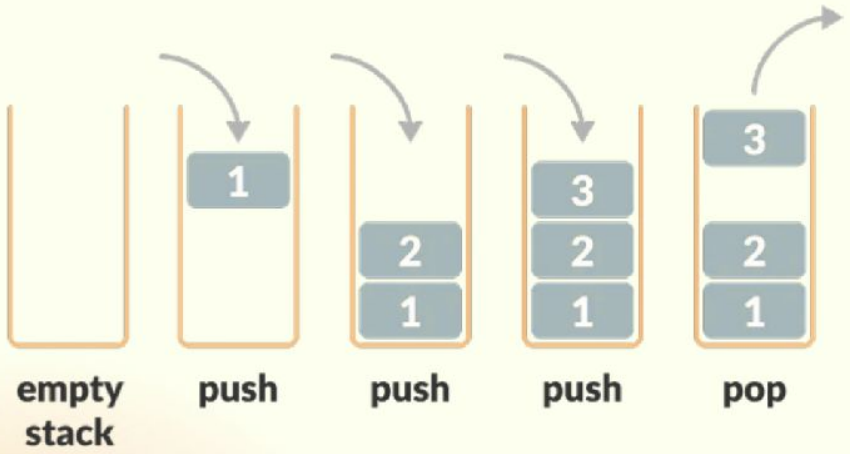
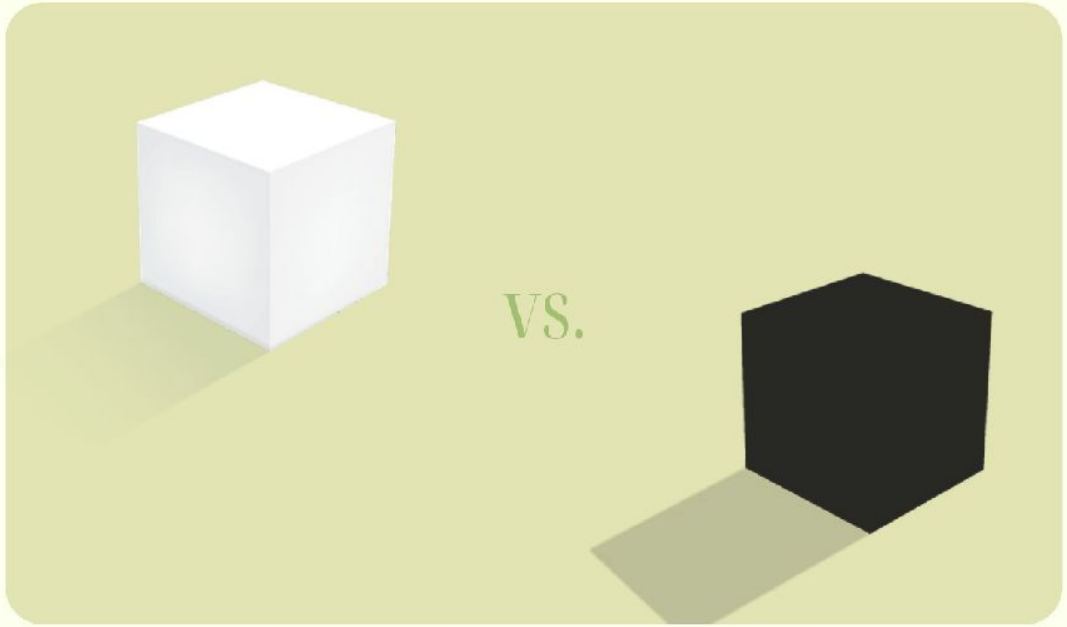
TDA Pila



¿Qué es un TDA?



La **caja negra**. "No me importa cómo está hecho, me importa **lo qué hace**."

TDA Pila

¿ Qué es?

Una **pila** es una colección de elementos en la que pueden insertarse y eliminarse por un **extremo**, denominado **tope**, sus elementos.



Algunos ejemplos de un comportamiento de un TDA Pila pueden ser un tubo de papas pringles, una pila de platos, una pila de libros, bandeja de entrada de correos,..



En esencia, una pila es un contenedor de elementos donde toda la acción ocurre en un solo lugar: el tope.
Esta restricción en el acceso es lo que le da su poder y su comportamiento característico.

PILA : Estructura LIFO

LIFO significa Last-In, First-Out, que en español se traduce como "**Último en Entrar, Primero en Salir**".

Este es el principio que define a una pila y la diferencia de otras estructuras de datos.







TDA Pila

Normalmente existe un conjunto de operaciones que se puede realizar sobre un tda.

¿Estas operaciones son estándares? No.

Existe un pequeño conjunto que siempre debería estar y después las mismas varían según las necesidades del programador.

Al conjunto que siempre debería estar se lo denominará: **conjunto mínimo de operaciones.**

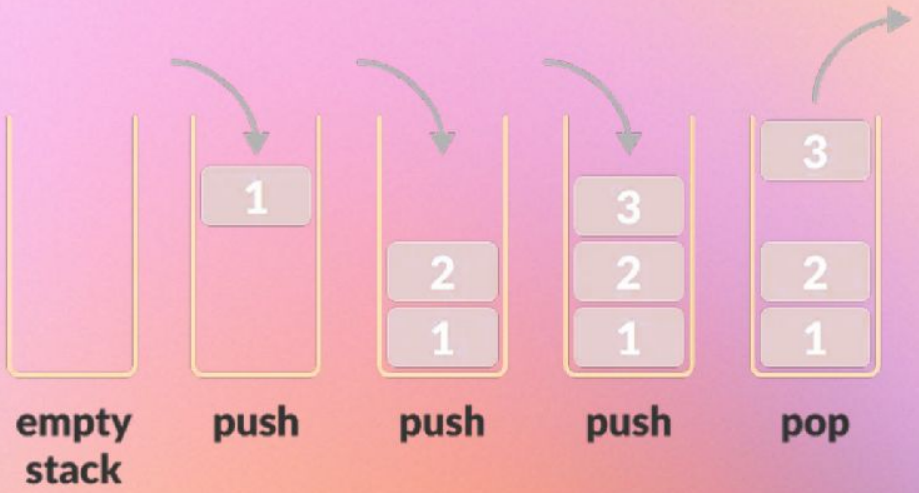


TDA Pila

Conjunto mínimo de operaciones

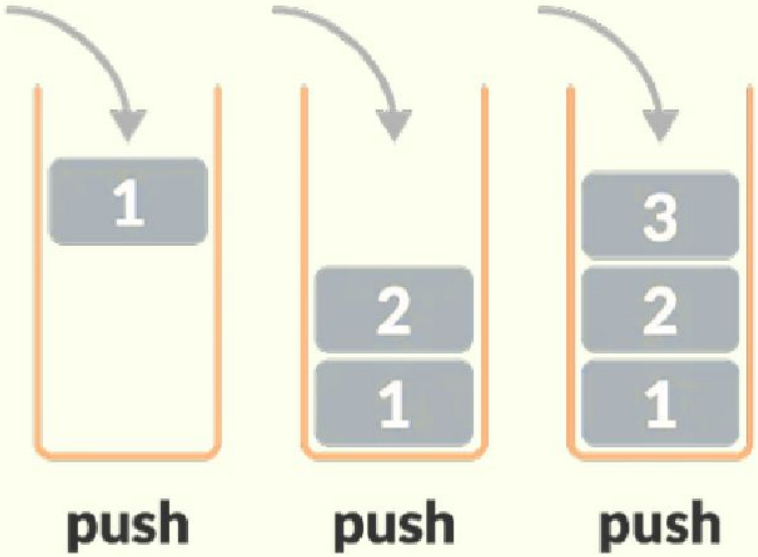
Para el caso de una pila el conjunto mínimo de operaciones es:

- crear (create)
- poner (push)
- sacar (pop)
- tope (top)
- esta_vacia (is_empty)
- destruir (destroy)



TDA Pila: apilar()

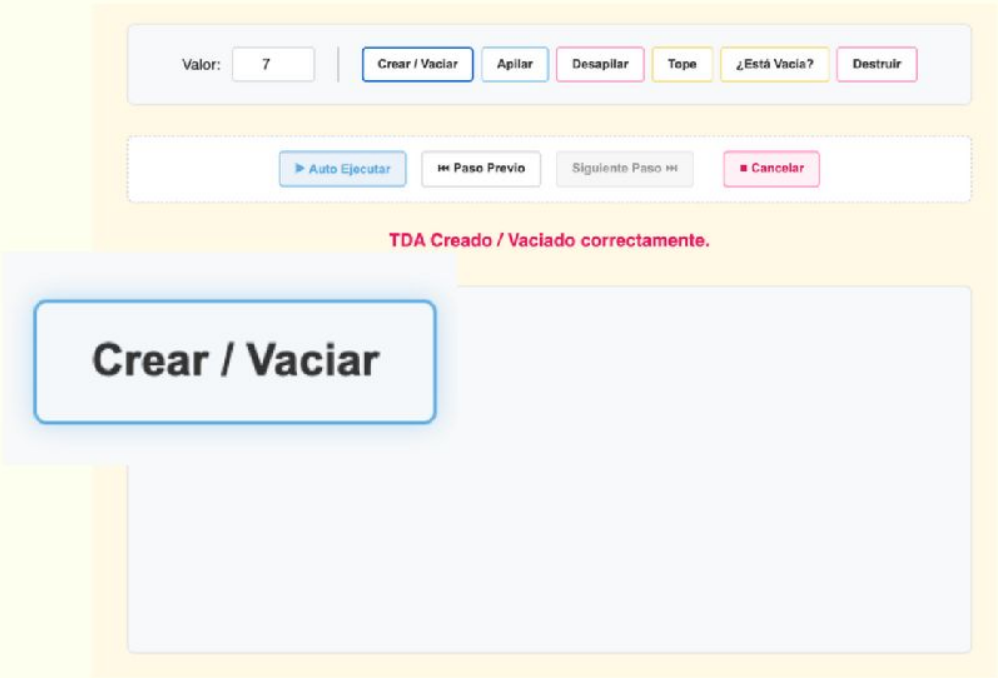
insertar() / push()



