

📌 CLASE 18 – Manipulación de Strings: `.split()`, `.strip()`, `.join()`, slicing

📘 **Tema oficial validado** 🎯 *Aprender a dividir, limpiar, unir y extraer partes de texto con técnicas fundamentales de programación en Python.*

🧠 ¿Por qué esta clase es clave en tu formación?

✓ Porque **todo programa real manipula texto**

- Formularios, inputs, logs, nombres de archivos, correos, etc.
- Procesamiento de datos (como CSV, JSON, HTML, APIs, scraping)
- Automatización de tareas administrativas

Si dominas las cadenas, puedes “entender y limpiar el lenguaje de las máquinas”.

📖 PARTE 1: `.split()` — DIVIDIR TEXTO

♦ Definición

El método `.split()` divide una cadena en partes más pequeñas (lista), usando un separador (espacio por defecto).

♦ Sintaxis

```
cadena.split(separador)
```

♦ Ejemplos

```
frase = "Hola Gabo, bienvenido al curso"
palabras = frase.split()
print(palabras)  # ['Hola', 'Gabo,', 'bienvenido', 'al', 'curso']
```

```
correo = "gabo@correo.com"
usuario = correo.split("@")[0]
print(usuario)  # 'gabo'
```

♦ Casos profesionales

- Leer líneas separadas por comas o puntos
- Dividir palabras clave

- Extraer partes de rutas, emails, etc.

PARTE 2: `.strip()` — LIMPIAR TEXTO

Definición

El método `.strip()` elimina **espacios u otros caracteres innecesarios** al inicio y al final de una cadena.

Sintaxis

```
cadena.strip()           # Elimina espacios
cadena.strip("abc")      # Elimina los caracteres indicados
```

Ejemplos

```
nombre = "  Gabriel Espinosa  "
print(nombre.strip()) # 'Gabriel Espinosa'
```

```
linea = "###Hola###"
print(linea.strip("#")) # 'Hola'
```

Casos profesionales

- Limpiar inputs de usuario
- Validar campos vacíos
- Preparar texto para ser guardado o exportado

PARTE 3: `.join()` — UNIR TEXTO

-Definición

`.join()` une elementos de una lista de strings en **una sola cadena**, usando un separador.

-Sintaxis

```
separador.join(lista)
```

-Ejemplos

```
palabras = ["Hola", "Gabo", "bienvenido"]
frase = " ".join(palabras)
print(frase) # 'Hola Gabo bienvenido'
```

```
csv = ",".join(["Juan", "23", "Cuba"])
print(csv) # 'Juan,23,Cuba'
```

-Casos profesionales

- Crear archivos tipo CSV
- Unir textos para generar logs, reportes
- Construir frases dinámicas o plantillas

PARTE 4: **Slicing** — EXTRAER POR POSICIÓN

_Definición

Slicing permite **extraer partes de una cadena** indicando su posición.

_Sintaxis

```
cadena[inicio:fin:paso]
```

- **inicio**: índice inicial (incluido)
- **fin**: índice final (no incluido)
- **paso**: (opcional) saltos

_Ejemplos

```
texto = "Python es poderoso"
print(texto[0:6]) # 'Python'
print(texto[11:]) # 'poderoso'
print(texto[-9:]) # también 'poderoso'
```

```
palabra = "Gabriel"
print(palabra[::-1]) # 'leirbaG' (texto invertido)
```

_Casos profesionales

- Extraer nombres, fechas, códigos de producto

- Manipular estructuras semiestructuradas (como logs)
- Validar formatos

Resumen de cada método

Método	¿Qué hace?	Retorna
<code>.split()</code>	Divide string en partes	Lista
<code>.strip()</code>	Elimina caracteres innecesarios	String limpio
<code>.join()</code>	Une una lista de strings	String resultante
slicing	Extrae parte del string por posición	Substring


Aplicación real en la vida profesional

Este conocimiento se usa en:

- Automatización de formularios y validaciones
 - Conversión de datos de texto a estructuras
 - Web scraping, análisis de archivos, APIs
 - Preprocesamiento de datos en Machine Learning
 - Bots, dashboards, paneles de monitoreo
-

¡Perfecto, Gabo! 🎯 Ahora sí comenzamos oficialmente con el primer ejercicio de esta clase.

