- ¿Cuál es la complejidad temporal de cada función implementada?

Todas las funciones de búsqueda implementadas utilizan **búsqueda lineal**, por lo tanto su **complejidad temporal es O(n)**, donde **n** es la cantidad de elementos en la lista.
Esto significa que, en el peor de los casos, el algoritmo debe recorrer **todos los elementos** antes de encontrar el resultado o determinar que no existe.

- ¿En qué casos la búsqueda lineal es eficiente?

La búsqueda lineal es eficiente cuando:

- La **lista es pequeña o de tamaño moderado**.
- No se requiere realizar **muchas búsquedas repetidas**.
- Los datos **no están ordenados**, ya que la búsqueda lineal no necesita orden previo.
- Se busca una **implementación simple y rápida de programar**, sin estructuras adicionales.

- ¿Cuándo sería mejor usar otro algoritmo de búsqueda?
- La lista es **muy grande** (miles o millones de elementos).
- Las búsquedas se realizan **frecuentemente** y se necesita mayor rendimiento.
En estos casos se recomienda:
- **Búsqueda binaria (O(log n))**, si los datos están ordenados.
- **Estructuras de datos avanzadas** como diccionarios (`dict`) o árboles balanceados, que permiten búsquedas en **O(1)** o **O(log n)**.

- ¿Qué pasa si la lista está vacía?
- El algoritmo **no entra al ciclo de búsqueda**, por lo tanto no encuentra ningún elemento.
- En ese caso, la función devuelve un **mensaje de "no encontrado"** o una **lista vacía**, dependiendo de la implementación.
No produce error, simplemente no hay coincidencias que mostrar.

- ¿Cómo manejar búsquedas con mayúsculas/minúsculas?

Se maneja convirtiendo tanto el texto ingresado por el usuario como los valores del diccionario a **minúsculas con .lower()** antes de compararlos.

- ¿Cómo buscar texto parcial en nombres?

Para buscar texto parcial, se usa el operador in dentro de la comparación