TDA Pila: apilar()

insertar()

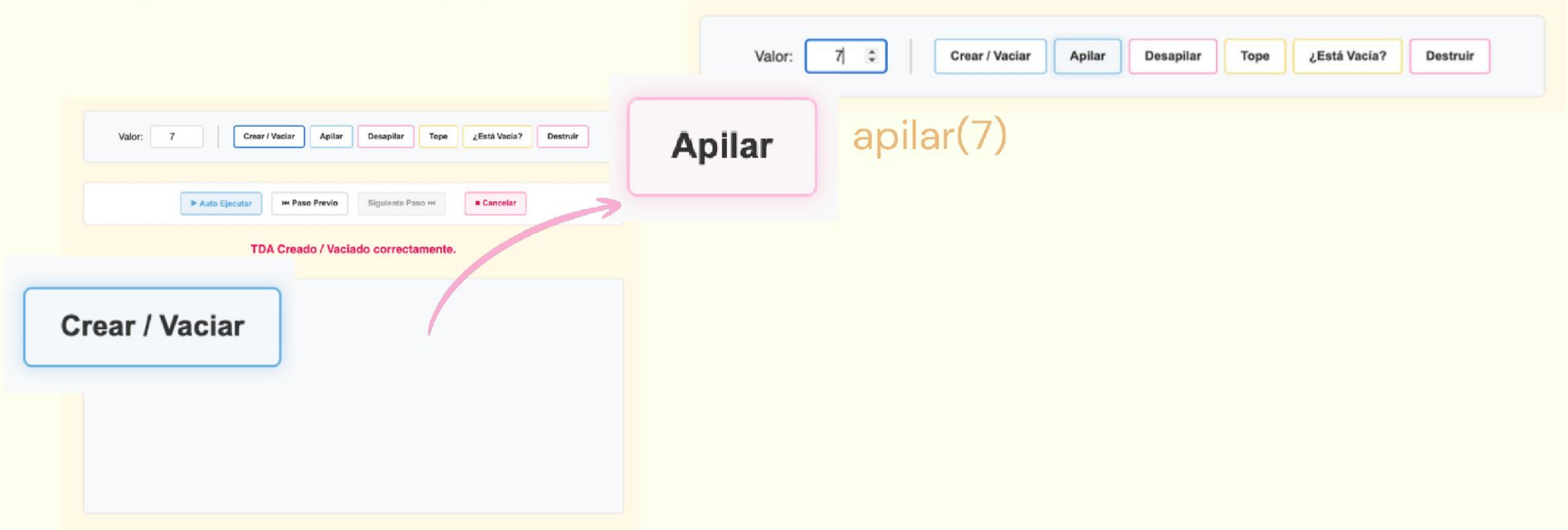
Esta operación pone un elemento en la pila por el tope de la misma, haciendo que el tope de la pila pase a ser el nuevo elemento introducido.



TDA Pila: apilar()

insertar()

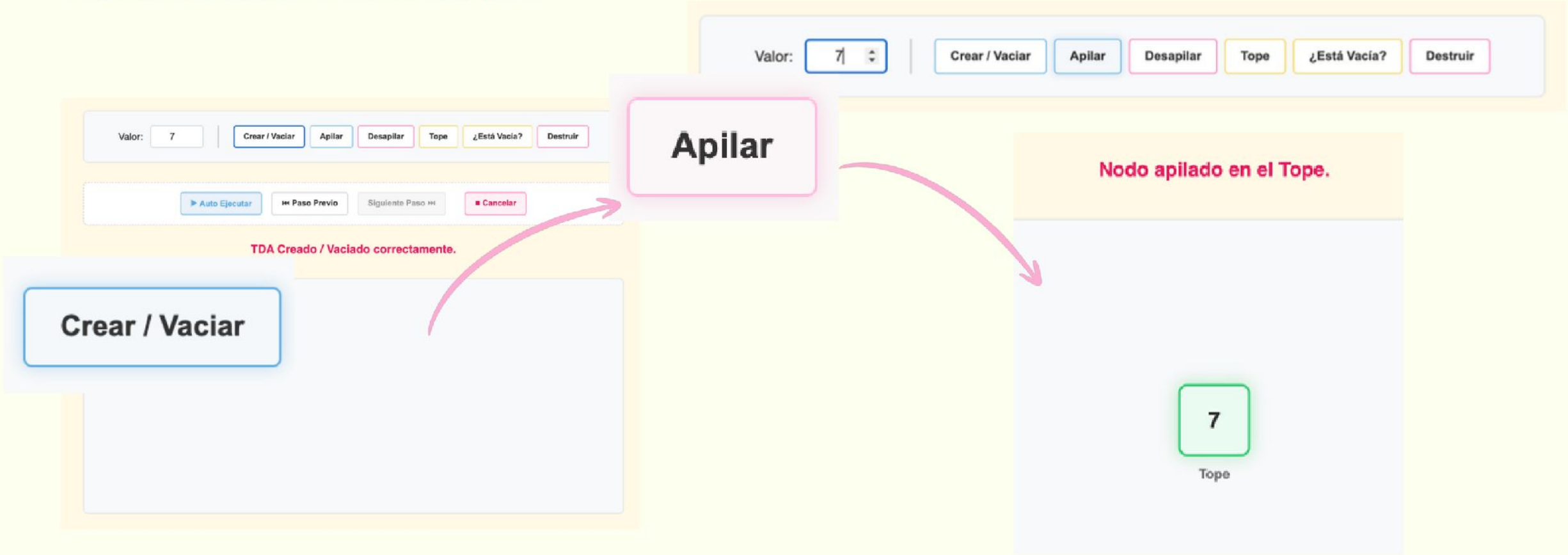
Esta operación pone un elemento en la pila por el tope de la misma, haciendo que el tope de la pila pase a ser el nuevo elemento introducido.



TDA Pila: apilar()

insertar()

Esta operación pone un elemento en la pila por el tope de la misma, haciendo que el tope de la pila pase a ser el nuevo elemento introducido.

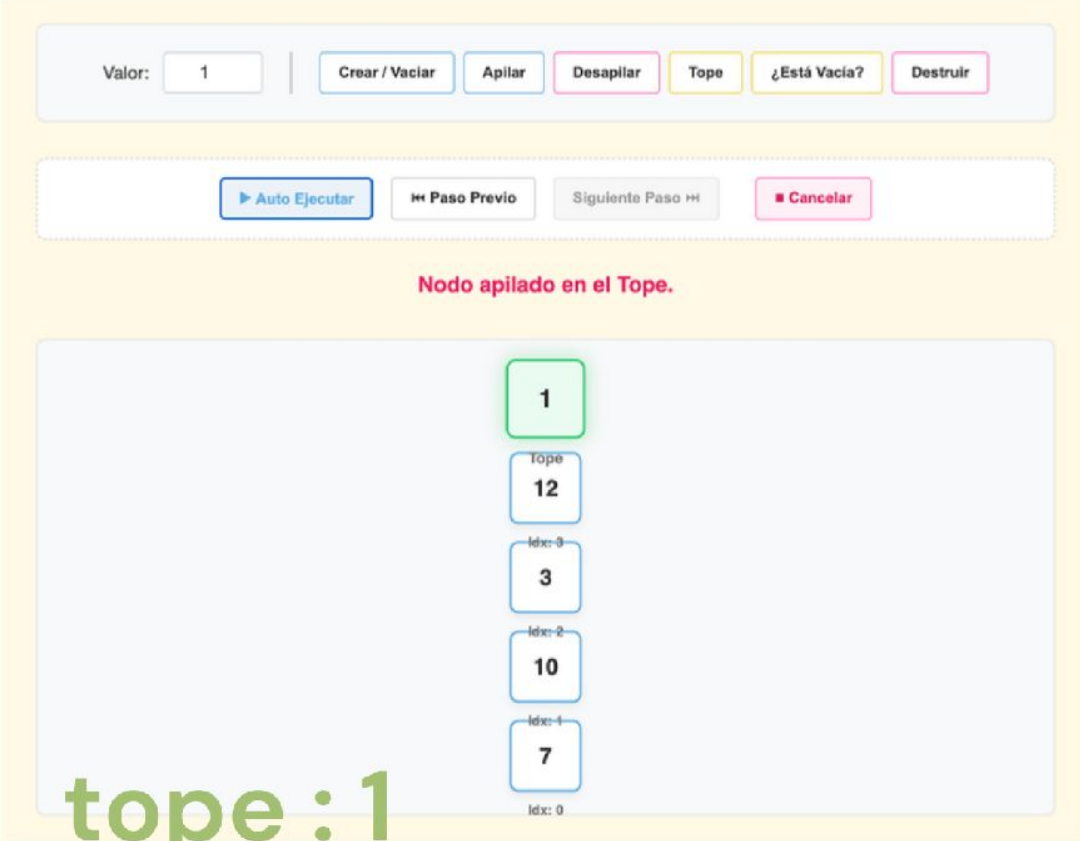


TDA Pila: apilar()

insertar()



