

PROGRAMACIÓN I



Profesor Coordinador: Alberto Cortez

Comisión 1



Profesora de la comisión: Cinthia Rigoni



Tutor: Martín A. García

Trabajo Práctico Integrador:

Algoritmos de Búsqueda y Ordenamiento en Python

Estudiantes:

Hernán E. Bula - hernanbula@gmail.com

Rodrigo Arias - roarias299@gmail.com

Fecha de Entrega: 09/06/2025

0. Índice

1. Introducción
2. Marco teórico
3. Caso práctico
4. Metodología
5. Resultados obtenidos
6. Conclusiones
7. Bibliografía
8. Anexos (enlaces a repositorio y recursos)



1. Introducción

Los **algoritmos de búsqueda y ordenamiento** son pilares fundamentales en el desarrollo de software, especialmente en la gestión eficiente de datos. Este trabajo práctico aborda su implementación en Python, enfocándose en un caso de uso real: la administración de una lista de productos y de precios de un supermercado.

Se comprende que la búsqueda resulta esencial para localizar información de forma rápida en conjuntos de datos, una operación recurrente en la mayoría de los programas. A su vez, se advierte que el ordenamiento no solo organiza los datos bajo ciertos criterios (por ejemplo, de menor a mayor), sino que también permite mejorar notablemente la eficiencia de las búsquedas, habilitando el uso de algoritmos más veloces, como la búsqueda binaria. Al introducirse estos conceptos en el uso de estructuras de datos como arrays o listas, se

identifica que estos algoritmos constituyen herramientas básicas para optimizar la interacción con dichas estructuras.

La comparación entre métodos como la búsqueda lineal (más simple pero menos eficiente) y la búsqueda binaria (más rápida, aunque requiere listas ordenadas), así como entre distintos tipos de ordenamiento, permite reflexionar sobre la importancia de la eficiencia algorítmica. Se reconoce, entonces, que la elección adecuada de un algoritmo depende de factores como la naturaleza del problema, el tamaño del conjunto de datos y el estado de orden previo de la información.

Si bien existen funciones nativas en Python como *sorted()* o *list.sort()* que ya realizan y simplifican las tareas, siendo más limpio, rápido y legible, este trabajo prioriza la implementación manual de métodos para comprender la lógica subyacente, reforzando así competencias esenciales en programación. Aprender métodos como Selection Sort para entender la lógica de selección y posicionamiento (que simula cómo ordenamos objetos manualmente), Bubble Sort (para comparar cómo funcionan los intercambios adyacentes y cuya lógica es sencilla pero no recomendable para grandes volúmenes de datos), o también Merge Sort o Insertion Sort, permite comprender cómo funciona el ordenamiento y saber cuales son algoritmos más eficientes en cada caso. Como estudiantes, este contenido es clave porque brinda herramientas iniciales para comprender cómo desarrollar soluciones escalables y más eficientes en programación.

2. Marco Teórico

¿Qué son los algoritmos de búsqueda y ordenamiento?

En programación, la búsqueda y el ordenamiento son operaciones fundamentales para la gestión eficiente de la información (Programación I, 2025a; Tecnicatura, 2025c; González & Benítez, 2025). Son esenciales en sistemas cotidianos y afectan el rendimiento y la experiencia del usuario (Tecnicatura, 2025c).

La búsqueda consiste en localizar un elemento específico dentro de un conjunto de datos. El tamaño del conjunto de datos impacta significativamente el tiempo de búsqueda. La eficiencia de los algoritmos se mide con la notación $O(n)$ (Programación I, 2025a; Tecnicatura, 2025a, 2025c).

- Búsqueda lineal: es el algoritmo más simple, que recorre cada elemento secuencialmente. Es fácil de implementar pero lenta para grandes conjuntos de datos. No requiere que la lista esté ordenada. Su complejidad en el peor caso es $O(n)$ (Programación I, 2025a, 2025b; Tecnicatura, 2025a; 2025c).
- Búsqueda binaria: es un algoritmo altamente eficiente que requiere que el conjunto de datos esté previamente ordenado. Funciona dividiendo repetidamente la lista a la mitad. Su eficiencia radica en la eliminación sistemática de la mitad restante en cada paso. Su complejidad en el peor caso es $O(\log n)$, siendo exponencialmente más rápida que la búsqueda lineal para listas grandes

Por otro lado, el ordenamiento es el proceso de organizar datos según un criterio. Permite estructurar datos de manera eficiente. El principal beneficio del ordenamiento es permitir una búsqueda mucho más eficiente, especialmente utilizando la búsqueda binaria en datos ya ordenados (Programación I, 2025a, 2025b; Tecnicatura, 2025b, 2025c; González & Benítez, 2025). Existen varios algoritmos de ordenamiento:

- Bubble Sort compara e intercambia elementos adyacentes. Es simple, pero ineficiente para listas grandes. Su complejidad en el peor caso es $O(n^2)$.
- Insertion Sort construye la lista ordenada insertando elementos uno a uno en su posición correcta. Es eficiente para listas pequeñas o parcialmente ordenadas. Su complejidad en el peor caso es $O(n^2)$.
- Selection Sort encuentra el elemento más pequeño y lo coloca al inicio, repitiendo el proceso. Es más eficiente que Bubble Sort, pero sigue siendo lento para listas grandes. Su complejidad es $O(n^2)$.
- Quick Sort utiliza un enfoque de "divide y vencerás" seleccionando un pivote. Es mucho más rápido que Bubble/Selection Sort en la práctica para listas grandes. Su complejidad promedio es $O(n \log n)$.

