Verificación de Programas Secuenciales



Conceptos, proceso y ejemplo práctico en Python

```
def verificar_programa(programa):
    estado = inicializar_estado()
    for instruccion in programa:
        estado = ejecutar(estado, instruccion)
        if not es_valido(estado):
            return False
    return True

def ejecutar(estado, instruccion):
    if instruccion.tipo == 'asignacion':
        estado[instruccion.var] = evaluar(i
    elif instruccion.tipo == 'salto':
        estado = actualizar_pc(estado, instru
    return estado

def es_valido(estado):
    return not hay_errores(estado)
```

Integrantes



JOSE ALEJANDRO ANGULO GONZáLEZ

AYLING DAYANA CABRERA GONZáLEZ



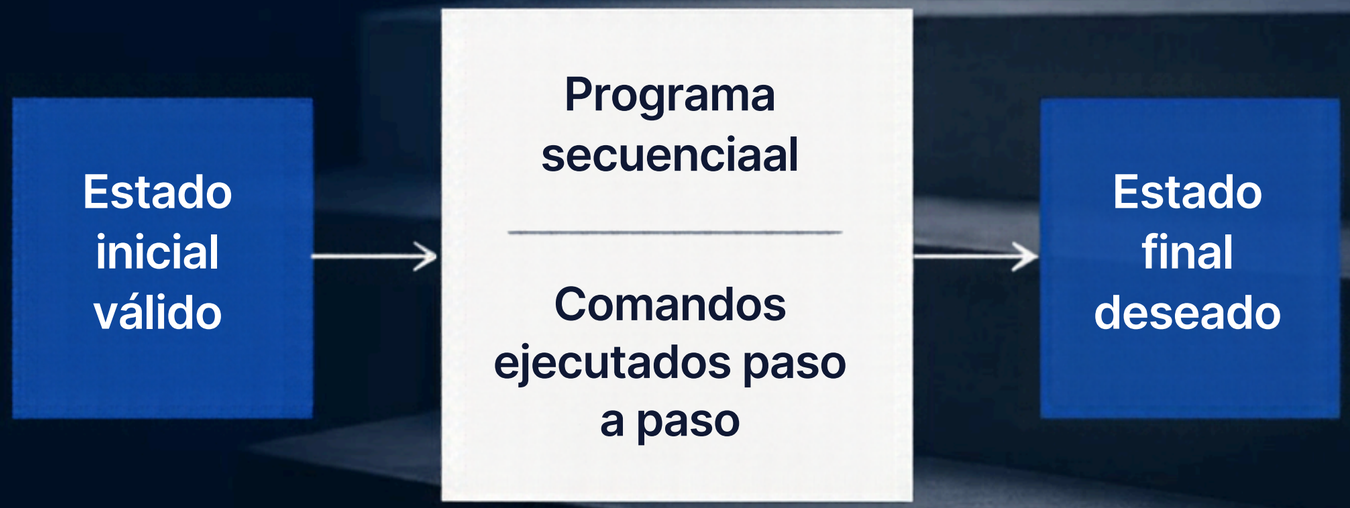
Definición y Propósito

La verificación formal demuestra matemáticamente que el programa cumple su especificación.

Un programa secuenciai transforma un estado inicial válido en un estado final deseado.

Cada comando ejecutado paso a paso preserva la corrección.

Aporta garantía lógica ante cualquier ejecución válida, más allá de las pruebas empíricas.