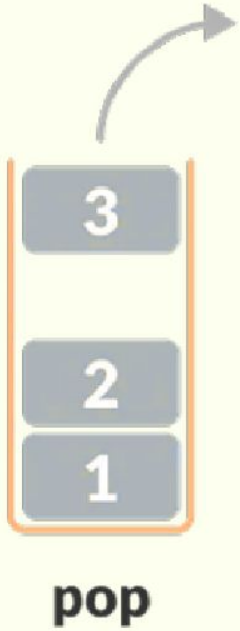
tope : 10



tope : 1

TDA Pila: desapilar()

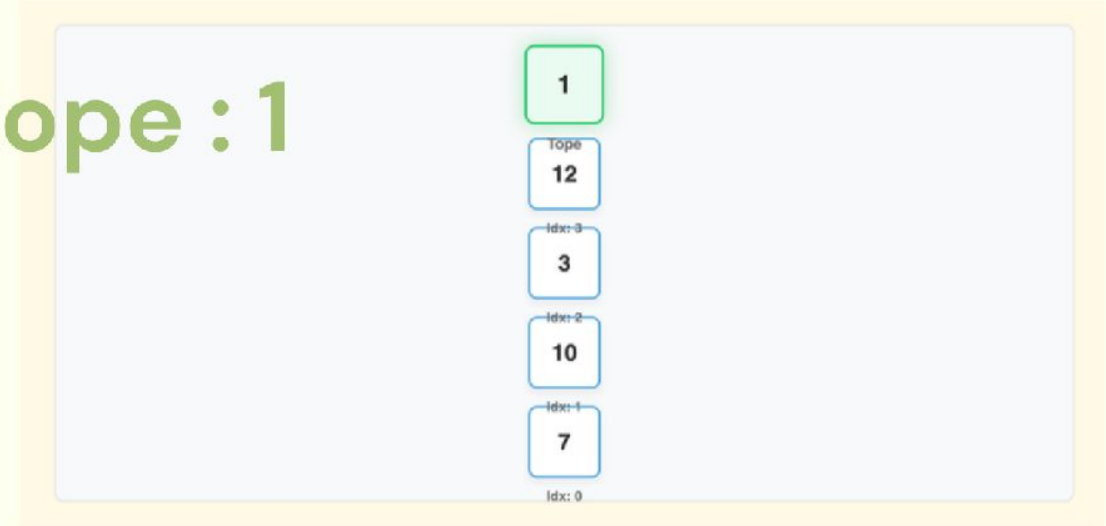
eliminar()



TDA Pila: desapilar()

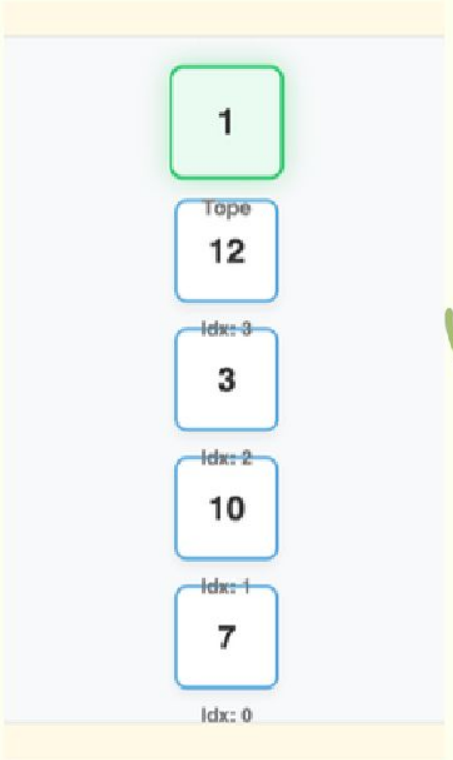
eliminar()

Esta operación **retira el elemento del tope de la pila** y mueve el tope de la pila al elemento anterior al extraído, si el elemento extraído es el último la pila queda vacía:

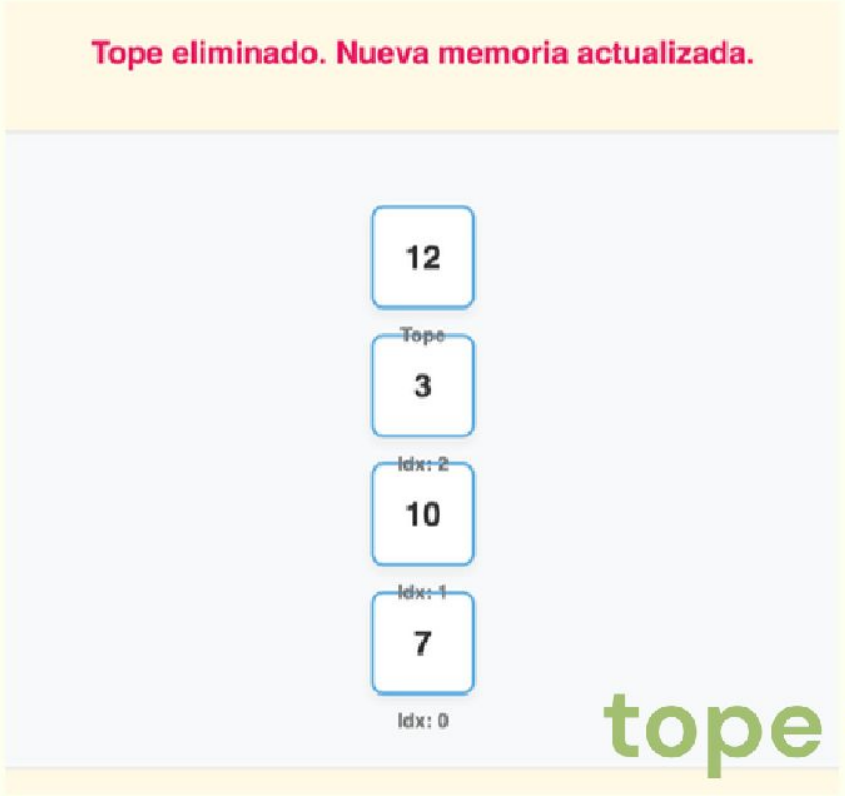


TDA Pila: desapilar()

eliminar()



