

Salida de variables

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. Imprimir una variable	2
0.4.2 2. Múltiples variables con comas	2
0.4.3 3. Concatenación con + (solo texto)	3
0.4.4 4. + con números (operación matemática)	3
0.4.5 5. Error común: mezclar texto y números con + ...	3
0.4.6 6. Solución: usar comas para mezclar tipos	3
0.4.7 7. Personalizar el separador con sep	3
0.4.8 8. Comparativa de métodos	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	7

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 variables

0.1 Motivación

Una vez que tienes datos almacenados en variables, necesitas poder mostrarlos. Python te ofrece varias formas de combinar y mostrar variables con `print()`. Elegir el método correcto hace tu código más legible y evita errores comunes como mezclar tipos de datos incorrectamente.

0.2 Intuición

Imagina que tienes piezas de un rompecabezas (variables) y quieres mostrarlas juntas. Puedes:

- **Colocarlas una al lado de la otra** (usando comas): `print(x, y)` → cada pieza se muestra con un espacio entre ellas.
- **Pegarlas** (usando +): `print(x + y)` → las piezas se unen formando una sola pieza (solo funciona con texto).
- **No puedes pegar una pieza de madera con una de plástico** (string + número) → da error.

0.3 Modelo mental

El `print()` maneja variables de diferentes maneras:

1. **Múltiples argumentos (con comas)**: muestra cada valor separado por un espacio (por defecto).
2. **Concatenación con +**: une valores en un solo valor (solo funciona con el mismo tipo).
3. **Diferencias clave**: + une, la coma separa.
4. **Restricción**: no puedes usar + entre texto y número.

0.4 Desarrollo

0.4.1 1. Imprimir una variable

La forma más simple:

```
nombre = "Ana"
print(nombre)  # Ana

edad = 25
print(edad)    # 25
```

0.4.2 2. Múltiples variables con comas

Separa variables con comas para mostrarlas juntas:

```
nombre = "Ana"
apellido = "García"
edad = 25

print(nombre, apellido, edad)
# Salida: Ana García 25
```

Python añade un espacio entre cada valor automáticamente.

0.4.3 3. Concatenación con + (solo texto)

Une texto con + para formar un solo valor:

```
nombre = "Ana"
apellido = "García"

print(nombre + " " + apellido)
# Salida: Ana García
```

Importante: + solo funciona si todos los elementos son texto (strings).

0.4.4 4. + con números (operación matemática)

Cuando usas + con números, Python suma:

```
x = 5
y = 10
print(x + y) # 15 (suma matemática)
```

0.4.5 5. Error común: mezclar texto y números con +

```
x = 5
y = "John"
print(x + y) # TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

0.4.6 6. Solución: usar comas para mezclar tipos

```
x = 5
y = "John"
print(x, y) # 5 John (funciona con diferentes tipos)
```

0.4.7 7. Personalizar el separador con sep

Puedes cambiar el espacio entre elementos usando sep:

```
x = "Ana"
y = "García"
z = 25

print(x, y, z, sep=" - ")
# Salida: Ana - García - 25

print(x, y, z, sep="")
# Salida: AnaGarcía25

print(x, y, z, sep=", ")
# Salida: Ana, García, 25
```

0.4.8 8. Comparativa de métodos

Método	Ejemplo	Resultado	Cuándo usar
Coma	<code>print(x, y)</code>	Hola Mundo	Siempre que tengas diferentes tipos
+ (texto)	<code>print(x + y)</code>	HolaMundo	Cuando quieras unir texto sin espacios
+ (números)	<code>print(x + y)</code>	15	Para sumar números
sep	<code>print(x, y, sep="-")</code>	Hola- Mundo	Cuando quieras controlar el separador

0.5 Ejemplos

Mínimo — Variables simples:

```
nombre = "Carlos"
print(nombre) # Carlos
```

Progresivo — Múltiples variables:

```
# Con comas (recomendado para diferentes tipos)
nombre = "Ana"
edad = 25
ciudad = "Madrid"
print(nombre, edad, ciudad) # Ana 25 Madrid

# Con + (solo texto)
saludo = "Hola"
nombre = "Ana"
print(saludo + " " + nombre) # Hola Ana

# Con sep
print(nombre, edad, ciudad, sep=" | ") # Ana | 25 | Madrid
```

Real — Informe de ventas:

```
# Datos de venta
producto = "Laptop"
precio = 850.50
cantidad = 3
total = precio * cantidad

# Método recomendado (comas)
print("Producto:", producto)
print("Precio unitario:", precio, "€")
print("Cantidad:", cantidad)
print("Total:", total, "€")
```

```
# También puedes usar f-strings (lo veremos después)
print(f"Total: {total}€") # Más elegante
```

0.6 Errores comunes

Error	Síntoma	Solución
Texto + número	<code>TypeError: unsupported operand type(s)</code>	Usa comas en lugar de +
Olvidar espacio en +	<code>"Hola""Mundo" → HolaMundo</code>	Añade espacio explícito: <code>"Hola " + "Mundo"</code>
Confundir + para texto y números	<code>"10" + 5 → error</code>	Convierte: <code>int("10") + 5</code> o usa comas
Olvidar convertir números	<code>"Total: " + 100 → error</code>	Usa comas o convierte a texto: <code>str(100)</code>

0.7 Buenas prácticas

1. Usa comas cuando tengas diferentes tipos:

