

Salida de texto con print()

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. Sintaxis básica	2
0.4.2 2. Comillas: simples o dobles	2
0.4.3 3. Múltiples print(): cada uno en nueva línea	3
0.4.4 4. Múltiples argumentos en un print()	3
0.4.5 5. Personalizar el final: end	3
0.4.6 6. Personalizar el separador: sep	3
0.5 Ejemplos	3
0.6 Errores comunes	4
0.7 Buenas prácticas	4
0.8 Ejercicio	5
0.9 Síntesis	5
0.10 Qué sigue	6
1 Publicaciones Similares	6

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

Salida de texto con print()

0.1 Motivación

`print()` es tu ventana al mundo: es cómo ves lo que tu programa está haciendo. Sin `print()`, estarías programando a ciegas, sin saber si tus variables tienen los valores correctos o si tu lógica funciona. Dominar `print()` te permite ver, depurar y comunicar resultados de manera efectiva.

0.2 Intuición

Imagina que `print()` es como un altavoz: le das un mensaje y lo reproduce en voz alta para que todos lo escuchen. Puedes usarlo tantas veces como quieras, y cada vez que lo usas, empieza una “nueva línea” automáticamente, como si presionaras Enter después de cada mensaje.

0.3 Modelo mental

La función `print()` funciona así:

1. **Acepta uno o más argumentos:** puede recibir texto, números, variables, o combinaciones.
2. **Convierte todo a texto:** si le pasas un número, lo convierte a texto antes de mostrarlo.
3. **Añade un salto de línea al final:** por defecto, después de imprimir pasa a la siguiente línea.
4. **Puede personalizarse:** con parámetros como `end` y `sep` cambias su comportamiento.

0.4 Desarrollo

0.4.1 1. Sintaxis básica

```
print("texto")
print(variable)
print("texto", variable)
```

0.4.2 2. Comillas: simples o dobles

Python acepta ambos tipos de comillas. La única regla: **abre y cierra con el mismo tipo**.

```
print("Hola")      # Válido
print('Hola')      # Válido
print("Hola'")     # Error: mezcla de comillas
```

¿Cuándo usar cada una?

- Si tu texto contiene comillas simples, usa dobles: `print("It's awesome")`
- Si contiene comillas dobles, usa simples: `print('Ella dijo "Hola"')`
- O usa `\` para escapar: `print("It\'s awesome")`

0.4.3 3. Múltiples print(): cada uno en nueva línea

Por defecto, cada print() comienza en una línea nueva:

```
print("Línea 1")
print("Línea 2")
print("Línea 3")
# Salida:
# Línea 1
# Línea 2
# Línea 3
```

0.4.4 4. Múltiples argumentos en un print()

Puedes pasar varios elementos separados por comas:

```
nombre = "Ana"
edad = 25
print("Nombre:", nombre, "Edad:", edad)
# Salida: Nombre: Ana Edad: 25
```

0.4.5 5. Personalizar el final: end

El parámetro end controla qué se añade al final (por defecto es \n, salto de línea):

```
print("Hola", end=" ")
print("Mundo")
# Salida: Hola Mundo

print("Manzana", end=", ")
print("Pera", end=", ")
print("Banana")
# Salida: Manzana, Pera, Banana
```

0.4.6 6. Personalizar el separador: sep

El parámetro sep controla qué se pone entre múltiples argumentos (por defecto es espacio):

```
print("uno", "dos", "tres", sep="-")
# Salida: uno-dos-tres

print(1, 2, 3, sep=" + ")
# Salida: 1 + 2 + 3
```

0.5 Ejemplos

Mínimo — Texto simple:

```
print("Hola Mundo")
# Salida: Hola Mundo
```

Progresivo — Variaciones de comillas:

```
print('Python es "genial"')
print("It's a beautiful day")
print("Ella dijo: \"Hola\"") # Escapando comillas
# Salida:
# Python es "genial"
# It's a beautiful day
# Ella dijo: "Hola"
```

Real — Menú de opciones con formato:

```
print("=== MENÚ PRINCIPAL ===")
print("1. Ver perfil")
print("2. Editar configuración", end=" ")
print("(requiere permisos)")
print("3. Salir")
print("=====")
# Salida:
# === MENÚ PRINCIPAL ===
# 1. Ver perfil
# 2. Editar configuración (requiere permisos)
# 3. Salir
# =====
```

0.6 Errores comunes

Error	Síntoma	Cómo detectarlo
Olvidar comillas	SyntaxError: invalid syntax	El texto siempre va entre " " o ' '
Comillas desbalanceadas	SyntaxError: unterminated string literal	Cuenta que abres y cierras el mismo número de comillas
Confundir print con Print	NameError: name 'Print' is not defined	Python es sensible a mayúsculas: usa print en minúsculas
Olvidar paréntesis	SyntaxError: Missing parentheses in call to 'print'	Python 3 requiere ()
Espacio extra al final	No da error, pero puede afectar formato	Usa end="" sin espacio si no quieres separador

0.7 Buenas prácticas

- Usa f-strings para combinar texto y variables (lo veremos pronto):

```
nombre = "Ana"
print(f"Hola, {nombre}") # Más legible que concatenar
```

- **Comillas simples por defecto**, usa dobles solo cuando el texto contenga comillas simples.
- **end=" "** con espacio para mejor legibilidad al unir líneas.
- **No concatenes con +** innecesariamente: usa múltiples argumentos en `print()` con comas.
- **Mensajes descriptivos**: en lugar de `print(x)`, usa `print("El resultado es:", x)`.

0.8 Ejercicio

Básico — Formato de salida

Escribe un programa que:

1. Imprima “— BIENVENIDO —” en una línea
2. Imprima “Usuario:” y luego el nombre “Carlos” en la misma línea, separados por un espacio
3. Imprima “Edad:” y luego “30” en la misma línea
4. Imprima “Estado: Activo” con un guión bajo (`_`) entre las palabras
5. Imprima “— FIN —” con un punto final (`.`) al final, sin salto de línea

Pista: usa los parámetros `sep`, `end` y múltiples argumentos.

Solución (no mires hasta haberlo intentado):

```
print("--- BIENVENIDO ---")
print("Usuario:", "Carlos", sep=" ")
print("Edad:", 30, sep=" ")
print("Estado", "Activo", sep="_")
print("--- FIN ---", end=".")
```

Salida esperada:

```
--- BIENVENIDO ---
Usuario: Carlos
Edad: 30
Estado_Activo
--- FIN ---.
```

0.9 Síntesis

- `print()` muestra texto y valores en la consola.
- **Comillas simples o dobles** son válidas, pero deben coincidir.
- **Cada `print()`** añade un salto de línea por defecto.
- **`end`** personaliza el final de línea (por defecto `\n`).
- **`sep`** personaliza el separador entre argumentos (por defecto espacio).
- **Múltiples argumentos** separados por comas se imprimen con espacio entre ellos.

0.10 Qué sigue

- **Relacionado:** variables (cómo almacenar datos para imprimir después), f-strings (formato avanzado)
- **Siguiente tema:** Variables y tipos de datos — aprenderás a crear y manejar datos que luego podrás imprimir.

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!