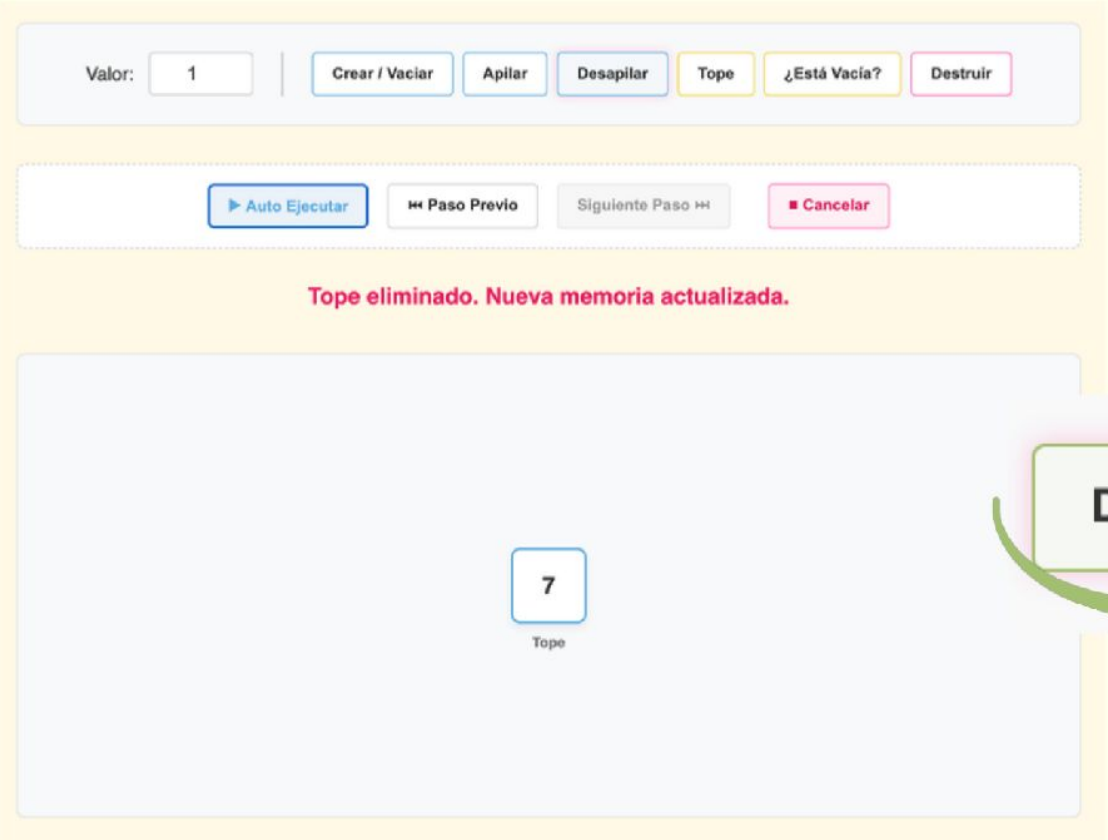




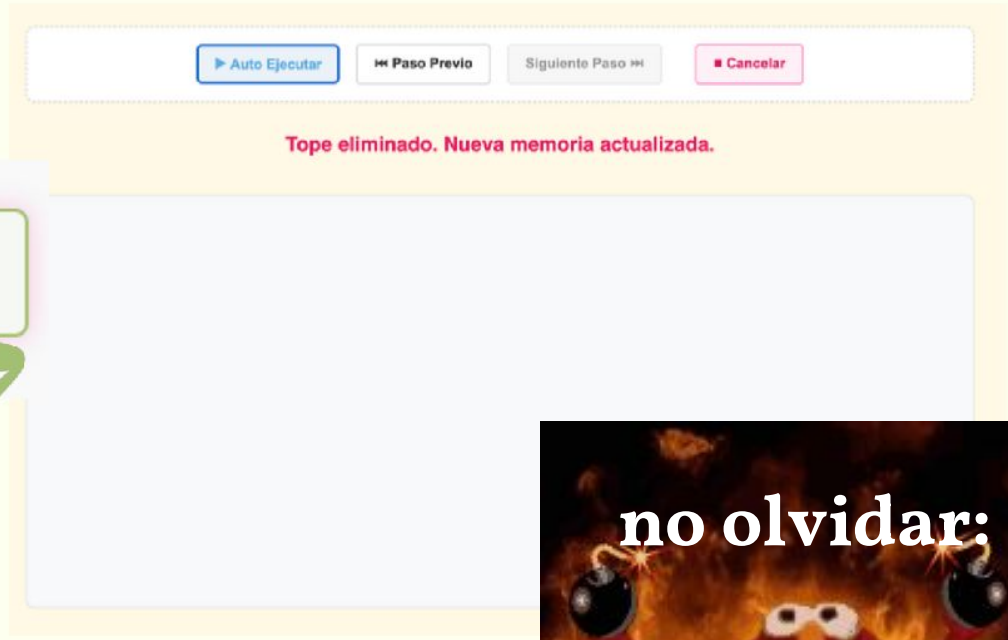


TDA Pila: desapilar()

eliminar()



Desapilar



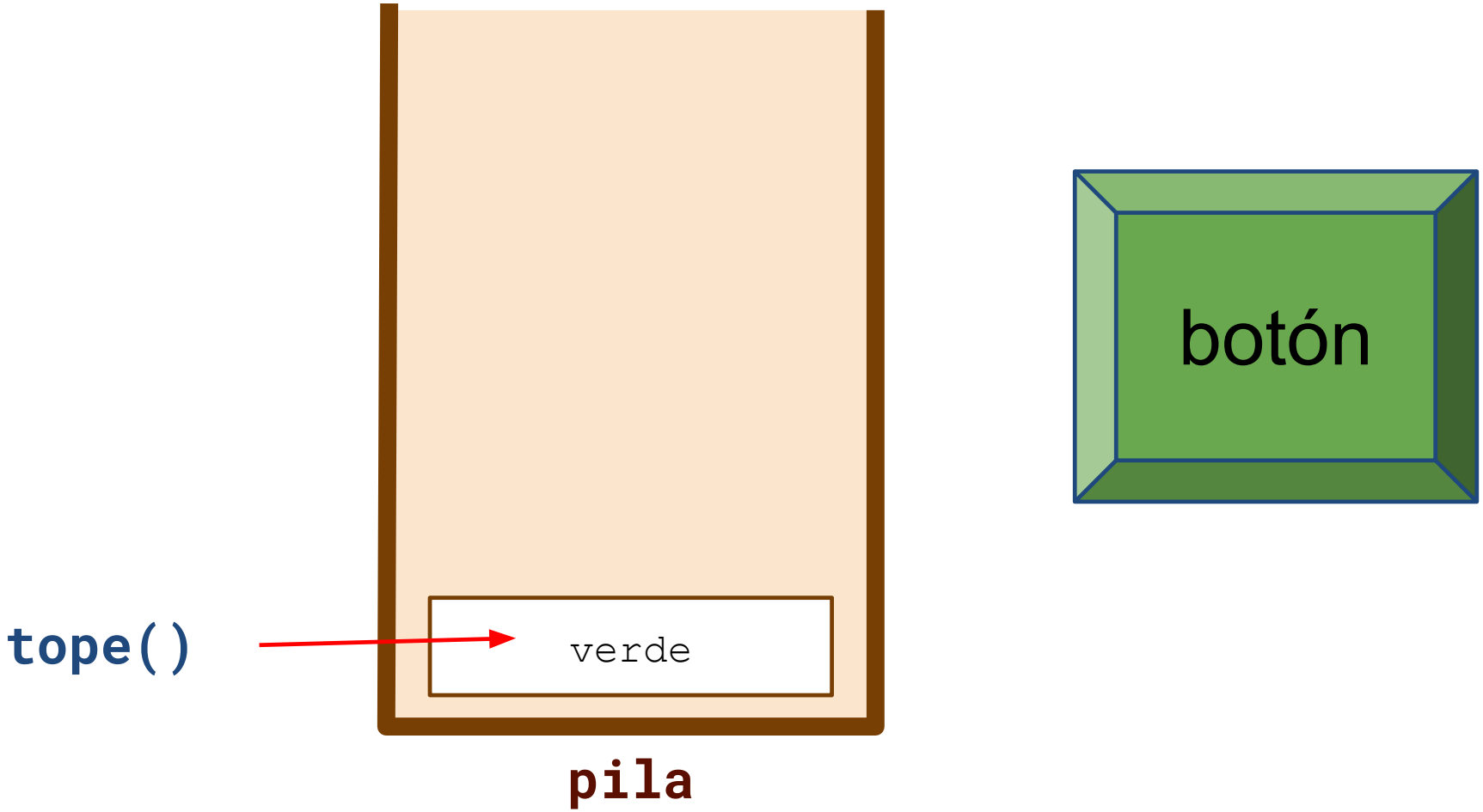
TDA Pila

- **crear** (create): Esta operación se ocupa de crear y reservar todos los recursos iniciales que se utilizarán para la pila.
- **tope** (top): Esta operación permite observar el valor del tope de la pila.
- **esta_vacia** (is_empty): Esta operación permite determinar si una pila tiene o no tiene elementos a través de un valor de verdad.
- **destruir** (destroy): Esta operación se ocupa de liberar y limpiar todos los recursos que se utilizan para la creación de una pila.

¿Cuándo usar una pila?

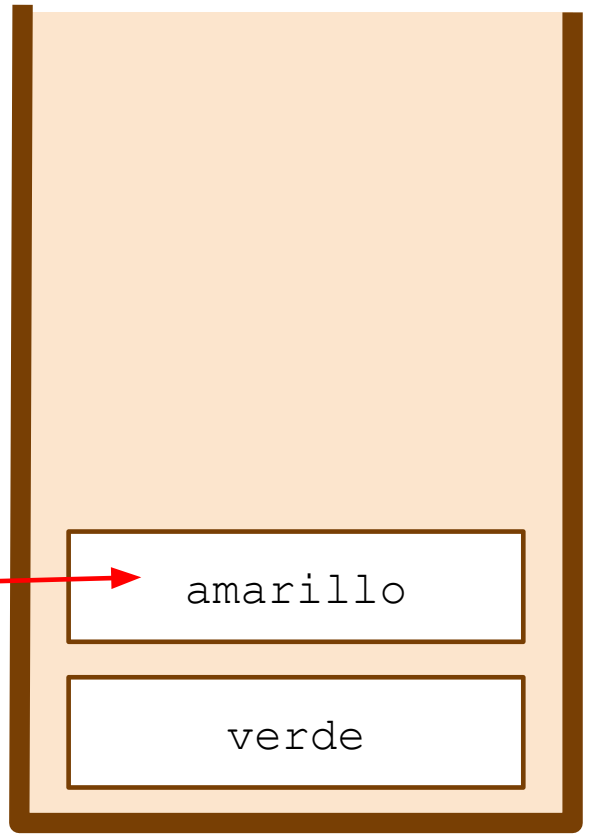
- Hacer/ Deshacer
- Orden Inverso
- Análisis de expresiones
- Control de funciones (Call Stack)