Ejercicio 1 – Dividir una frase en palabras

 Archivo sugerido: `01_dividir_texto_split.py` 🎯 **Objetivo:** Usar el método `.split()` para convertir una cadena completa en una lista de palabras separadas.

Teoría puntual antes del ejercicio

- ♦ ¿Qué hace `.split()`?

Divide un string en partes más pequeñas (por defecto, separa por espacios) y devuelve una **lista**.

```
frase = "Hola Gabo, bienvenido al curso"
palabras = frase.split()
# Resultado: ['Hola', 'Gabo,', 'bienvenido', 'al', 'curso']
```

 Puedes usar separadores personalizados:

```
texto = "nombre,edad,ciudad"  
print(texto.split(",")) # ['nombre', 'edad', 'ciudad']
```

▼ DIAGRAMA DE FLUJO PROFESIONAL

Formato secuencial textual (como tú lo pediste):

```
Inicio  
↓  
Solicitar al usuario que escriba una frase  
↓  
Guardar la frase como cadena de texto  
↓  
Aplicar método .split() para dividir la frase en palabras (usando espacio  
como separador)  
↓  
Guardar el resultado en una lista  
↓  
Mostrar la lista completa en consola  
↓  
Finalizar programa
```

📄 ESQUELETO GUÍA CON PISTAS

```
#❶ Solicitar al usuario que escriba una frase  
# 💡 Usa input() para capturar una oración completa  
  
#❷ Aplicar .split() para dividir la frase en palabras  
# 💡 Al no pasar argumento a .split(), divide por espacio  
automáticamente  
  
#❸ Guardar el resultado en una lista  
# 💡 Almacena la lista en una variable llamada 'palabras' por claridad  
  
#❹ Mostrar la lista completa en consola  
# 💡 Usa print() para mostrar la estructura tipo ['Hola', 'Gabo', ...]  
  
#❺ (Opcional) Mostrar cuántas palabras se detectaron  
# 💡 Usa len(lista) para mostrar la cantidad total de palabras extraídas
```

EJERCICIO 1 RESUELTO

```

from colorama import init, Fore, Style
init(autoreset=True)

def procesar_frase():
    try:
        frase = input(Fore.GREEN + "📄 Por favor escribe una frase
").strip().capitalize()
        if not frase:
            raise ValueError("La frase no puede estar vacía")

        palabras = frase.split()
        print(Fore.CYAN + f"📋 Lista de palabras: {palabras}")
        print(Fore.CYAN + f"📊 total de palabras: {len(palabras)}")

        palabra_mas_larga = max(palabras, key=len)
        print(Fore.YELLOW + f"👤 Palabra mas larga: [{palabra_mas_larga}]
con {len(palabra_mas_larga)} letsas")

        frecuencias = {palabra: palabras.count(palabra) for palabra in
set(palabras)}
        print(Fore.MAGENTA + f"Frecuencia de palabras: {frecuencias}")

    except ValueError as e:
        print(Fore.RED + f"❌ Error: {e}")
    except Exception as e:
        print(Fore.RED + f"⚠️ Error inesperado: {e}")

procesar_frase()

```

```

"""
RESPUESTA DE LA CONSOLA
📄 Por favor escribe una frase Las cicatrices son recordatorios de que
fuimos más fuertes que nuestras heridas.
📋 Lista de palabras: ['Las', 'cicatrices', 'son', 'recordatorios', 'de',
'que', 'fuimos', 'más', 'fuertes', 'que',
'nuestras', 'heridas.']
📊 total de palabras: 12
👤 Palabra mas larga: [recordatorios] con 13 letras
Frecuencia de palabras: {'fuertes': 1, 'de': 1, 'Las': 1, 'más': 1,
'recordatorios': 1, 'fuimos': 1, 'nuestras': 1, 'que':
2, 'cicatrices': 1, 'heridas.': 1, 'son': 1}
"""

```

¡Perfecto, Gabo! Pasamos oficialmente al:

■ Ejercicio 2 – Limpieza de nombres con `.strip()`

📄 Archivo sugerido: `02_limpieza_entrada_strip.py` 🎯 **Objetivo:** Usar `.strip()` para **limpiar entradas de texto** y validar que los nombres ingresados sean correctos y profesionales.

