Errores en Go

En Go, los errores son una parte esperable y manejable del flujo del programa, representan comportamientos esperables o inevitables en el programa.
Las funciones que pueden encontrar un error usualmente retornan un valor adicional del tipo `error` junto con el resultado esperado.
El tipo error es una interfaz que define un único método `Error() string`, el cual devuelve un mensaje de error.

Estrategias de Manejo de Errores
Cuando una función produce un error, es responsabilidad del "llamador" (la función que la invoco) verificarlo y tomar la acción apropiada.

1.Propagacion del error: La función simplemente devuelve el error recibido a su propio llamador, permitiendo que la lógica de manejo se realice en un nivel superior-

2.Reemplazo/Envoltura de error: Se crea un nuevo error con un mensaje mas contextual o especifico, que puede envolver el error original. Esto se hace comúnmente con `fmt.Errorf.`
`fmt.Errorf`: formatea el mensaje de error utilizando fmt.Sprintf y devuelve un nuevo valor de tipo **`error`**.

3.Reintentar la operación: Si el error es temporal, se puede intentar ejecutar la operación nuevamente, a menudo con un retraso exponencial.

4.Terminacion controlada: Si es posible evitar o recuperarse de un error crítico, el programa puede terminar de manera controlada, registrando el error y saliendo con un código de estado apropiado.

5.Registrar el error y continuar: Si el error no es fatal, se puede registrar y permitir que el programa continúe su ejecución, aunque quizás con alguna funcionalidad deshabilitada.

6.Ignorar el error: En algunos casos el error puede ser no critico i manejable con un valor por defecto, permitiendo ignorar el error.

Go tiene un mecanismo para manejar errores esperables o situaciones excepcionales
`Panic` y `Recover`.

• Panic: Ocurre cuando go detecta un error en tiempo de ejecución
Ej: división por cero, acceso a un puntero nil sin manejo explicito).
- Detiene la ejecución normal del programa.
- Puede ser generado explícitamente por el programador
- Se utiliza para errores que el programa no puede o no debe manejar, como un fallo irrecuperable o una condición inesperada.
- Cuando un panic ocurre las funciones diferidas (defer) se ejecutan antes de que el programa falle.

• Recover: Es una función que permite recuperar la ejecución ante un panic o al menos realizar acciones de limpieza.
- Solo tiene efecto si es invocada dentro de una función diferida (defer) durante un panic.
- Al ser llamada en un defer durante un panic, recover() finaliza el estado de pánico y retorna el valor de panic.
- La función que entro en pánico no continua, pero el programa puede evitar una falla total.
- Si el recover es invocado en cualquier otro momento (fuera de un defer durante un panic), no tiene ningún efecto y retorna nil.