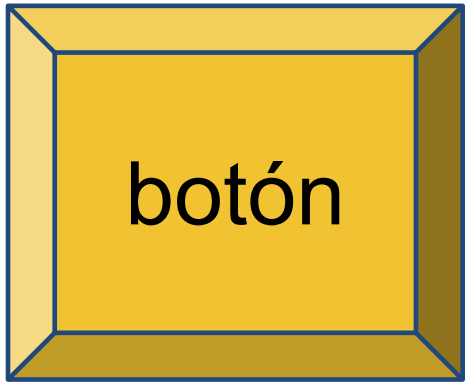
Usos: hacer y deshacer



Usos: hacer y deshacer

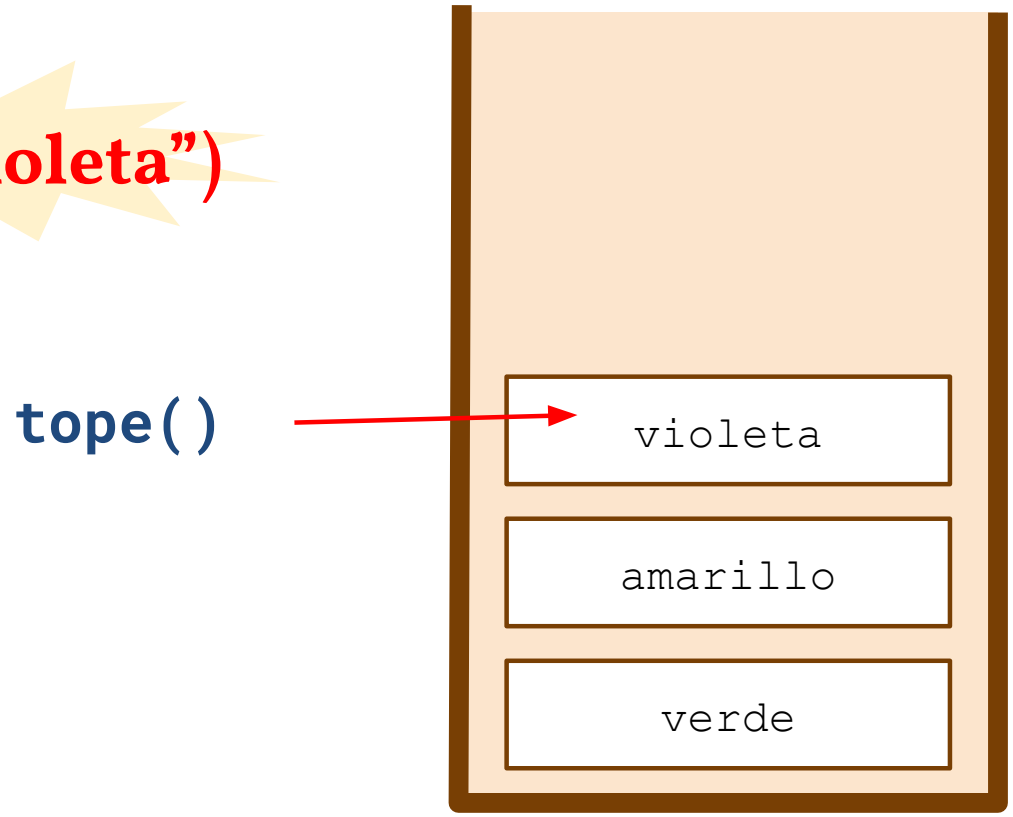
push(“amarillo”)

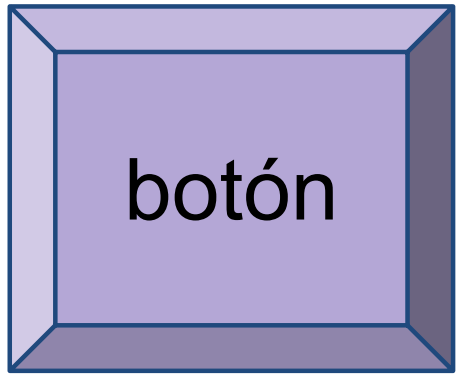




Usos: hacer y deshacer

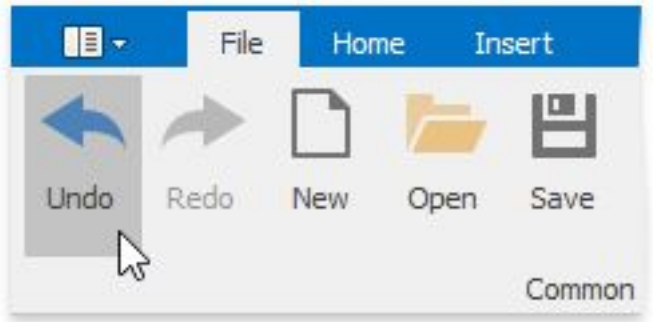
push(“violeta”)