Si bien en esta instancia se trabaja con algoritmos de búsqueda y ordenamiento más "artesanales", es importante aclarar, como se dijo en la introducción, que ya existen funciones en Python para realizar estas tareas de manera más sencilla. Las listas se pueden ordenar usando la función `sorted()`, que retorna una nueva lista ordenada, o el método `list.sort()`, que modifica la lista original (Codecademy Team, 2025). Ambas permiten usar el parámetro `key` con funciones lambda para definir criterios de ordenación personalizados, como ordenar por elementos específicos en listas anidadas o por múltiples atributos (Codecademy Team, 2025).

Es importante aclarar también, que la elección del algoritmo de búsqueda y ordenamiento adecuado es crucial y depende del contexto, el tamaño y el estado de los datos (ordenados o no). (Programación I, 2025a; Tecnicatura, 2025c; González & Benítez, 2025).

Estructuras de datos: listas o arrays y listas anidadas

Según Cimino (2024), para manejar grandes cantidades de datos relacionados de forma eficiente, se utilizan estructuras de datos. Una de esas estructuras, que se usará en este trabajo, son los arrays o vectores, referenciados por una única variable. Un array guarda datos en una sola dimensión. Por definición, los datos en un array deben ser generalmente del mismo tipo (aunque Python sea más flexible en este caso). Se accede a los elementos mediante un índice numérico que comienza en cero. El acceso aleatorio por índice es instantáneo. Una limitación clave es que los arrays tienen tamaño fijo; no se pueden añadir ni quitar elementos una vez creados, aunque se pueden modificar sus valores internos. Lenguajes de alto nivel (como el caso de Python) pueden dar la *apariencia* de redimensionar arrays creando nuevas estructuras subyacentes (Cimino, 2024).

Por último, para comprender el trabajo que se realizará, es importante comprender que las listas pueden contener otras listas, formando listas anidadas o "listas de listas", donde cada

elemento de una lista principal es a su vez otra lista. Estas sub-listas pueden representar registros o filas en una tabla, donde cada elemento dentro de la sub-lista corresponde a un punto de dato específico (ej. [ID_articulo, nombre_producto, precio]). Son útiles para representar datos estructurados como tablas o registros (Codecademy Team, 2025). Acceder a elementos en listas anidadas requiere usar múltiples índices, uno por cada “nivel” de profundidad (La Geekipedia De Ernesto, 2022). Por ej. `lista[indice_lista_exterior][indice_lista_interior]`.

3. Caso práctico

Objetivo:

Aplicar y analizar algoritmos de búsqueda y ordenamiento en Python, en el marco de la gestión de una lista de productos con precios.

Problema: Se dispone de una lista de productos de supermercado. Cada artículo está representado por una estructura de datos de lista que contiene ID de artículo, nombre de producto y precio. La información inicial se presenta como una lista de listas en Python (lista anidada), donde cada sub-lista representa un artículo. Sobre esta estructura se desarrolla un sistema interactivo que permita:

1. Gestionar artículos (CRUD:crear, leer, modificar, eliminar).
2. Buscar artículos por ID, nombre o precio.
3. Ordenar la lista por distintos criterios (ascendente/descendente).

Partes del algoritmo a desarrollar:

-Match-case para que el usuario elija una función:

- 1: imprimir lista de precios
- 2: agregar artículo
- 3: eliminar artículo
- 4: buscar un artículo (por ID, nombre o precio)
- 5: ordenar lista por elemento (por ID, nombre o precio) y ordenar creciente o decrecientemente.

-Definir las funciones de cada caso

-Elegir un método de búsqueda

-Elegir un método de ordenamiento

Ejemplo de lista:

```
lista_precios = [  
    [1, "Aceite Natura - 900 cc", 2090.50],  
    [2, "Tomate Arco en lata - 400 grs", 1190.00],  
    [3, "Mayonesa RI-K - 250 grs", 1090.50],  
    [4, "Aceite Patito - 1500 cc", 3590.50],  
    [5, "Azúcar Chango - 1 kg", 990.90],  
    [6, "Ensalada frutas lata - 850 grs", 1930.00],  
    [7, "Arroz Gallo Oro - 1 kg", 1850.00],  
    [8, "Fideos Matarazzo - 500 grs", 1680.50],  
    [9, "Leche entera La Serenísima - 1 lt", 2320.00],  
]
```

```
[10, "Yogur bebible Ilolay - 1 lt", 2850.75],  
[11, "Queso crema Casancrem - 290 grs", 3890.00],  
[12, "Jabón líquido Ala - 750 ml", 6920.30],  
[13, "Papel higiénico Elite - 4 rollos", 4120.00],  
[14, "Café La Morenita - 500 grs", 9150.00],  
[15, "Galletas Oreo - 117 grs", 1500.00],  
[16, "Chocolate Águila - 100 grs", 5010.00]  
]
```