```
# Recomendado
print("Edad:", edad, "años")

# Propenso a error
print("Edad: " + str(edad) + " años")
```

2. Usa f-strings para formato avanzado (más elegante):

```
nombre = "Ana"
edad = 25
print(f"{nombre} tiene {edad} años") # Ana tiene 25 años
```

3. Para unir texto, prefiere comas o f-strings en lugar de +:

```
# Más legible
print(nombre, apellido)

# Más control
print(f"{nombre} {apellido}")

# Menos legible
print(nombre + " " + apellido)
```

4. Usa sep para separadores consistentes:

```
print(nombre, edad, ciudad, sep=", ")  
# Ana, 25, Madrid
```

0.8 Ejercicio

Básico — Mostrar variables

Escribe un programa que:

1. Cree variables para: nombre (texto), edad (número), altura (decimal), ciudad (texto)
2. Muestre los datos con comas (usando diferentes tipos)
3. Muestre un mensaje completo usando + (solo la parte de texto)
4. Muestre la edad que tendrá en 10 años (operación matemática)
5. Use sep para mostrar los datos separados por " | "

Solución (no mires hasta haberlo intentado):

```
# Datos  
nombre = "Carlos"  
edad = 28  
altura = 1.75  
ciudad = "Barcelona"  
  
# 1. Con comas (diferentes tipos)  
print("Datos personales:")  
print("Nombre:", nombre)  
print("Edad:", edad, "años")  
print("Altura:", altura, "metros")  
print("Ciudad:", ciudad)  
  
# 2. Con + (solo texto)  
print("Bienvenido " + nombre + " de " + ciudad)  
  
# 3. Operación con números  
edad_futura = edad + 10  
print("Edad en 10 años:", edad_futura)  
  
# 4. Usando sep  
print(nombre, edad, altura, ciudad, sep=" | ")
```

Salida esperada:

```
Datos personales:  
Nombre: Carlos  
Edad: 28 años  
Altura: 1.75 metros  
Ciudad: Barcelona  
Bienvenido Carlos de Barcelona  
Edad en 10 años: 38  
Carlos | 28 | 1.75 | Barcelona
```

0.9 Síntesis

- **Comas** (`print(x, y)`) → separa con espacio, funciona con cualquier tipo.
- **+** (**texto**) → une strings, solo funciona con strings.
- **+** (**números**) → suma matemática.
- **Error común:** `texto + número` → `TypeError`.
- **Solución:** usa comas o convierte con `str()`.
- **sep:** personaliza el separador entre elementos.
- **f-strings:** la forma más elegante (lo veremos más adelante).

0.10 Qué sigue

- **Relacionado:** f-strings (formato avanzado), strings y métodos de string
- **Siguiente tema:** Tipos de datos — aprenderás sobre strings, números, booleanos y más.

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. [!\[\]\(e7a5b2ecc7ab80b32b565dd7dfa9a5a9_img.jpg\) Instalacion De Anaconda](#)
2. [!\[\]\(51a3b3d739efe92b5a87bb7fdd8bc4bf_img.jpg\) Configurar Entorno Virtual Python Anaconda](#)
3. [!\[\]\(ebb8dcf4bf19cae7a44506695af49594_img.jpg\) 01 Introducion A La Programacion Con Python](#)
4. [!\[\]\(07dc25d18d466be82f39170ef767d9cd_img.jpg\) 02 Variables Expresiones Y Statements Con Python](#)
5. [!\[\]\(96664e733fc01177c2aa9b1fca87093c_img.jpg\) 03 Objetos De Python](#)
6. [!\[\]\(d36218996868ebf3f9a154f16e8f90c9_img.jpg\) 04 Ejecucion Condicional Con Python](#)
7. [!\[\]\(ff962ee77705b2bd42eba8e509d05e2a_img.jpg\) 05 Iteraciones Con Python](#)
8. [!\[\]\(34f6c5f8c474a3110d0b8de52a1097d0_img.jpg\) 06 Funciones Con Python](#)
9. [!\[\]\(35b21e08b342659af497e3c6c7a4690c_img.jpg\) 07 Dataframes Con Python](#)
10. [!\[\]\(e5c8eba8e46ee7176d9190b42d767b62_img.jpg\) 08 Prediccion Y Metrica De Performance Con Python](#)
11. [!\[\]\(5dc6a3716876fd06546d520876273c88_img.jpg\) 09 Metodos De Machine Learning Para Clasificacion Con Python](#)
12. [!\[\]\(53004109e56632bf639d38d4a35510dd_img.jpg\) 10 Metodos De Machine Learning Para Regresion Con Python](#)
13. [!\[\]\(7f541027eff0a02a1d79f764376a52f1_img.jpg\) Visualizacion De Datos Con Python](#)

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