La Terna de Hoare

$$\{P\}\ C\ \{Q\}$$



Estado inicial requerido antes de ejecutar el comando

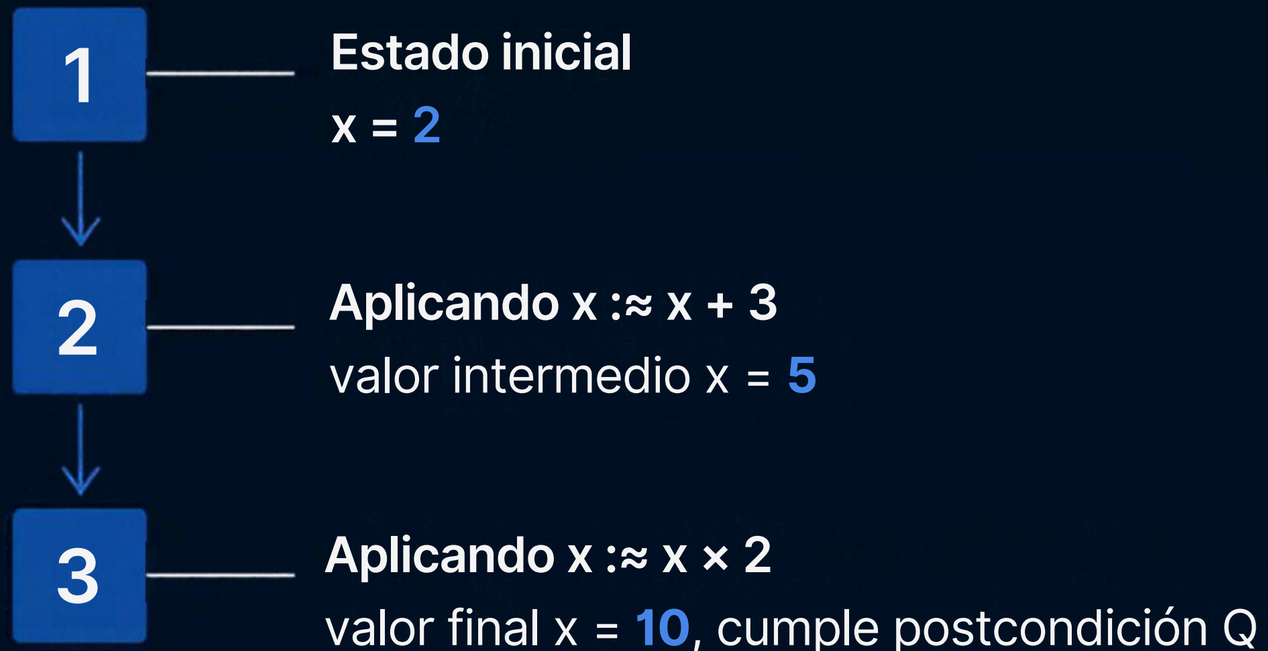
Instrucción o secuencia de instrucciones ejecutadas

Propiedad garantizada al finalizar si P era verdadera

Correctitud parcial: si **P** se cumple y el programa termina, Q es verdadera. La terminación se analiza por separado.

Trazado de Variables en una Secuencia

$\{x = 2\} \rightarrow x :\approx x + 3; x :\approx x \times 2 \rightarrow \{x = 10\}$



De la Teoría al Código

Aserciones en el código verifican estados intermedios y el resultado final.

Los contratos PRE y POST establecen responsabilidades claras entre quien llama y quien implementa.

Teoría formal aplicada directamente en el desarrollo real.

```
1  int suma_no_negativa(int a, int b) {
2      // PRE: las entradas deben ser no negativas
3      assert(a >= 0 && b >= 0);
4
5      int resultado = a + b;
6
7      // Invariante: el resultado parcial es consistente
8      assert(resultado >= a);
9      assert(resultado >= b);
10
11     // POST: el resultado cumple la especificación
12     assert(resultado == a + b);
13
14     return resultado;
15 }
16
```

PRE

Quien llama debe proveer datos válidos.

ASERCIONES

Validan estados intermedios durante la ejecución.

POST

Quien implementa garantiza que el resultado es correcto.

Aspectos a Tener en Cuenta

Confundir un dato de entrada con una precondición.

Creer que la terna compila y corrige el código automáticamente.

La verificación formal aporta rigor y previene errores lógicos desde la etapa de diseño de software.