📖 TEORÍA EJERCICIO 2

♦ ¿Qué es `.strip()`?

Es un **método de string** que elimina automáticamente los **espacios en blanco** (o cualquier otro carácter no deseado) **al principio y al final** de una cadena de texto.

♦ Sintaxis general

```
cadena.strip()           # Elimina espacios por defecto
cadena.strip("x")        # Elimina caracteres específicos (como "x", "#",
etc.)
```

♦ Ejemplo básico

```
nombre = "  Gabriel  "
nombre_limpio = nombre.strip()
print(nombre_limpio) # Resultado: 'Gabriel'
```

🧠 ¿Qué NO elimina?

- `.strip()` **no borra espacios internos**, solo en los extremos.

```
nombre = "  Gabriel Espinosa  "
print(nombre.strip()) # Resultado: 'Gabriel Espinosa'
```

🏢 Aplicación profesional

`.strip()` es esencial en:

- Validación de formularios
- Limpieza de inputs antes de guardarlos en bases de datos
- Eliminación de caracteres basura en archivos

- Preparación de texto para procesamiento (CSV, XML, etc.)

DIAGRAMA DE FLUJO PROFESIONAL

```
Inicio
↓
Solicitar al usuario que ingrese su nombre completo
↓
Aplicar strip() para eliminar espacios al principio y al final
↓
Verificar si el nombre está vacío tras aplicar strip()
├── Sí → Mostrar error y terminar el programa
└── No → Mostrar el nombre limpio con confirmación
↓
Finalizar el programa
```

ESQUELETO GUÍA CON PISTAS

```
#❶ Solicitar al usuario que ingrese su nombre
# 💡 Usa input() para capturar un nombre con posibles espacios
innecesarios

#❷ Aplicar .strip() para limpiar la entrada
# 💡 Guarda el resultado en una nueva variable llamada nombre_limpio

#❸ Verificar si el nombre está vacío tras limpiar
# 💡 Usa if nombre_limpio == "": para detectar que no ingresó nada útil

#❹ Si está vacío, mostrar mensaje de error y terminar
# 💡 Puedes usar print() con mensaje profesional y opcionalmente exit()

#❺ Si no está vacío, mostrar el nombre limpio con un mensaje de éxito
# 💡 Usa print() para confirmar la limpieza del nombre con buena
presentación

#❻ (Opcional) Mostrar cuántos caracteres tenía antes y cuántos después
# 💡 Usa len(nombre_original) y len(nombre_limpio)
```

Ejemplo profesional de uso

👉 Entrada del usuario:

```
"    Ana María Pérez    "
```


🔗 Resultado tras `.strip()`:

"Ana María Pérez"

📝 Puede ser ahora almacenado en una base de datos, impreso en una credencial, etc.

EJERCICIO 2 RESUELTO

```
#=====
=====
#! Ejercicio 2 – Limpieza de nombres con .strip()
# Archivo sugerido: 02_limpieza_entrada_strip.py
# 🎯 Objetivo: Usar .strip() para limpiar entradas de texto y validar que
los nombres ingresados sean correctos y profesionales.

#===ESTILOS===
=====

from colorama import init, Fore, Style
init(autoreset=True)

#===SCRIPT=====
=====

while True:
    nombre_completo = input(Style.BRIGHT + Fore.CYAN + " 👤 Por favor
ingrese su nombre completo (solo letras y espacios): ")
    nombre_limpio = nombre_completo.strip()

    if not nombre_limpio:
        print(Fore.RED + " ❌ El nombre no puede estar vacío. inténtelo
nuevamente")
    elif not all(palabra.isalpha() for palabra in nombre_limpio.split()):
        print(Fore.YELLOW + " ❌ El nombre solo puede contener letras.
inténtelo nuevamente")
    else:
        nombre_formateado = ' '.join(palabra.capitalize() for palabra in
nombre_limpio.split())
        print(Fore.MAGENTA + "Nombre ingresado correctamente")
        print(Fore.GREEN + f" 🙌 {nombre_formateado} sea usted Bienvenido")
        break

#===RESPUESTA DE LA
CONSOLA=====
=====
"""
👤 Por favor ingrese su nombre completo (solo letras y espacios): Gabo9807
❌ El nombre solo puede contener letras. inténtelo nuevamente
👤 Por favor ingrese su nombre completo (solo letras y espacios):
❌ El nombre no puede estar vacío. inténtelo nuevamente
👤 Por favor ingrese su nombre completo (solo letras y espacios):
GABRIEL ESPINOSA izada
```

```
Nombre ingresado correctamente
👏 Gabriel Espinosa Izada sea usted Bienvenido
"""
#=====
=====
```