Decisiones de diseño:

Para el diseño del algoritmo, se decidió organizar un código modular, usando funciones para cada una de los pequeños problemas que se debían resolver, bajo la premisa de “divide y vencerás”. Las funciones, a su vez, llaman a otras funciones más pequeñas. Así, la función principal es un menú Interactivo a partir de un match-case para gestionar opciones, mejorando la legibilidad y para que el usuario elija una función. La función principal opera como controlador del flujo, delegando las tareas a funciones secundarias. Esta estructura favorece la claridad del código y su mantenimiento.

Se usó el método de Selection Sort, uno de los más intuitivos para principiantes, ya que selecciona el mínimo/máximo y lo coloca en su posición correcta, además de que solo usa bucles for y comparaciones, adecuado a nuestro nivel de conocimientos actual.

Código en Python:

```
# TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA  
  
# Programación 1  
  
# Profesor Coordinador: Alberto Cortez  
# Profesora comisión 1: Prof. Cinthia Rigoni  
# Tutor: Prof. Martín A. García  
  
# Trabajo Práctico Integrador: Búsqueda y Ordenamiento  
  
# Estudiantes:  
Hernán Enrique Bula - hernanbula@gmail.com  
Rodrigo Arias - roarias299@gmail.com  
  
# Actividades  
  
'''  
Objetivo: Aplicar y analizar los algoritmos fundamentales de búsqueda y  
ordenamiento utilizando Python, en el contexto de la gestión de datos de una  
lista de precios de artículos de supermercado.  
  
Problema: Se dispone de una lista de artículos de supermercado. Cada  
artículo está representado por una estructura de datos de lista que contiene
```

ID de artículo, nombre de producto y precio. La información inicial se presenta como una lista de listas en Python (lista anidada), donde cada sub-lista representa un artículo.

Se propuso realizar un programa que permita al usuario incorporar nuevos artículos a la lista o borrar los existentes. Asimismo, mediante un algoritmo de búsqueda, el usuario puede buscar un artículo y mediante un algoritmo de ordenamiento, puede solicitar ordenarlos por alguno de sus elementos: número de artículo, nombre del artículo o precio.

Partes del algoritmo a desarrollar

- Un switch que el usuario elija una función:
 - 1: imprimir lista de precios
 - 2: agregar artículo
 - 3: eliminar artículo
 - 4: buscar un artículo (por ID, nombre o precio)
 - 5: ordenar lista por elemento (por ID, nombre o precio) y de manera ascendente o descendente.
- Definir las funciones de cada caso
- Elegir un método de búsqueda
- Elegir un método de ordenamiento

```
'''
```

```
# -----  
# Definición de funciones  
# -----
```

```
# Función para solicitar al usuario un dato, indicando el mensaje
```

```
def solicitar_dato(variable):  
    variable = input(f"\nIngrese {variable}: ")  
    return variable
```

```
# Función para solicitar al usuario número float validado y configurar el  
mensaje, mínimo y máximo
```

```
def leer_float_validado(mensaje, min = float("-Inf"), max = float("Inf")):  
    n = float(input(f"{mensaje}: "))  
    while n < min or n > max:  
        n = float(input(f"ERROR. {mensaje}: "))  
    return n
```

```
# Función para solicitar al usuario número entero validado y configurar el  
mensaje, mínimo y máximo
```

```
def leer_numero_validado(mensaje, min = float("-Inf"), max = float("Inf")):  
    n = int(input(f"{mensaje}: "))  
    while n < min or n > max:  
        n = int(input(f"ERROR. {mensaje}: "))  
    return n
```

```
# Función para solicitar al usuario la opción de menú entre 1, 2 o 3
```

```

def leer_opcion_menu():
    opcion = None
    valida = False
    while not valida:
        opcion = input("Seleccione opción (1/2/3): ")
        if opcion in {'1', '2', '3'}:
            valida = True
        else:
            print("ERROR: Debe ser 1, 2 o 3")
    return int(opcion)

# Función para imprimir la lista anidada en formato tabla. Para armar el
# formato tabla se solicitó asesoramiento de IA Deepseek
def imprimir_lista(lista):
    print("\n\n")
    print("-"*65)
    print(f"{'ID':<5} | {'PRODUCTO':<40} | {'PRECIO':>12}")
    print("-"*65)
    for id_prod, articulo, precio in lista:
        print(f"{'id_prod':<5} | {'articulo':<40} | ${precio:>12.2f}")

# Función de imprimir un artículo de una lista en formato tabla. Se adaptó
# la función imprimir_lista(lista)
def imprimir_producto(producto):
    print("-"*65)
    print(f"{'ID':<5} | {'PRODUCTO':<40} | {'PRECIO':>12}")
    print("-"*65)
    id_prod, articulo, precio = producto
    print(f"{'id_prod':<5} | {'articulo':<40} | ${precio:>12.2f}")

# Función para agregar artículo a la lista, asignando un ID correlativo y
# solicitando datos al usuario (nombre + precio)
def agregar_articulo():
    id_prod = len(lista_precios)+1
    articulo = input("\nIngrese el nombre del artículo: ")
    precio = leer_float_validado("\nIngrese el precio del artículo", min = 0,
max = float("Inf"))
    lista_precios.append([id_prod, articulo, precio])

# Función para eliminar un artículo de la lista, solicitando el número de ID
# al usuario
def eliminar_articulo():
    id_prod = leer_numero_validado("Ingrese el ID del artículo para
eliminarlo", 1, (len(lista_precios)))
    del lista_precios[id_prod-1]
    imprimir_lista(lista_precios)