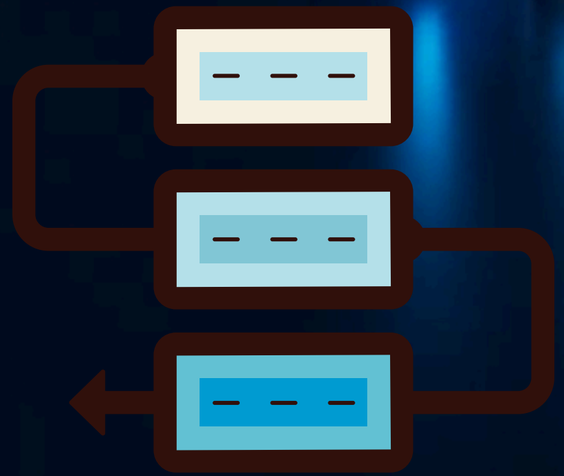
¿Qué es un Programa Secuencial?



Conjunto de instrucciones que se ejecutan una tras otra.

Sigue un orden determinado y lineal.

Cada paso inicia solo cuando el anterior ha terminado.



Objetivos de la Verificación

Detectar errores en el código

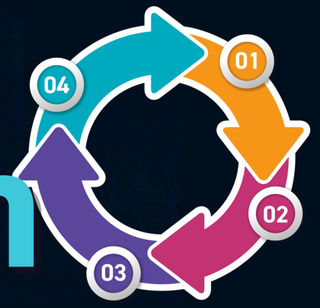
Comprobar que los resultados son correctos