📌 .Ejercicio 3 – Unir datos con .join()

📄 Archivo sugerido: 03_unir_datos_join.py 🎯 **Objetivo:** Usar el método `.join()` para **combinar múltiples piezas de información** en una sola cadena de texto estructurada, estilo CSV o tabla profesional.

📘 TEORÍA PUNTUAL – .join()

♦ ¿Qué es .join()?

Es un método de string que permite **unir elementos de una lista** (u otra secuencia iterable) **en una sola cadena**, usando un separador específico.

Sintaxis general

```
separador.join(lista)
```

🔧 Donde:

- **separador**: string que deseas usar para unir los elementos (puede ser una coma `,`, espacio `" "`, tabulación `\t`, etc.)
 - **lista**: secuencia de strings que deseas unir
-

Ejemplo básico

```
datos = ["Gabriel", "Espinosa", "Cuba"]
resultado = ", ".join(datos)
print(resultado)
# Salida: Gabriel, Espinosa, Cuba
```

📌 Reglas importantes

1. **Todos los elementos deben ser strings.** Si hay enteros o floats, primero conviértelos con `str()`.
 2. `.join()` **no modifica la lista**, devuelve una **nueva cadena combinada**.
-

🧠 Aplicaciones profesionales

- Crear líneas CSV (**Juan,25,Cuba**)
- Preparar texto para exportar a archivos
- Generar logs o mensajes unificados
- Armar sentencias SQL o comandos con parámetros
- Combinar múltiples respuestas del usuario

.DIAGRAMA DE FLUJO PROFESIONAL

```
Inicio
↓
Solicitar al usuario que ingrese nombre, edad y país por separado
↓
Almacenar cada entrada en una variable (nombre, edad, país)
↓
Convertir todos los datos a string (si es necesario)
↓
Crear una lista con los tres elementos
↓
Aplicar .join() usando coma como separador
↓
Guardar el resultado en una nueva variable
↓
Mostrar la línea CSV final en consola
↓
Finalizar
```

.ESQUELETO GUÍA CON PISTAS

```
#1 Solicitar al usuario que ingrese su nombre
# 💡 Usa input() y .strip() para limpiar espacios

#2 Solicitar edad y convertirla a string (si es un número)
# 💡 Usa int() para validación y luego str() para unir

#3 Solicitar país y limpiarlo
# 💡 Usa input() y .strip()

#4 Crear una lista con los tres elementos
# 💡 Ejemplo: datos = [nombre, edad, país]

#5 Aplicar .join() con coma como separador
# 💡 Usa: resultado = ",".join(datos)

#6 Mostrar el resultado final en consola
# 💡 Ejemplo de salida: 'Gabriel,25,Cuba'
```