```

```

# Función para modificar los datos de un artículo de la lista, solicitando
los datos el ID al usuario y los datos a modificar nombre y precio.
def modificar_articulo():
    id_prod = leer_numero_validado("\nIngrese el ID del artículo a
modificar", 1, (len(lista_precios)))
    articulo = input("\nIngrese el nombre nuevo del producto: ")
    precio = leer_float_validado("\nIngrese el precio nuevo del producto", 0)
    lista_precios[id_prod-1][1] = articulo
    lista_precios[id_prod-1][2] = precio
    imprimir_producto(lista_precios[id_prod-1])

# Función para buscar un artículo de la lista, solicitando el dato al
usuario para la búsqueda (id, nombre o precio)
def buscar_articulo():
    print("\nIngrese la opción que necesite \n 1: para buscar por ID del
producto \n 2: para buscar por nombre del producto \n 3: para buscar por
precio.\n")
    opcion = leer_opcion_menu()

    match opcion:
        case 1:
            id_prod = leer_numero_validado("\nIngrese ID del producto", 1,
(len(lista_precios)))
            imprimir_producto(lista_precios[id_prod-1])
        case 2:
            prod = solicitar_dato("nombre del producto") # falta mejorar la
funcion para validar dato (upper, solo letra, etc.)
            encontrado = False # Bandera para saber si encuentra el producto
o no
            for i in range(len(lista_precios)):
                if prod == lista_precios[i][1]:
                    imprimir_producto(lista_precios[i])
                    encontrado = True
            if not encontrado:
                print(f"\n{"-"*65}\nNO existe un producto con el nombre:
{prod}\n{"-"*65}")
                buscar_articulo()
        case 3:
            precio = leer_float_validado("\nIngrese precio del producto", 0)
            encontrado = False # Bandera para saber si encuentra el producto o
no
            for i in range(len(lista_precios)):
                if precio == lista_precios[i][2]:
                    imprimir_producto(lista_precios[i])
                    encontrado = True
            if not encontrado:
                print(f"No existe un producto con el precio: {precio}")

        case _:
            pass

```



```

# Función para ordenar los artículos de la lista, solicitando al usuario por
cual criterio: id, nombre o precio
def ordenar_lista():
    print("\nIngrese la opción que necesite \n 1: para ordenar por el ID del
producto \n 2: para ordenar alfabéticamente por nombre del producto \n 3:
para ordenar por precio.\n")
    indice = leer_opcion_menu() # Lee la opción validada entre 1,2 o 3
    orden = bool(input(f"\n{"-"*65}\nSi presiona enter, la lista se ordena de
menor a Mayor por defecto. \nSi quiere ordenarla de manera descendente,
ingrese cualquier tecla: "))
    match indice:
        case 1:
            ordenar_selection_sort(lista_precios,indice-1,orden)
        case 2:
            ordenar_selection_sort(lista_precios,indice-1,orden)
        case 3:
            ordenar_selection_sort(lista_precios,indice-1,orden)
        case _:
            pass

# Función con algoritmo de ordenamiento usando el método Selection Sort
adaptado a una lista (de productos) de listas.
# Para desarrollarlo trabajamos con IA Deepseek.
def ordenar_selection_sort(lista, indice_elemento, orden=False): # orden: Si
True (mayor a menor), si False (menor a mayor). Default: False
    n = len(lista)
    for i in range(n):
        # Inicializamos la posición del elemento extremo (mínimo o máximo)
        extremo = i
        for j in range(i+1, n):
            # Comparamos según el orden deseado
            if orden: # Orden descendente (mayor a menor)
                if lista[j][indice_elemento] >
lista[extremo][indice_elemento]:
                    extremo = j
            else: # Orden ascendente (menor a mayor)
                if lista[j][indice_elemento] <
lista[extremo][indice_elemento]:
                    extremo = j
            # Intercambiamos los elementos
            lista[i], lista[extremo] = lista[extremo], lista[i]

    imprimir_lista(lista)

# Función para ver opciones del menú principal:
def menu():
    menu = input("\n¿Quiere ingresar al menu de opciones? (S/N): ")
    if menu.upper() == "S":
        print(f"\n{"-"*65} \n{"-"*65} \n\n Ingrese la opción de la acción
que necesite realizar: \n\n A: imprimir lista de precios \n B: agregar
artículo \n C: eliminar artículo \n D: modificar artículo \n E: buscar un