Garantizar el cumplimiento de los requisitos

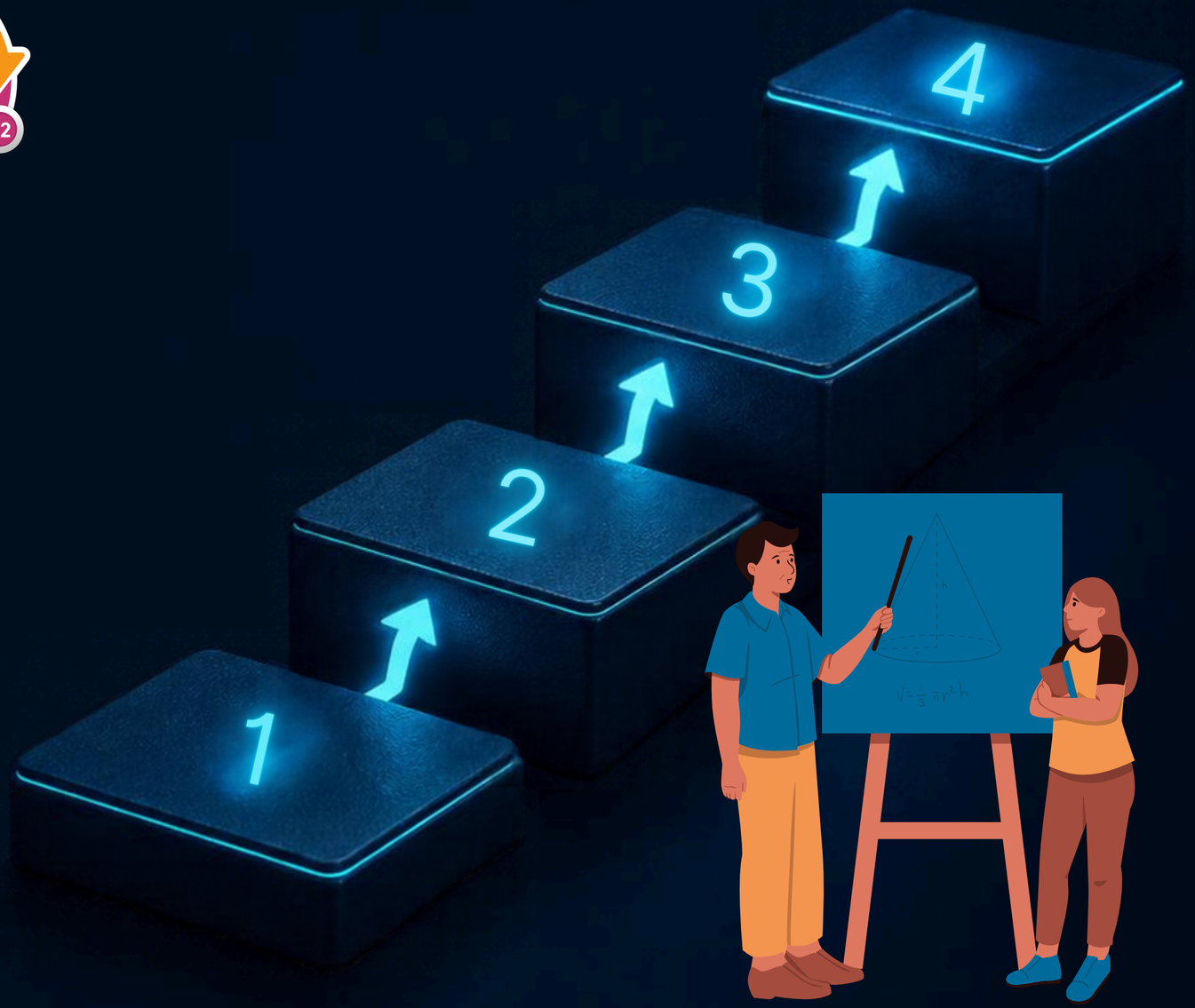
Mejorar la confiabilidad del software



Proceso de Verificación



1. Definir las especificaciones del programa
2. Revisar el código fuente
3. Ejecutar pruebas controladas
4. Comparar resultados obtenidos con los esperados