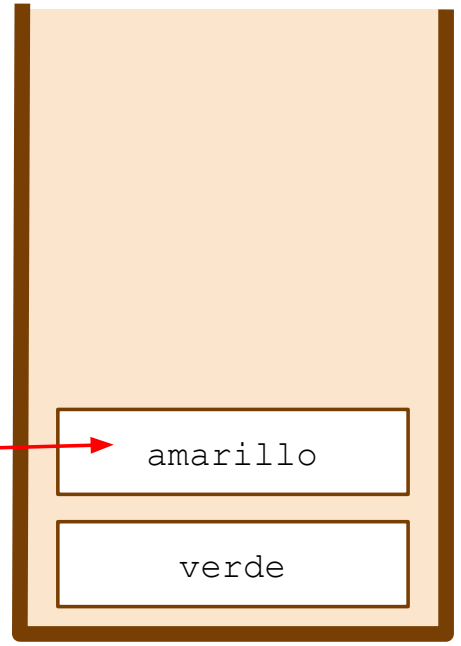


Usos: hacer y deshacer

pop



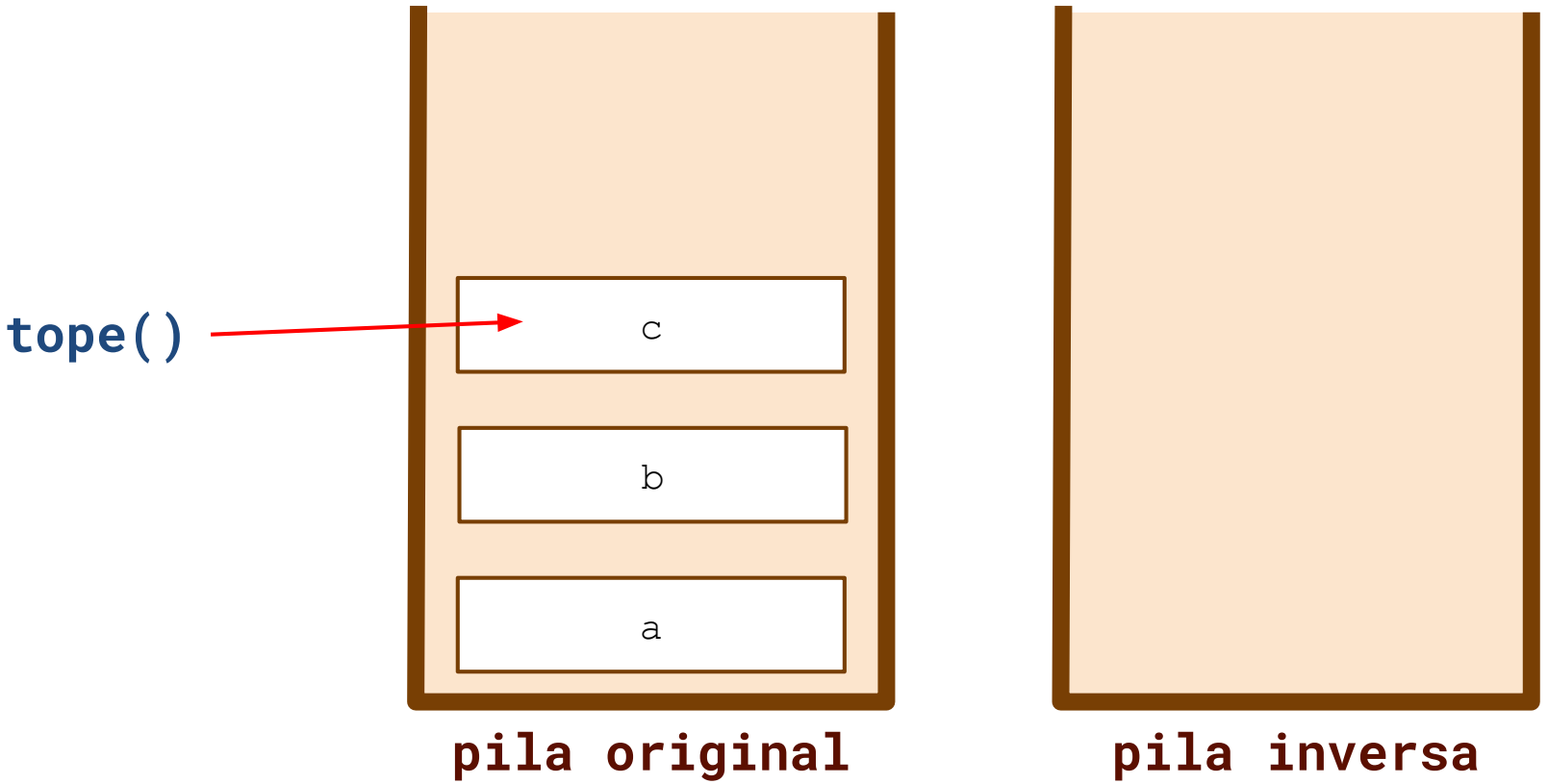
`tope()`



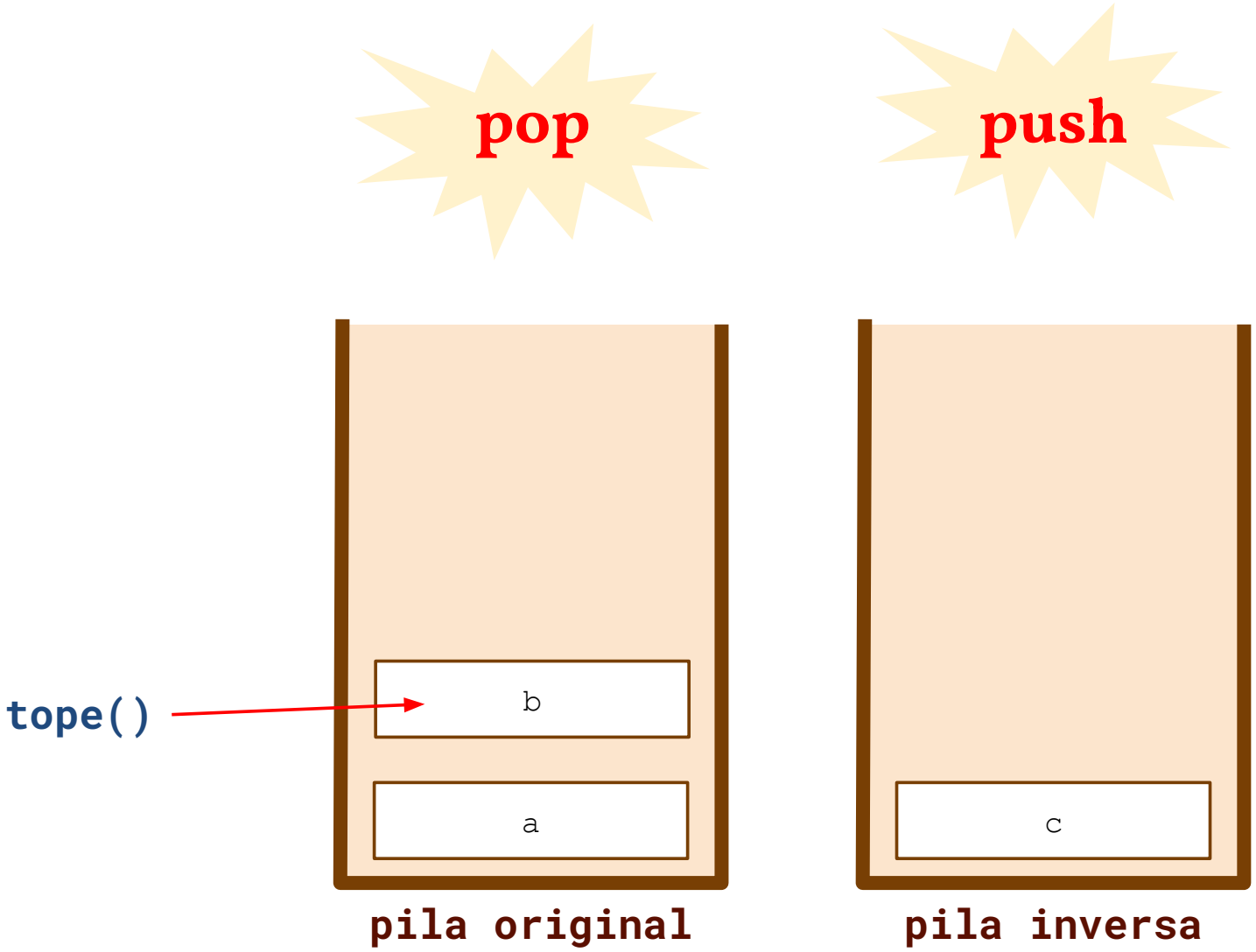
`pila`

botón

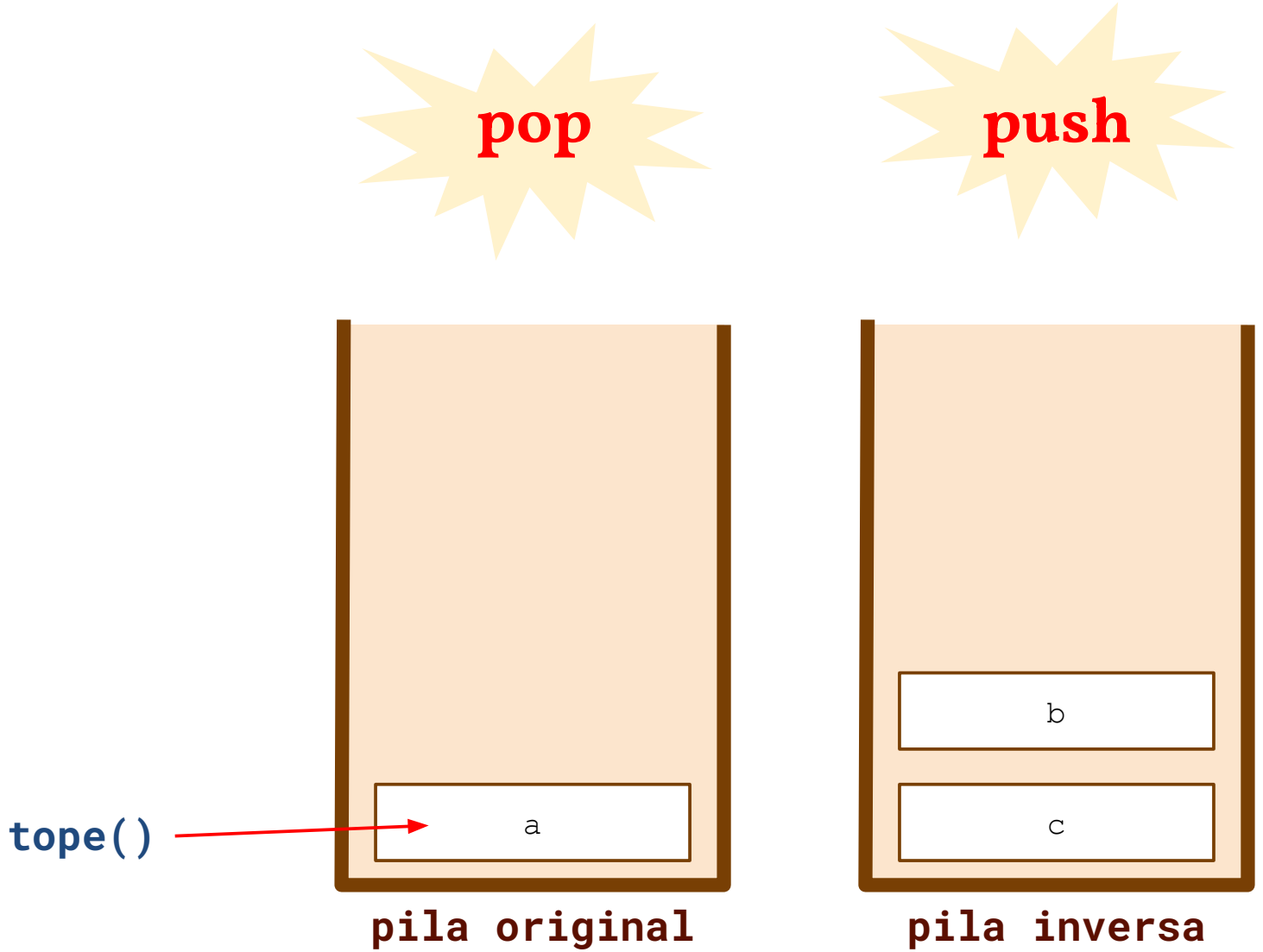
Usos: orden inverso



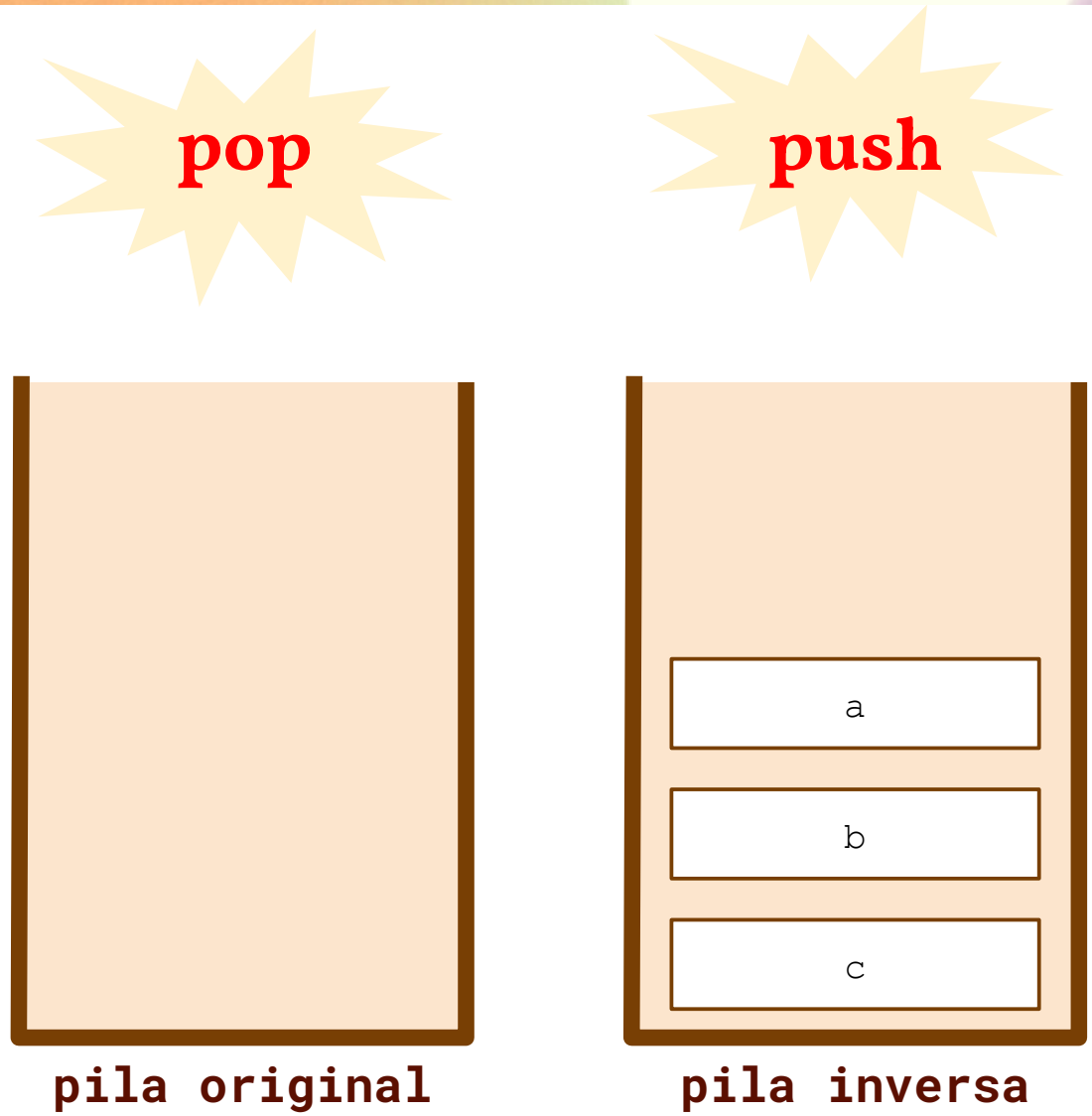
Usos: orden inverso



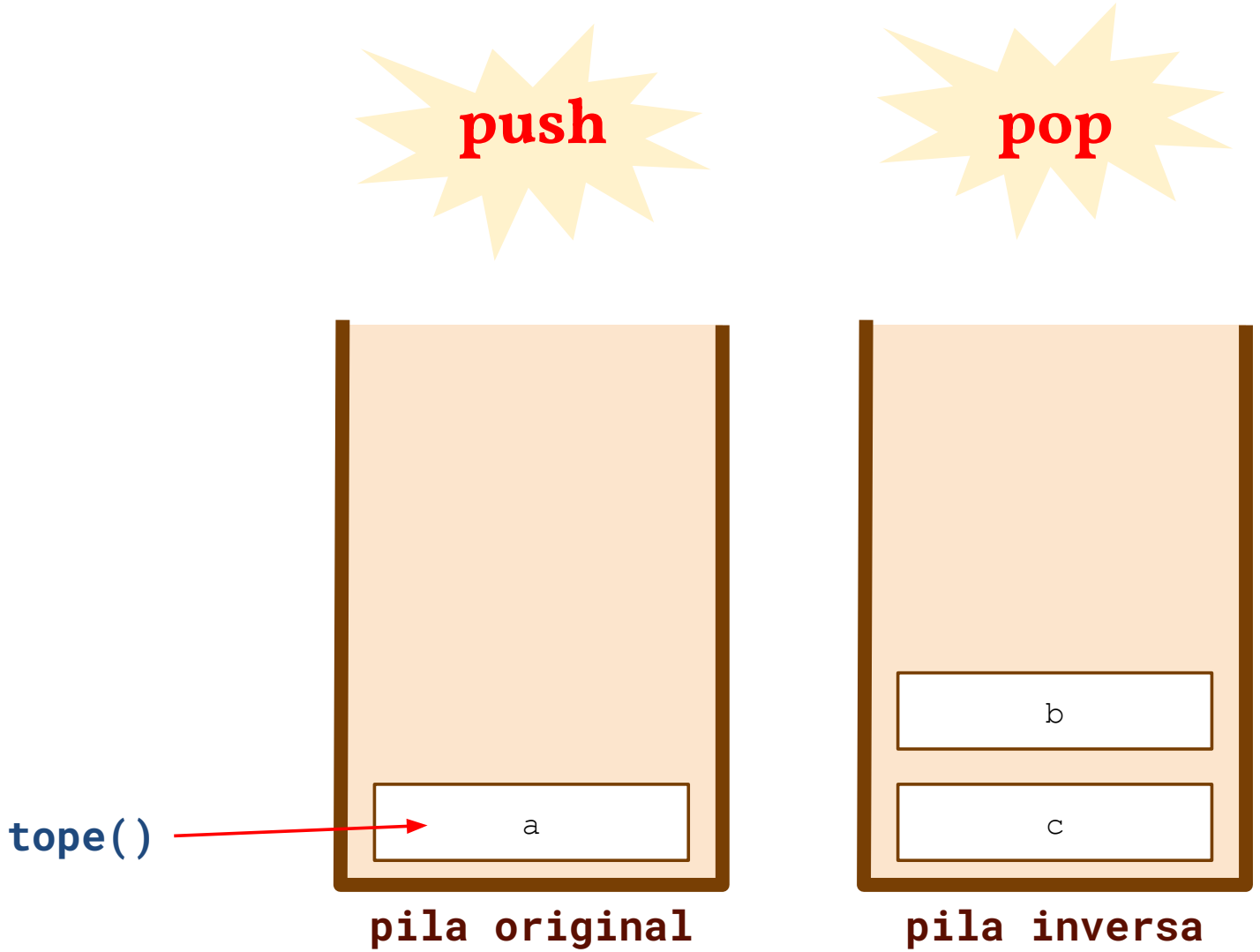
Usos: orden inverso



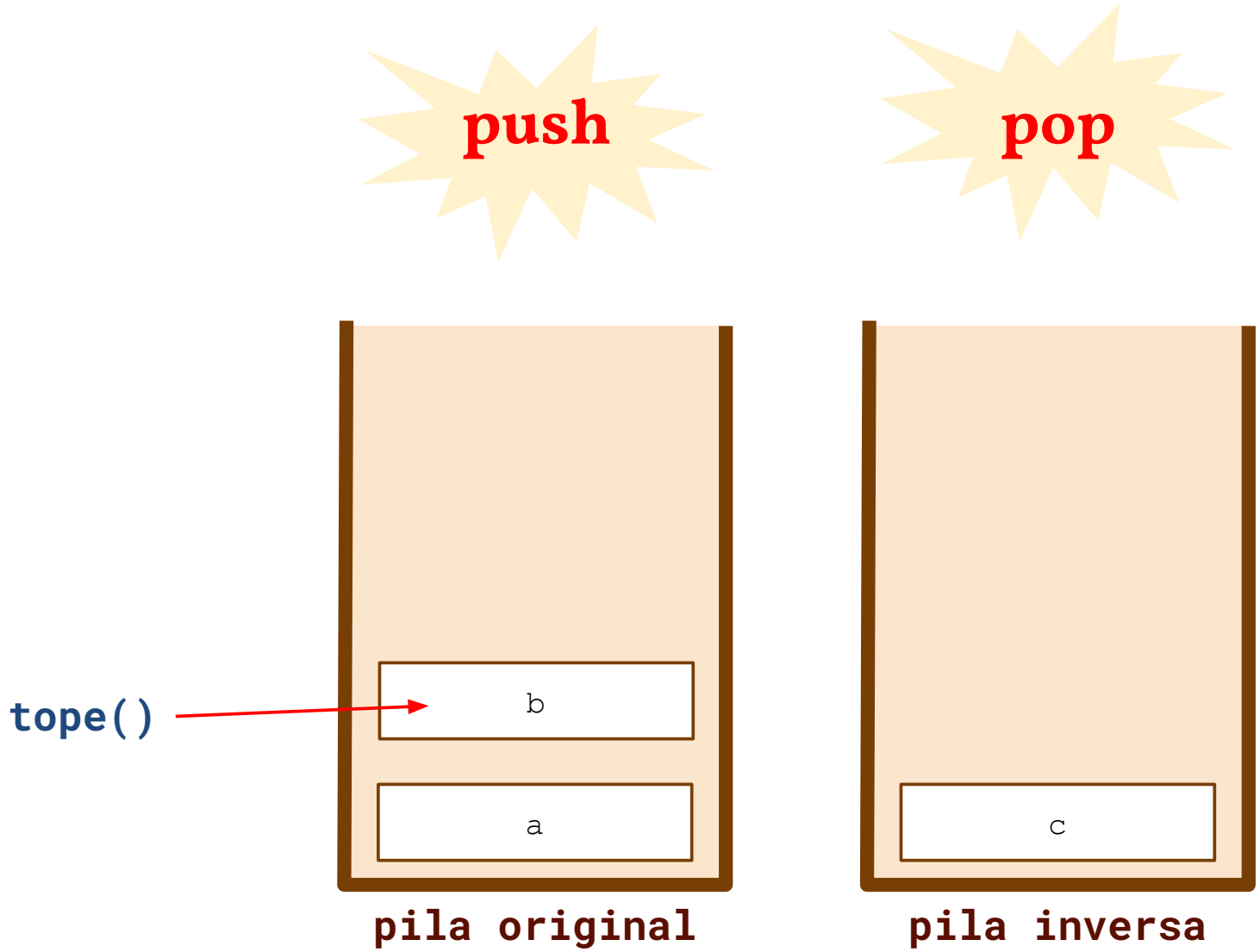
Usos: orden inverso



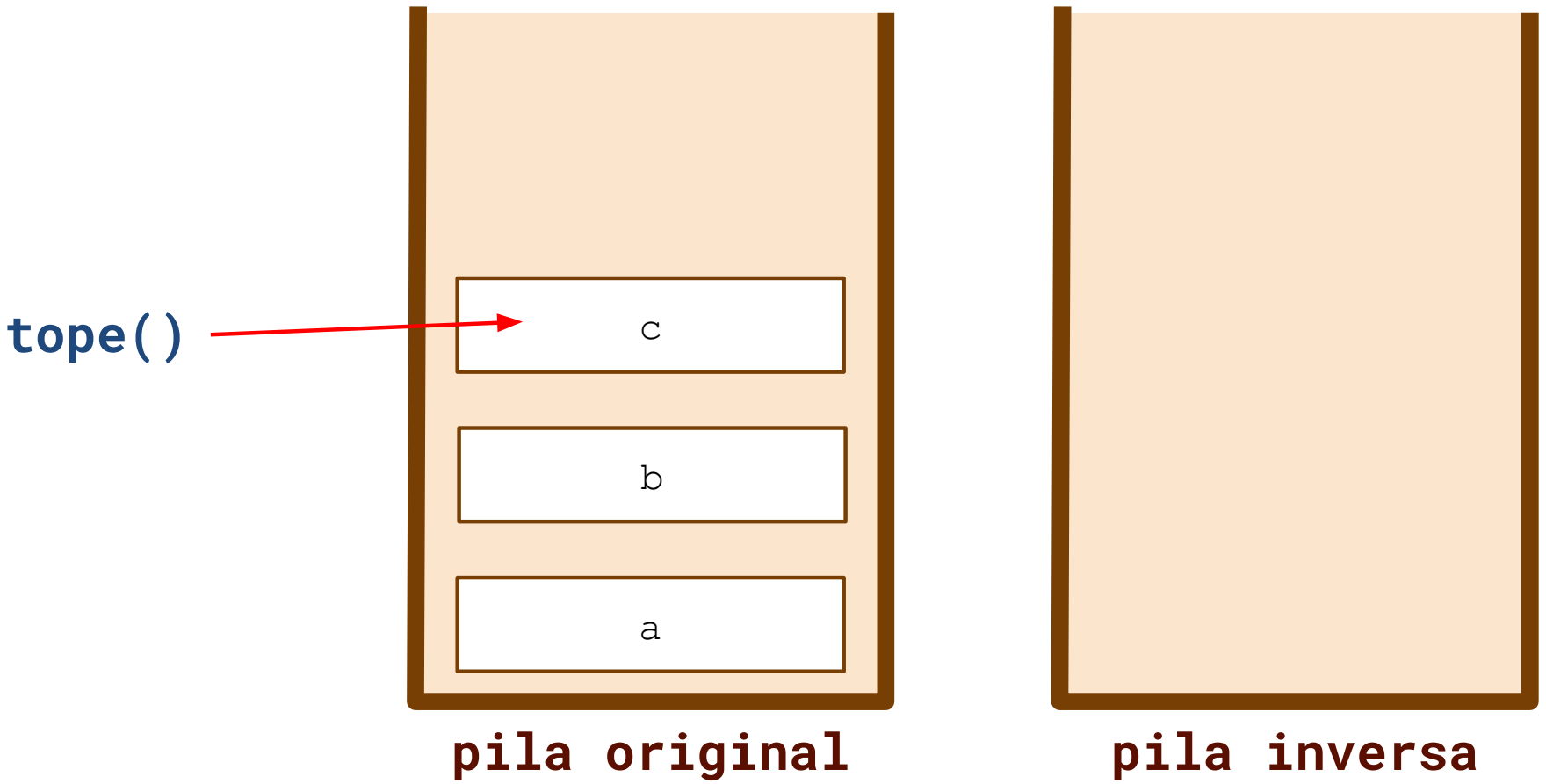
Usos: orden inverso



Usos: orden inverso

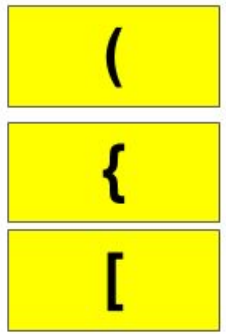