```

```

artículo \n F: ordenar lista por elemento \n\n IMPORTANTE: Si ingresa
cualquier otra tecla y enter, sale de la aplicación.\n")
    opcion = solicitar_dato("opcion")
    return opcion.upper()

# Función principal para ejecutar funciones del menu de la aplicación y
realizar acciones sobre la lista según la opción seleccionada
# Controla el flujo principal de la aplicación.

def manipular_lista(opcion):

    match opcion: # Switch para que el usuario elija la opción:
        case "A": # Imprime lista de precios
            imprimir_lista(lista_precios)
            manipular_lista(menu())
        case "B": # Agrega artículo
            agregar_articulo()
            imprimir_lista(lista_precios)
            manipular_lista(menu())
        case "C": # Elimina artículo
            imprimir_lista(lista_precios)
            eliminar_articulo()
            manipular_lista(menu())
        case "D": # Modifica artículo existente
            imprimir_lista(lista_precios)
            modificar_articulo()
            manipular_lista(menu())
        case "E": # Busca un artículo
            buscar_articulo()
            manipular_lista(menu())
        case "F": # Ordena lista por elemento: id_prod, articulo, precio
            # Buscar y elegir el mejor método de ordenamiento para lista de
listas
            ordenar_lista()
            manipular_lista(menu())
        case _:
            print("Saliste de la aplicación.")

# -----
# Main
# -----
# LISTA INICIAL Y EJECUCIÓN
# -----

lista_precios = [
    [1, "Aceite Natura - 900 cc", 2090.50],
    [2, "Tomate Arco en lata - 400 grs", 1190.00],
    [3, "Mayonesa RI-K - 250 grs", 1090.50],
    [4, "Aceite Patito - 1500 cc", 3590.50],
    [5, "Azúcar Chango - 1 kg", 990.90],
    [6, "Ensalada frutas lata - 850 grs", 1930.00],

```

```
[7, "Arroz Gallo Oro - 1 kg", 1850.00],  
[8, "Fideos Matarazzo - 500 grs", 1680.50],  
[9, "Leche entera La Serenísima - 1 lt", 2320.00],  
[10, "Yogur bebible Ilolay - 1 lt", 2850.75],  
[11, "Queso crema Casancrem - 290 grs", 3890.00],  
[12, "Jabón líquido Ala - 750 ml", 6920.30],  
[13, "Papel higiénico Elite - 4 rollos", 4120.00],  
[14, "Café La Morenita - 500 grs", 9150.00],  
[15, "Galletas Oreo - 117 grs", 1500.00],  
[16, "Chocolate Águila - 100 grs", 5010.00]  
]  
  
manipular_lista(menu())
```

Validación del funcionamiento:

Se llevaron a cabo las siguientes validaciones de funcionamiento.

- A: imprimir lista de precios
- B: agregar artículo
- C: eliminar artículo
- D: modificar artículo
- E: buscar un artículo
- F: ordenar lista por elemento
- Validación de opciones (1/2/3)

A continuación dejamos **capturas** de pantalla de la **terminal** con las pruebas realizadas.

A: imprimir lista de precios

```
¿Quiere ingresar al menu de opciones? (S/N): s

-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: a

-----
ID | PRODUCTO | PRECIO
-----
1 | Aceite Natura - 900 cc | $ 2090.50
2 | Tomate Arco en lata - 400 grs | $ 1190.00
3 | Mayonesa RI-K - 250 grs | $ 1090.50
4 | Aceite Patito - 1500 cc | $ 3590.50
5 | Azúcar Chango - 1 kg | $ 990.90
6 | Ensalada frutas lata - 850 grs | $ 1930.00
7 | Arroz Gallo Oro - 1 kg | $ 1850.00
8 | Fideos Matarazzo - 500 grs | $ 1680.50
9 | Leche entera La Serenisima - 1 lt | $ 2320.00
10 | Yogur bebible Ilolay - 1 lt | $ 2850.75
11 | Queso crema Casancrem - 290 grs | $ 3890.00
12 | Jabón liquido Ala - 750 ml | $ 6920.30
13 | Papel higiénico Elite - 4 rollos | $ 4120.00
14 | Café La Morenita - 500 grs | $ 9150.00
15 | Galletas Oreo - 117 grs | $ 1500.00
16 | Chocolate Águila - 100 grs | $ 5010.00

¿Quiere ingresar al menu de opciones? (S/N): █
```