Técnicas de Verificación

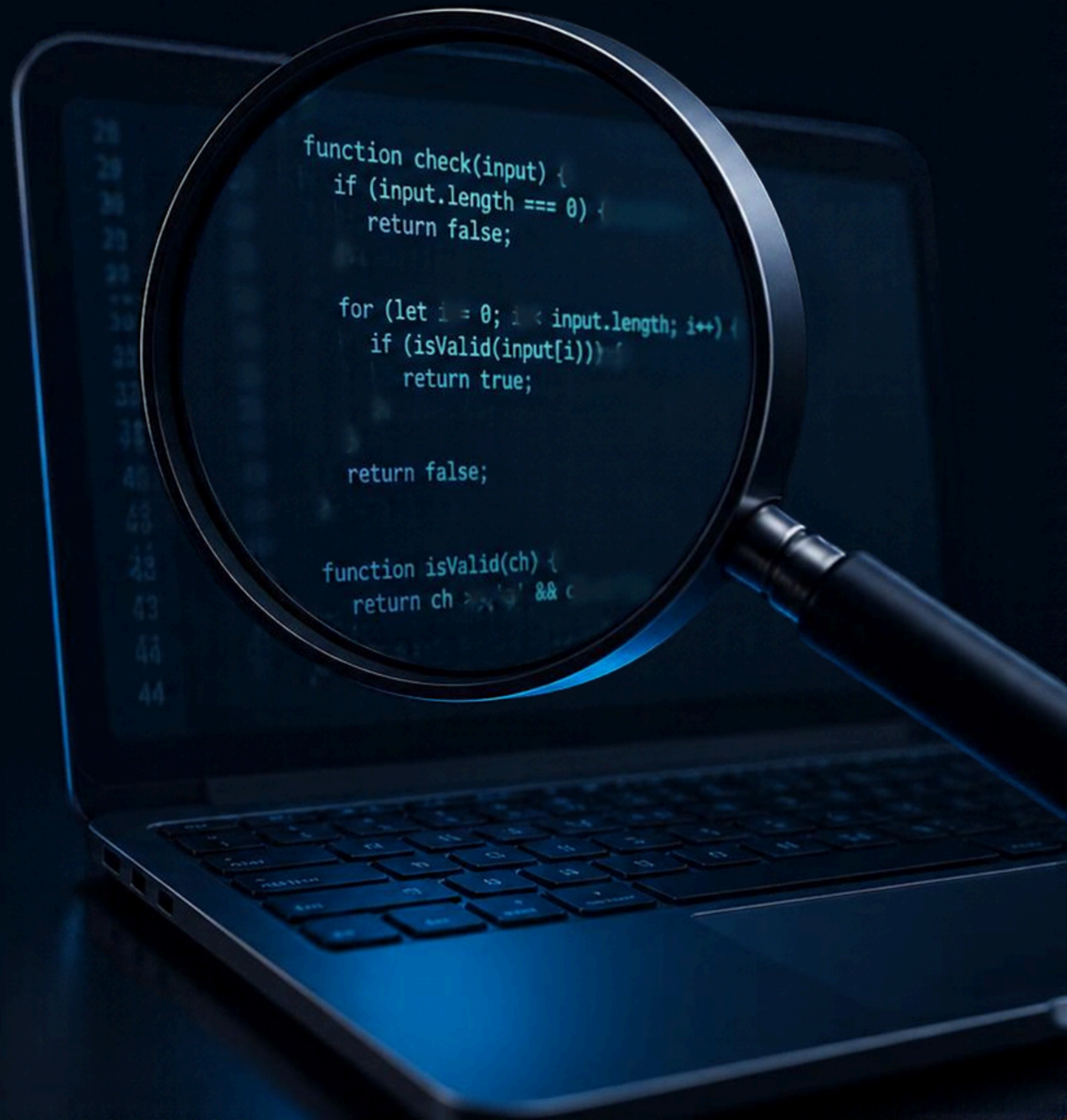
Pruebas de escritorio

Pruebas de ejecución

Depuración de errores

Verificación formal



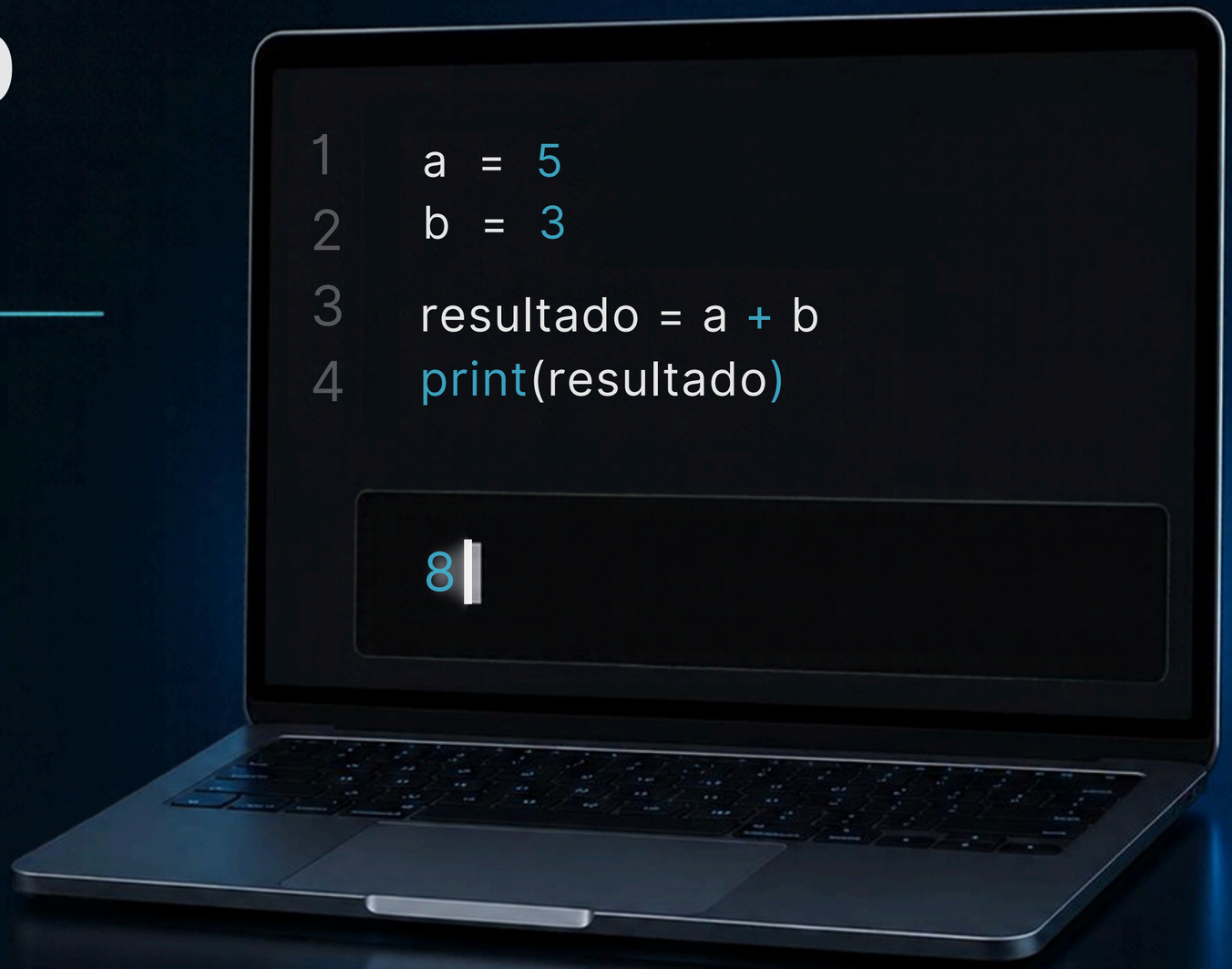


Ejemplo Práctico en Python

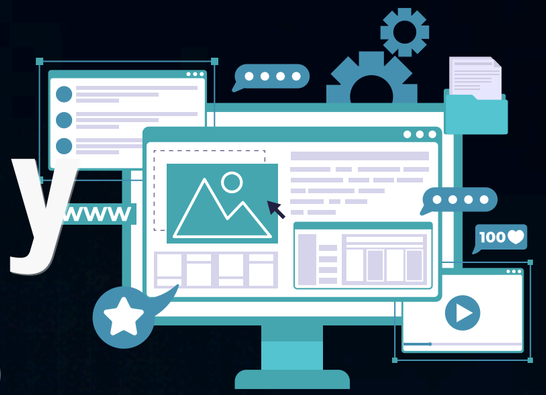
Este programa suma dos números y muestra el resultado en pantalla.

Resultado esperado: **8**

La verificación consiste en ejecutar el programa y confirmar que la salida coincida con el resultado esperado.



Precondiciones y Poscondiciones



Las **precondiciones** establecen lo que debe cumplirse antes de ejecutar el programa.

Las **poscondiciones** definen lo que debe cumplirse después de la ejecución.

Ejemplo

Si saldo > monto antes de transferir, entonces saldo' = saldo – monto después.



ANTES

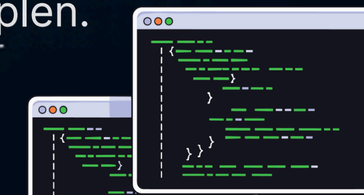
Las condiciones necesarias se cumplen.

EJECUCIÓN

El programa se ejecuta.

DESPUÉS

Las condiciones esperadas se cumplen.



Importancia y Conclusión



La verificación detecta errores y mejora la calidad del software

Aumenta la confiabilidad de los programas secuenciales

La verificación asegura que las instrucciones se ejecuten correctamente y produzcan resultados esperados

Verificar es construir software más confiable y de mayor calidad.

