Funciones

¿Qué son las funciones?

Una función es un **módulo de código que realiza una tarea específica**. Es como una pequeña máquina dentro de tu programa que toma datos de entrada, los procesa y produce un resultado.

Las funciones **encapsulan** una tarea específica. Esto significa que agrupan código relacionado en un bloque reutilizable que tiene un propósito claro y bien definido.

Pueden recibir datos (llamados parámetros), procesarlos según las instrucciones programadas, y devolver un resultado o realizar una acción específica.

¿Para qué sirven las funciones?

Reutilización de código

En lugar de escribir el mismo código una y otra vez, escribes una función una vez y la usas cuantas veces necesites. Esto ahorra tiempo y reduce errores.

División de problemas

Permiten dividir un problema complejo en partes más pequeñas y manejables. Cada función resuelve una parte específica del problema general.

Facilitan las pruebas

Puedes probar cada función individualmente para asegurar que funciona correctamente antes de integrarla en el programa completo.

Anatomía de una función

Toda función en programación tiene componentes esenciales que debemos conocer:

01

def (palabra clave)

Define que estamos creando una función

02

Nombre

Identificador único de la función

03

Parámetros

Datos de entrada (pueden ser ninguno o varios)

04

Cuerpo

Bloque de código que ejecuta la tarea

05

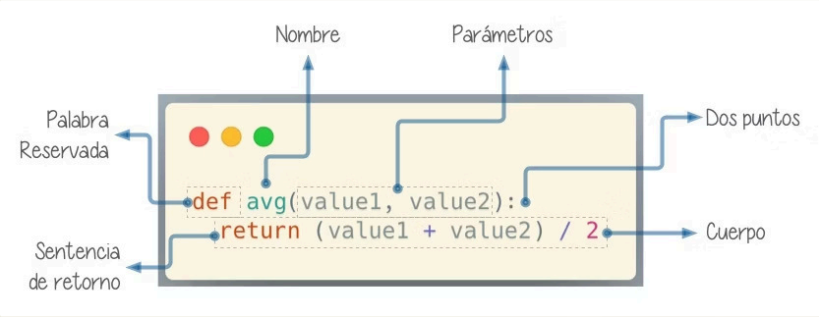
Valor de retorno

Resultado que devuelve (opcional)