EJERCICIO 3 RESUELTO

```
#=====
#
#!#!📄 Archivo sugerido: 03_unir_datos_join.py
#!🎯 Objetivo: Usar el método .join() para combinar múltiples piezas de
información en una sola cadena de texto estructurada,
#! estilo CSV o tabla profesional.

#====ESTILOS=====
#
from colorama import init, Fore, Style
init(autoreset=True)

#====SCRIPT=====
#
while True:
    nombre_completo = input(Style.BRIGHT + Fore.CYAN + "👤 Por favor
ingrese su nombre completo (solo letras y espacios): ")
    nombre_limpio = nombre_completo.strip()

    if not nombre_limpio:
        print(Fore.RED + "🚫 El nombre no puede estar vacío. Inténtelo
nuevamente")
    elif not all(palabra.isalpha() for palabra in nombre_limpio.split()):
        print(Fore.YELLOW + "🚫 El nombre solo puede contener letras.
Inténtelo nuevamente")
    else:
        nombre_formateado = ' '.join(palabra.capitalize() for palabra in
nombre_limpio.split())
        print(Fore.MAGENTA + "Nombre ingresado correctamente")
        print(Fore.GREEN + f"👋 {nombre_formateado} sea usted Bienvenido")
        break

# 2 Solicitar edad, validarla y convertirla a string
while True:
    try:
        edad_input = input(Style.BRIGHT + Fore.CYAN + "🎂 Ingrese su edad
(número entero positivo): ").strip()
        edad = int(edad_input)
        if edad <= 0:
            print(Fore.YELLOW + "🚫 La edad debe ser un número positivo.
Inténtelo nuevamente")
            continue
        edad = str(edad)
        print(Fore.MAGENTA + "✅ Edad ingresada correctamente")
        break
    except ValueError:
        print(Fore.RED + "🚫 Error: La edad debe ser un número entero.
Inténtelo nuevamente")
    except KeyboardInterrupt:
        print(Fore.RED + "\n🛑 Programa interrumpido por el usuario.")
```

```

        exit()

# ③ Solicitar país y limpiarlo
while True:
    pais = input(Style.BRIGHT + Fore.CYAN + "🌐 Ingrese su país (solo
letras y espacios): ").strip()

    if not pais:
        print(Fore.RED + "❌ El país no puede estar vacío. Inténtelo
nuevamente")
    elif not all(palabra.isalpha() for palabra in pais.split()):
        print(Fore.YELLOW + "❌ El país solo puede contener letras.
Inténtelo nuevamente")
    else:
        pais_formateado = ' '.join(palabra.capitalize() for palabra in
pais.split())
        print(Fore.MAGENTA + "País ingresado correctamente")
        break

# ④ Crear una lista con los tres elementos
# Usamos nombre_formateado para mantener el formato con mayúsculas
datos = [nombre_formateado, edad, pais_formateado]

# ⑤ Aplicar .join() con coma como separador
resultado = ",".join(datos)

# ⑥ Mostrar el resultado final en consola
print(Style.BRIGHT + Fore.BLUE + "💡 Resultado final: " + resultado)

#====**RESPUESTA DE LA
CONSOLA**=====
=====
"""
Por favor ingrese su nombre completo (solo letras y espacios): GAB0123
❌ El nombre solo puede contener letras. Inténtelo nuevamente
👤 Por favor ingrese su nombre completo (solo letras y espacios):
❌ El nombre no puede estar vacío. Inténtelo nuevamente
👤 Por favor ingrese su nombre completo (solo letras y espacios):
GABRIEL ESPINOSA izada
Nombre ingresado correctamente
👏 Gabriel Espinosa Izada sea usted Bienvenido
🎂 Ingrese su edad (número entero positivo): edad = 27a
❌ Error: La edad debe ser un número entero. Inténtelo nuevamente
🎂 Ingrese su edad (número entero positivo): -9
❌ La edad debe ser un número positivo. Inténtelo nuevamente
🎂 Ingrese su edad (número entero positivo): 27
✅ Edad ingresada correctamente
🌐 Ingrese su país (solo letras y espacios): 123
❌ El país solo puede contener letras. Inténtelo nuevamente
🌐 Ingrese su país (solo letras y espacios): 🇨🇺Cuba
❌ El país solo puede contener letras. Inténtelo nuevamente
🌐 Ingrese su país (solo letras y espacios): Cuba
País ingresado correctamente
💡 Resultado final: Gabriel Espinosa Izada,27,Cuba

```

```
"""
#=====
=====
```

■ Ejercicio 4 – Extraer con slicing

📄 Archivo sugerido: `04_extraer_con_slicing.py` 🎯 **Objetivo:** Usar la sintaxis de **slicing** (`[inicio:fin:paso]`) para extraer partes específicas de un string (como nombre, código, extensión, etc.).

📘 TEORÍA PROFUNDA – SLICING

♦ ¿Qué es slicing?

Es una técnica que te permite **extraer una subparte de un string** (o de otras secuencias) **usando índices de posición**.

📌 Funciona sobre strings, listas, tuplas y otros iterables.

