!"

print(f"Nombre: {nombre_limpio}")
print(f"Email: {email_limpio}")
print(f"Archivo: {archivo_limpio}")
print(f"Comentario: {comentario_limpio}")
```

Real — Procesamiento de CSV y validación:

```
# Datos de una línea CSV
linea = "  Ana García, 28, ana@email.com, activo  "

# Limpiar y dividir
datos = linea.strip().split(",")
nombre = datos[0].strip().title()
edad = int(datos[1].strip())
email = datos[2].strip().lower()
estado = datos[3].strip().lower()

print(f"Nombre: {nombre}")
```

```

print(f"Edad: {edad}")
print(f"Email: {email}")
print(f"Estado: {estado}")

# Validación
if not email.endswith("@email.com"):
    print("Email no válido")

if estado == "activo":
    print("Usuario activo")

```

0.6 Errores comunes

Error	Síntoma	Solución
Esperar que el método modifique el original	El string original no cambia	Asigna el resultado: <code>texto = texto.upper()</code>
Encadenar métodos en orden incorrecto	<code>strip().upper()</code> vs <code>upper().strip()</code>	El orden importa: primero limpias, luego transformas
Confundir <code>split()</code> con <code>join()</code>	Uso incorrecto de separadores	<code>split()</code> → string a lista; <code>join()</code> → lista a string
Llamar método en literal	<code>"texto".upper()</code> es válido	Los métodos funcionan en literales también

0.7 Buenas prácticas

1. Encadena métodos para limpieza:

```

# Limpio y eficiente
texto_limpio = texto.strip().lower().replace(" ", "_")

# Más verbose
texto = texto.strip()
texto = texto.lower()
texto = texto.replace(" ", "_")

```

2. Usa `.strip()` para limpiar entrada de usuario:

```

nombre = input("Nombre: ").strip().title()
email = input("Email: ").strip().lower()

```

3. Verifica contenido antes de operar:

```

if nombre: # No está vacío
    nombre = nombre.strip()

```

4. Usa `split()` con límite para casos específicos:

```
# Primera parte antes del primer espacio
primero, resto = "Hola mundo Python".split(" ", 1)
print(primero)   # "Hola"
print(resto)     # "mundo Python"
```

5. Prefiere `join()` en lugar de `+` para unir muchos strings:

```
# Eficiente
resultado = "".join(lista_de_strings)

# Ineficiente (crea muchos objetos)
resultado = ""
for s in lista_de_strings:
    resultado += s
```

0.8 Ejercicio

Intermedio — Procesamiento de texto de archivo

Escribe un programa que procese el siguiente texto:

```
texto = """
nombre: Ana García
email: ANA@GMAIL.COM
ciudad: Madrid
profesión: Ingeniera de Software
años_exp: 8
"""
```

Requisitos:

1. Limpiar cada línea (eliminar espacios)
2. Separar cada línea en clave: valor usando :
3. Limpiar las claves (`strip`, `lower`)
4. Limpiar los valores (`strip`, formatear según el caso)
5. Mostrar los datos formateados

Pista: usa `split("\n")` para dividir el texto en líneas.

Solución (no mires hasta haberlo intentado):

```
texto = """
nombre: Ana García
email: ANA@GMAIL.COM
ciudad: Madrid
profesión: Ingeniera de Software
años_exp: 8
"""
```

```
"""

# Dividir en líneas y limpiar
lineas = texto.strip().split("\n")
datos = {}

for linea in lineas:
    if ":" in linea:
        # Dividir por el primer ":"
        clave, valor = linea.split(":", 1)

        # Limpiar
        clave = clave.strip().lower()
        valor = valor.strip()

        # Formatear según el tipo de dato
        if clave == "email":
            valor = valor.lower()
        elif clave == "nombre":
            valor = valor.title()
        elif clave == "ciudad":
            valor = valor.strip().title()
        elif clave == "años_exp":
            valor = int(valor) # Convertir a entero

        datos[clave] = valor

# Mostrar resultados
print("=== Datos Procesados ===")
for clave, valor in datos.items():
    print(f"{clave}: {valor}")

print(f"\nTipo de años_exp: {type(datos['años_exp'])}")
```

0.9 Síntesis

- `.upper()` / `.lower()`: cambio de mayúsculas/minúsculas.
- `.strip()` / `.lstrip()` / `.rstrip()`: eliminar espacios al inicio/final.
- `.replace()`: reemplaza texto por otro.
- `.split()`: divide un string en lista según separador.
- `.join()`: une una lista en string (inverso de `split`).
- **Inmutabilidad**: todos los métodos devuelven strings nuevos.
- **Encadenamiento**: puedes aplicar múltiples métodos secuencialmente.
- `.title()` / `.capitalize()`: formatos especiales de capitalización.

0.10 Qué sigue

- **Relacionado**: métodos de string avanzados, f-strings, expresiones regulares

- **Siguiente tema:** Concatenación y formateo de strings — aprende las mejores formas de combinar texto y variables.

1 Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1.  [Instalacion De Anaconda](#)
2.  [Configurar Entorno Virtual Python Anaconda](#)
3.  [01 Introducion A La Programacion Con Python](#)
4.  [02 Variables Expresiones Y Statements Con Python](#)
5.  [03 Objetos De Python](#)
6.  [04 Ejecucion Condicional Con Python](#)
7.  [05 Iteraciones Con Python](#)
8.  [06 Funciones Con Python](#)
9.  [07 Dataframes Con Python](#)
10.  [08 Prediccion Y Metrica De Performance Con Python](#)
11.  [09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12.  [10 Metodos De Machine Learning Para Regresion Con Python](#)
13.  [Visualizacion De Datos Con Python](#)

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!