06

Llamada/Invocación

Ejecución de la función desde el código



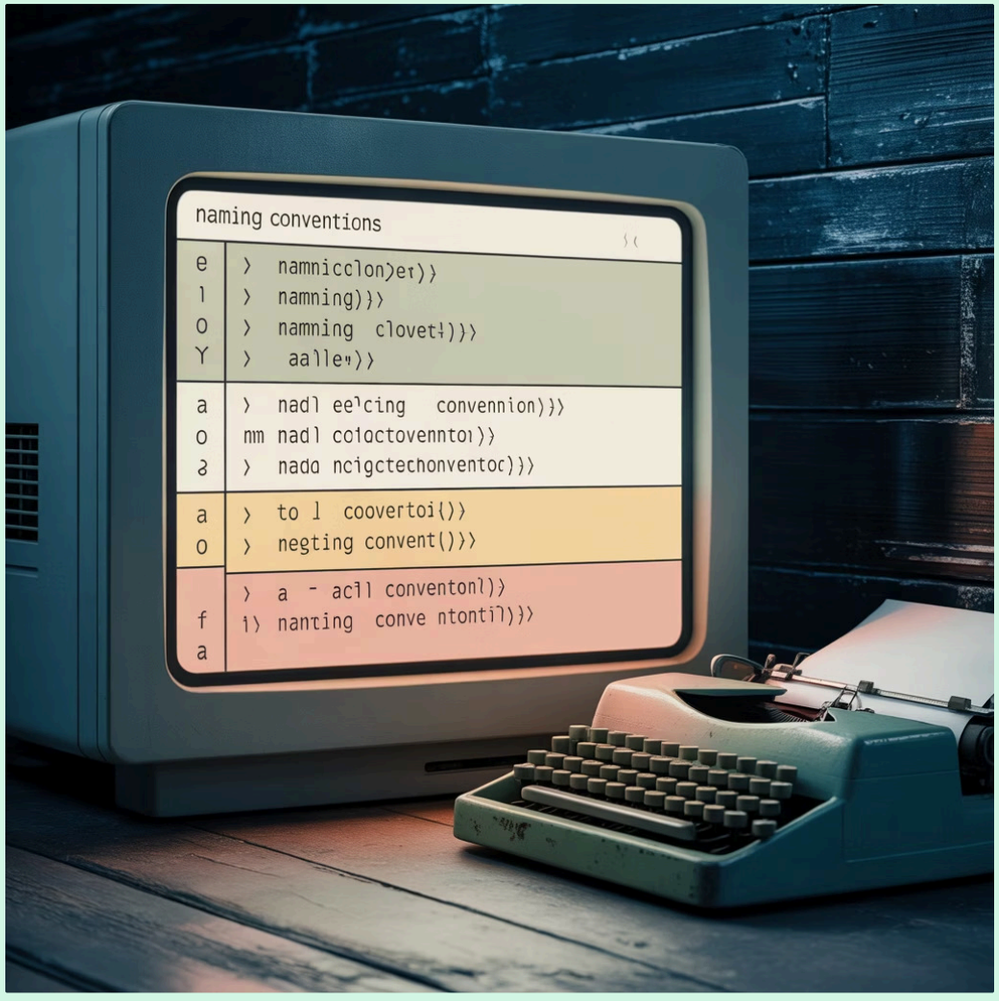
Estructura visual de una función

Esta imagen muestra la estructura típica de una función en Python. Observa cómo cada elemento tiene su lugar y propósito específico en la definición de la función.

Reglas para nombrar funciones

Reglas obligatorias:

- Solo puede contener letras, dígitos numéricos o guión bajo (_)
- No puede empezar con un número
- No puede ser igual a una palabra reservada del lenguaje



Buena práctica: El nombre debe ser **representativo** de lo que hace la función. Por ejemplo: calcular_promedio(), validar_email(), mostrar_menu()

Todo sobre los parámetros

¿Qué son?

Datos que se envían a la función para que los procese. Son como las **materias primas** que la función necesita para trabajar.

Características

- Pueden ser de cualquier tipo de dato
- Si hay varios, se separan con comas
- Dentro de la función actúan como variables locales

Valores por defecto

Puedes asignar valores predeterminados a los parámetros para que sean opcionales al llamar la función.

Importante: La cantidad de parámetros definidos debe coincidir con la cantidad de argumentos enviados al llamar la función.

El cuerpo de la función

El cuerpo es el **corazón de la función**, donde se encuentra toda la lógica que ejecutará:

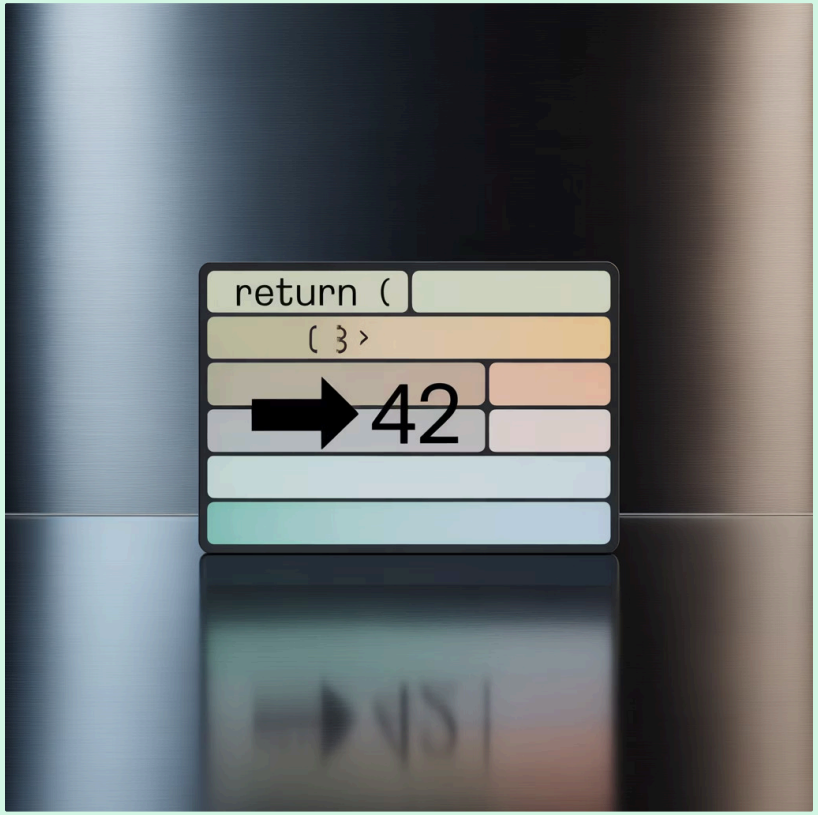
- Es un bloque de código como cualquier otro (puede incluir variables, bucles, decisiones, etc.)
- Si incluye variables locales, su ámbito se restringe solo a esa función
- Los parámetros se pueden utilizar como variables dentro del cuerpo
- Si cambias el valor de un parámetro, no se modifica la variable original fuera de la función