B: agregar artículo

```
¿Quiere ingresar al menu de opciones? (S/N): s

-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: B

Ingrese el nombre del artículo: Crema de leche Sancor - 200ml

Ingrese el precio del artículo: 3456.50

-----
ID | PRODUCTO | PRECIO
-----
1 | Aceite Natura - 900 cc | $ 2090.50
2 | Tomate Arco en lata - 400 grs | $ 1190.00
3 | Mayonesa RI-K - 250 grs | $ 1090.50
4 | Aceite Patito - 1500 cc | $ 3590.50
5 | Azúcar Chango - 1 kg | $ 990.90
6 | Ensalada frutas lata - 850 grs | $ 1930.00
7 | Arroz Gallo Oro - 1 kg | $ 1850.00
8 | Fideos Matarazzo - 500 grs | $ 1680.50
9 | Leche entera La Serenísima - 1 lt | $ 2320.00
10 | Yogur bebible Ilolay - 1 lt | $ 2850.75
11 | Queso crema Casancrem - 290 grs | $ 3890.00
12 | Jabón líquido Ala - 750 ml | $ 6920.30
13 | Papel higiénico Elite - 4 rollos | $ 4120.00
14 | Café La Morenita - 500 grs | $ 9150.00
15 | Galletas Oreo - 117 grs | $ 1500.00
16 | Chocolate Águila - 100 grs | $ 5010.00
17 | Crema de leche Sancor - 200ml | $ 3456.50

¿Quiere ingresar al menu de opciones? (S/N): █
```

C: eliminar artículo

```
¿Quiere ingresar al menu de opciones? (S/N): s

-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: c
```

ID	PRODUCTO	PRECIO
1	Aceite Natura - 900 cc	\$ 2090.50
2	Tomate Arco en lata - 400 grs	\$ 1190.00
3	Mayonesa RI-K - 250 grs	\$ 1090.50
4	Aceite Patito - 1500 cc	\$ 3590.50
5	Azúcar Chango - 1 kg	\$ 990.90
6	Ensalada frutas lata - 850 grs	\$ 1930.00
7	Arroz Gallo Oro - 1 kg	\$ 1850.00
8	Fideos Matarazzo - 500 grs	\$ 1680.50
9	Leche entera La Serenísima - 1 lt	\$ 2320.00
10	Yogur bebible Ilolay - 1 lt	\$ 2850.75
11	Queso crema Casancrem - 290 grs	\$ 3890.00
12	Jabón líquido Ala - 750 ml	\$ 6920.30
13	Papel higiénico Elite - 4 rollos	\$ 4120.00
14	Café La Morenita - 500 grs	\$ 9150.00
15	Galletas Oreo - 117 grs	\$ 1500.00
16	Chocolate Águila - 100 grs	\$ 5010.00
17	Crema de leche Sancor - 200ml	\$ 3456.50

Ingrese el ID del artículo para eliminarlo: 17

ID	PRODUCTO	PRECIO
1	Aceite Natura - 900 cc	\$ 2090.50
2	Tomate Arco en lata - 400 grs	\$ 1190.00
3	Mayonesa RI-K - 250 grs	\$ 1090.50
4	Aceite Patito - 1500 cc	\$ 3590.50
5	Azúcar Chango - 1 kg	\$ 990.90
6	Ensalada frutas lata - 850 grs	\$ 1930.00
7	Arroz Gallo Oro - 1 kg	\$ 1850.00
8	Fideos Matarazzo - 500 grs	\$ 1680.50
9	Leche entera La Serenísima - 1 lt	\$ 2320.00
10	Yogur bebible Ilolay - 1 lt	\$ 2850.75
11	Queso crema Casancrem - 290 grs	\$ 3890.00
12	Jabón líquido Ala - 750 ml	\$ 6920.30
13	Papel higiénico Elite - 4 rollos	\$ 4120.00
14	Café La Morenita - 500 grs	\$ 9150.00
15	Galletas Oreo - 117 grs	\$ 1500.00
16	Chocolate Águila - 100 grs	\$ 5010.00

¿Quiere ingresar al menu de opciones? (S/N):

D: modificar artículo

```
-----
Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: D

-----
ID | PRODUCTO | PRECIO
-----
1 | Aceite Natura - 900 cc | $ 2090.50
2 | Tomate Arco en lata - 400 grs | $ 1190.00
3 | Mayonesa RI-K - 250 grs | $ 1090.50
4 | Aceite Patito - 1500 cc | $ 3590.50
5 | Azúcar Chango - 1 kg | $ 990.90
6 | Ensalada frutas lata - 850 grs | $ 1930.00
7 | Arroz Gallo Oro - 1 kg | $ 1850.00
8 | Fideos Matarazzo - 500 grs | $ 1680.50
9 | Leche entera La Serenísima - 1 lt | $ 2320.00
10 | Yogur bebible Ilolay - 1 lt | $ 2850.75
11 | Queso crema Casancrem - 290 grs | $ 3890.00
12 | Jabón líquido Ala - 750 ml | $ 6920.30
13 | Papel higiénico Elite - 4 rollos | $ 4120.00
14 | Café La Morenita - 500 grs | $ 9150.00
15 | Galletas Oreo - 117 grs | $ 1500.00
16 | Chocolate Águila - 100 grs | $ 5010.00

Ingrese el ID del artículo a modificar: 1

Ingrese el nombre nuevo del producto: Aceite Patito - 1500 cc

Ingrese el precio nuevo del producto: 3200

-----
ID | PRODUCTO | PRECIO
-----
1 | Aceite Patito - 1500 cc | $ 3200.00

¿Quiere ingresar al menu de opciones? (S/N): █
```