_Sintaxis general

```
cadena[inicio:fin:paso]
```

Elemento	Significado
inicio	Posición inicial (incluida)
fin	Posición final (excluida)
paso	Cuántos caracteres saltar (opcional)

♦ Ejemplos rápidos

```
texto = "GabrielEspinosa"

print(texto[0:7])    # 'Gabriel'
print(texto[7:])     # 'Espinosa'
print(texto[-3:])    # 'osa'
print(texto[::-1])   # 'asonosilEp leirbaG'
```

🧠 Casos profesionales

- Extraer nombre de archivo sin extensión
- Obtener las 4 primeras letras de un código
- Invertir texto
- Validar estructura de datos formateados (como fechas, claves, logs)

▼ DIAGRAMA DE FLUJO

```
Inicio
↓
Solicitar al usuario que escriba una cadena de texto
↓
Aplicar diferentes operaciones de slicing sobre la cadena:
|— Extraer los primeros N caracteres
|— Extraer los últimos M caracteres
|— Invertir el texto
|— Extraer desde el medio
↓
Mostrar los resultados uno por uno
↓
Finalizar
```

_ESQUELETO GUÍA CON PISTAS

```
#❶ Solicitar al usuario que escriba una cadena de texto
# 💡 Usa input() y .strip() para limpiar los bordes

#❷ Extraer los primeros 4 caracteres
# 💡 Usa slicing: cadena[0:4]

#❸ Extraer los últimos 3 caracteres
# 💡 Usa cadena[-3:] para tomar desde atrás

#❹ Extraer una subcadena desde la posición 3 hasta la 8
# 💡 Usa cadena[3:9] (el fin es excluyente)

#❺ Invertir toda la cadena
# 💡 Usa cadena[::-1]

#❻ Mostrar cada uno de los resultados con etiquetas claras
# 💡 Usa print() bien organizado, con color si deseas

#❼ (Opcional avanzado) Mostrar la longitud original con len()
```

EJERCICIO 4 RESUELTO

```
#=====
=====
"""
■ Ejercicio 4  Extraer con slicing
📄 Archivo sugerido: 04_extraer_con_slicing.py
🎯 Objetivo: Usar la sintaxis de slicing ([inicio:fin:paso]) para extraer
partes específicas de un string (como nombre, código, extensión, etc.).
"""

#====ESTILOS=====
=====
from colorama import init, Fore, Style
init(autoreset=True)

#====SCRIPT=====
=====
texto = input(Fore.CYAN + "📄 Por favor, ingresa una cadena de texto: " +
Style.RESET_ALL).strip()

primeros_cuatro = texto[0:4]
ultimos_tres = texto[-3:]
subcadena = texto[3:9]
invertida = texto[::-1]

print(f"\n{Fore.GREEN}Resultados:{Style.RESET_ALL}")
print(f"{Fore.YELLOW}Cadena original:{Style.RESET_ALL} {texto}")
print(f"{Fore.YELLOW}Primeros 4 caracteres:{Style.RESET_ALL}
{primeros_cuatro}")
print(f"{Fore.YELLOW}Últimos 3 caracteres:{Style.RESET_ALL}
{ultimos_tres}")
print(f"{Fore.YELLOW}Subcadena (posiciones 3 a 8):{Style.RESET_ALL}
{subcadena}")
print(f"{Fore.YELLOW}Cadena invertida:{Style.RESET_ALL} {invertida}")

print(f"{Fore.YELLOW}Longitud de la cadena:{Style.RESET_ALL}
{len(texto)}")

#====RESPUESTA DE LA
CONSOLA=====
=====
"""
📄 Por favor, ingresa una cadena de texto: La vida es como un espejo: si
sonríes, te devuelve la sonrisa

Resultados:
Cadena original: La vida es como un espejo: si sonríes, te devuelve la
sonrisa
Primeros 4 caracteres: La v
Últimos 3 caracteres: isa
Subcadena (posiciones 3 a 8): vida e
Cadena invertida: asirnos al evleuved et ,seírnos is :ojepse nu omoc se
adiv al
```



```
Longitud de la cadena: 61
"""
#=====
=====
```

✓ Clase 18 – Completada oficialmente

Ejercicio	Tema	Nota
1	.split()	10
2	.strip()	10
3	.join()	11 ☆
4	Slicing ([inicio:fin:paso])	10

🏆 Clase completamente aprobada con excelencia 🎓 Promedio general: 10.25 / 10