Valor de retorno: el resultado

El valor de retorno es el **resultado** de la operación que realiza la función.
Es lo que la función "devuelve" al código que la llamó.

Características importantes:

- Puede ser de cualquier tipo de dato
- Es completamente opcional
- Solo puede retornar **una cosa** por vez
- Donde aparece `return`, la función termina su ejecución



Llamar a una función

1

¿Desde dónde?

Desde cualquier parte del código, incluso desde dentro de otra función. Esto permite crear funciones que cooperan entre sí.

2

¿Cómo?

Escribiendo el nombre de la función seguido de paréntesis con los argumentos necesarios: `nombre_funcion(argumentos)`

3

¿Cuántas veces?

Las veces que necesites, con diferentes argumentos cada vez. Una función bien diseñada puede reutilizarse infinitamente.

4

El resultado

La llamada se **reemplaza** por el valor de retorno. Puedes asignarlo a una variable, usarlo en operaciones, imprimirlo, etc.

Dónde crear las funciones

En el mismo archivo

La forma más sencilla: crear la función dentro del mismo archivo donde está el resto del programa principal.

En archivos separados

Para proyectos más grandes, puedes crear las funciones en archivos diferentes y luego importarlas:

- `from archivo import *`
- `import archivo`

Esta organización te permite mantener el código más limpio y reutilizar funciones en múltiples programas.

Errores comunes que debes evitar

Usar variables globales

Evita usar variables del ámbito global dentro de las funciones. Esto hace el código impredecible y difícil de mantener.

Funciones que hacen demasiado

Una función debe realizar **una sola tarea**. Si hace varias cosas, divídela en funciones más pequeñas.

Argumentos incorrectos

Llamar a una función sin los argumentos necesarios genera un error. Siempre verifica que coincidan.

Ignorar el valor de retorno

Si una función devuelve algo útil, úsalo. No hacerlo es desperdiciar funcionalidad.

Ejercicio práctico

El reto:

Vamos a crear un programa que:

1. Solicite números al usuario hasta que ingrese cero
2. Para cada número, muestre la suma de sus dígitos
3. Al finalizar, muestre la sumatoria total y la suma de dígitos

Este ejercicio nos permitirá practicar la creación de funciones, el manejo de parámetros y valores de retorno de manera práctica.

Implementación del ejercicio

Para resolver este problema, necesitaremos crear varias funciones especializadas:

01

suma_digitos(numero)

Calcula y devuelve la suma de los dígitos de un número

02

solicitar_numero()

Pide un número al usuario y lo valida

03

programa_principal()

Coordina todo el flujo del programa usando las otras funciones

Cada función tiene una **responsabilidad específica**, lo que hace el código más fácil de entender, probar y mantener.

Reflexión final

Las funciones son uno de los **pilares fundamentales** de la programación. Dominarlas te permitirá:

- Escribir código más limpio y organizado
- Resolver problemas complejos de manera sistemática
- Reutilizar tu trabajo y ser más eficiente
- Colaborar mejor con otros programadores

Recuerda: Una función bien diseñada es como una herramienta perfecta: hace exactamente lo que necesitas, cuando lo necesitas, y es fácil de usar.

¡Ahora es tu turno de practicar y crear tus propias funciones!