E: buscar un artículo

- Busca artículo por ID

```
¿Quiere ingresar al menu de opciones? (S/N): s

-----
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: e

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Ingrese el número elegido: 1

Ingrese ID del producto: 12

-----
ID | PRODUCTO | PRECIO
-----
12 | Jabón líquido Ala - 750 ml | $ 6920.30

¿Quiere ingresar al menu de opciones? (S/N): █
```

- *Busca artículo por nombre*

```
¿Quiere ingresar al menu de opciones? (S/N): s
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: E

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Ingrese el número elegido: 2

Ingrese nombre del producto: Pan Integral

-----
NO existe un producto con el nombre: Pan Integral
-----

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Ingrese el número elegido: 2

Ingrese nombre del producto: Galletas Oreo - 117 grs
-----


| ID | PRODUCTO                | PRECIO     |
|----|-------------------------|------------|
| 15 | Galletas Oreo - 117 grs | \$ 1500.00 |


-----
¿Quiere ingresar al menu de opciones? (S/N): █
```

- *Busca artículo por precio*

```
¿Quiere ingresar al menu de opciones? (S/N): s
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: e

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Ingrese el número elegido: 3

Ingrese precio del producto: 1500
-----


| ID | PRODUCTO                | PRECIO     |
|----|-------------------------|------------|
| 15 | Galletas Oreo - 117 grs | \$ 1500.00 |


-----
¿Quiere ingresar al menu de opciones? (S/N): █
```


F: ordenar lista por elemento

- Ordena por ID (ascendente)

```
-----
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: F

Ingrese la opción que necesite
1: para ordenar por el ID del producto
2: para ordenar alfabéticamente por nombre del producto
3: para ordenar por precio.

Seleccione opción (1/2/3): 1

-----
Si presiona enter, la lista se ordena de menor a Mayor por defecto.
Si quiere ordenarla de manera descendente, ingrese cualquier tecla:

-----
ID      | PRODUCTO                                | PRECIO
-----|-----|-----
1       | Aceite Natura - 900 cc                  | $ 2090.50
2       | Tomate Arco en lata - 400 grs          | $ 1190.00
3       | Mayonesa RI-K - 250 grs                 | $ 1090.50
4       | Aceite Patito - 1500 cc                 | $ 3590.50
5       | Azúcar Chango - 1 kg                    | $ 990.90
6       | Ensalada frutas lata - 850 grs          | $ 1930.00
7       | Arroz Gallo Oro - 1 kg                   | $ 1850.00
8       | Fideos Matarazzo - 500 grs              | $ 1680.50
9       | Leche entera La Serenisima - 1 lt       | $ 2320.00
10      | Yogur bebible Ilolay - 1 lt             | $ 2850.75
11      | Queso crema Casancrem - 290 grs         | $ 3890.00
12      | Jabón líquido Ala - 750 ml               | $ 6920.30
13      | Papel higiénico Elite - 4 rollos         | $ 4120.00
14      | Café La Morenita - 500 grs              | $ 9150.00
15      | Galletas Oreo - 117 grs                 | $ 1500.00
16      | Chocolate Águila - 100 grs              | $ 5010.00

¿Quiere ingresar al menu de opciones? (S/N):
```

- Ordena por producto (ascendente)

```
-----
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: f

Ingrese la opción que necesite
1: para ordenar por el ID del producto
2: para ordenar alfabéticamente por nombre del producto
3: para ordenar por precio.

Seleccione opción (1/2/3): 2

-----
Si presiona enter, la lista se ordena de menor a Mayor por defecto.
Si quiere ordenarla de manera descendente, ingrese cualquier tecla:

-----
ID      | PRODUCTO                                     | PRECIO
-----
1  | Aceite Natura - 900 cc                      | $    2090.50
4  | Aceite Patito - 1500 cc                     | $    3590.50
7  | Arroz Gallo Oro - 1 kg                      | $    1850.00
5  | Azúcar Chango - 1 kg                       | $     990.90
14 | Café La Morenita - 500 grs                  | $    9150.00
16 | Chocolate Águila - 100 grs                 | $    5010.00
6  | Ensalada frutas lata - 850 grs              | $    1930.00
8  | Fideos Matarazzo - 500 grs                 | $    1680.50
15 | Galletas Oreo - 117 grs                    | $    1500.00
12 | Jabón líquido Ala - 750 ml                  | $    6920.30
9  | Leche entera La Serenísima - 1 lt           | $    2320.00
3  | Mayonesa RI-K - 250 grs                    | $    1090.50
13 | Papel higiénico Elite - 4 rollos            | $    4120.00
11 | Queso crema Casancrem - 290 grs            | $    3890.00
2  | Tomate Arco en lata - 400 grs              | $    1190.00
10 | Yogur bebible Ilolay - 1 lt                | $    2850.75

¿Quiere ingresar al menu de opciones? (S/N): █
```