Usos: orden inverso



Usos: análisis de expresiones

[{ ( ) }]

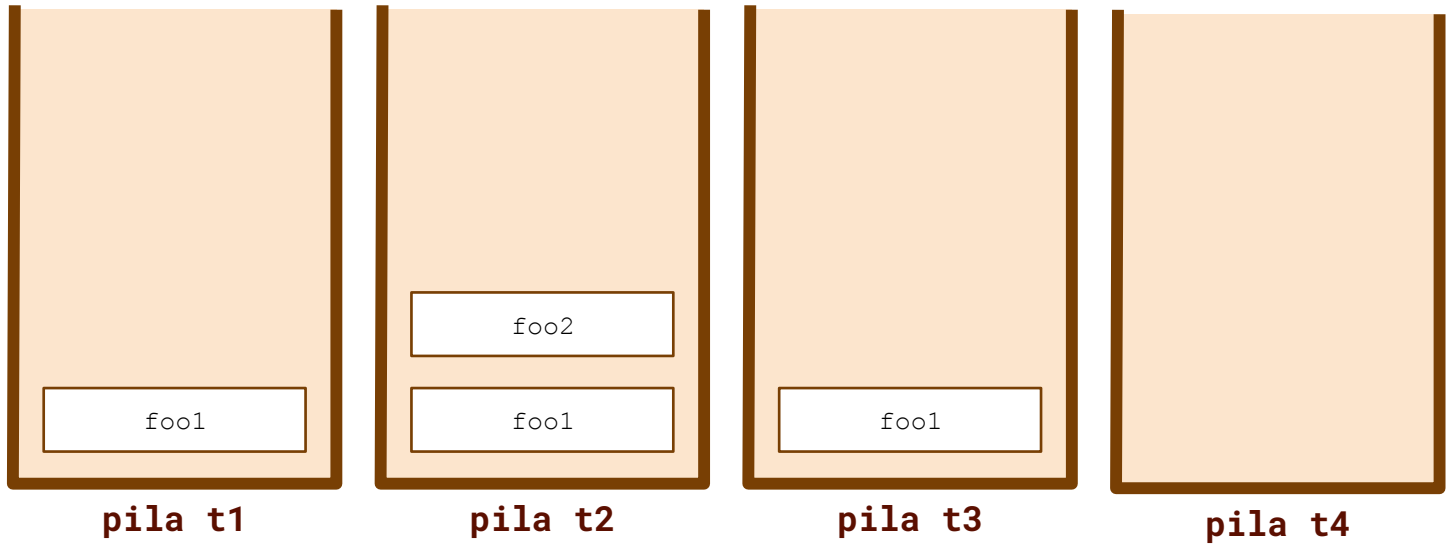


The stack

Usos: control de funciones (Call Stack)

```
void foo1() {
     print(“Hola 1\n”);
    foo2();
     print(“Chau 1\n”);
}

void foo2() {
     print(“Foo\n”);
}
```



Pros

- Simples de implementar.
- Optimización: Operaciones rápidas.
- Controlan el orden de manera natural:
  - Orden inverso
  - Recursividad
  - Historial

Contras

- Acceso limitado de los elementos (**tope**).
- Implementación suele implicar un tamaño fijo: **stack overflow**.
- No sirve para problemas en los que necesitás acceso aleatorio, ordenar, buscar por posición, etc.

¿ Preguntas ?

Ejercicios

Homero tiene un tubo con donas que compró en el kwik e mart y Marge quiere saber en qué orden las comería , sabiendo que Homero siempre agarra la dona de arriba del tubo.

Dado un paquete de donas, implementar una función que imprima el orden en el cúal Homero comerá las donas. El paquete debe quedar vacío al final (Marge ya sabe que Homero nunca dejaría unas donas por la mitad).