- *Ordena por precio (descendente)*

```
-----
-----

Ingrese la opción de la acción que necesite realizar:
A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: F

Ingrese la opción que necesite
1: para ordenar por el ID del producto
2: para ordenar alfabéticamente por nombre del producto
3: para ordenar por precio.

Seleccione opción (1/2/3): 3

-----
Si presiona enter, la lista se ordena de menor a Mayor por defecto.
Si quiere ordenarla de manera descendente, ingrese cualquier tecla: 1

-----
ID | PRODUCTO | PRECIO
-----
14 | Café La Morenita - 500 grs | $ 9150.00
12 | Jabón líquido Ala - 750 ml | $ 6920.30
16 | Chocolate Águila - 100 grs | $ 5010.00
13 | Papel higiénico Elite - 4 rollos | $ 4120.00
11 | Queso crema Casancrem - 290 grs | $ 3890.00
4 | Aceite Patito - 1500 cc | $ 3590.50
10 | Yogur bebible Ilolay - 1 lt | $ 2850.75
9 | Leche entera La Serenisima - 1 lt | $ 2320.00
1 | Aceite Natura - 900 cc | $ 2090.50
6 | Ensalada frutas lata - 850 grs | $ 1930.00
7 | Arroz Gallo Oro - 1 kg | $ 1850.00
8 | Fideos Matarazzo - 500 grs | $ 1680.50
15 | Galletas Oreo - 117 grs | $ 1500.00
2 | Tomate Arco en lata - 400 grs | $ 1190.00
3 | Mayonesa RI-K - 250 grs | $ 1090.50
5 | Azúcar Chango - 1 kg | $ 990.90

¿Quiere ingresar al menu de opciones? (S/N): █
```

Validación de opciones

Sólo puede ingresar 1, 2 o 3. Si ingresa otra cosa da error y vuelve a solicitar el número.

```

¿Quiere ingresar al menu de opciones? (S/N): s
-----
-----

Ingrese la opción de la acción que necesite realizar:

A: imprimir lista de precios
B: agregar artículo
C: eliminar artículo
D: modificar artículo
E: buscar un artículo
F: ordenar lista por elemento

IMPORTANTE: Si ingresa cualquier otra tecla y enter, sale de la aplicación.

Ingrese opcion: e

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Seleccione opción (1/2/3): 6
ERROR: Debe ser 1, 2 o 3
Seleccione opción (1/2/3): h
ERROR: Debe ser 1, 2 o 3
Seleccione opción (1/2/3): 2

Ingrese nombre del producto: s

-----
-----
NO existe un producto con el nombre: s
-----
-----

Ingrese la opción que necesite
1: para buscar por ID del producto
2: para buscar por nombre del producto
3: para buscar por precio.

Seleccione opción (1/2/3): 

```

4. Metodología

Para realizar el trabajo se llevaron a cabo una serie de pasos que se detallarán a continuación:

- Investigación previa (fuentes utilizadas).

Para esta etapa primera, se realizó una revisión individual de todo el material disponible en el Aula Virtual sobre el tema de búsqueda y ordenamiento: apuntes de cátedra (Programación I, 2025a, 2025b), videos de la cátedra (Tecnatura, 2025a, 2025b, 2025c), material complementario (González & Benítez, 2025), manuales de Python (Python Software Foundation, 2025), tutoriales en video (Cimino, 2024; La Geekipedia De Ernesto, 2022) y material complementario (Codecademy Team, 2025; González & Benítez, 2025) para comprender los fundamentos de algoritmos de búsqueda y ordenamiento y cuestiones sobre listas anidadas que era el tema que se decidió trabajar.

- Elección del tema, el problema, el objetivo y el abordaje.

Una vez revisado individualmente el material teórico por cada miembro del grupo, se realizó una reunión virtual para poner en común lo leído, decidir el abordaje, plantear el objetivo, el problema a resolver, dividir tareas y establecer fechas claves en función de los plazos establecidos y de los tiempos de los integrantes.

Una vez decidido el problema se procedió a escribirlo de manera conjunta y plantearlo en su complejidad. Se charló en las diferentes posibilidades de abordarlo y se establecieron las funcionalidades y las partes que debía tener el algoritmo a desarrollar.

Posteriormente, se comenzó a escribir el marco teórico que encuadraría el enfoque y ayudaría a comprender mejor el tema elegido. Esta etapa fue útil para comprender más el tema y las posibilidades de abordaje del mismo.

- Etapas de diseño y prueba del código.

Una vez planteado el problema, se le presentó al tutor de la comisión para evaluar si estaba correctamente planteado y si era viable trabajarlo para el trabajo práctico integrador. Se hicieron los ajustes necesarios, se desarrolló la lista de ejemplo sobre la que se trabajaría y se comenzó a armar el código en módulos.

Lo primero que se desarrolló fue la función principal para ejecutar funciones del menú de la aplicación y realizar acciones sobre la lista según la opción seleccionada:

manipular_lista(opcion) es la función que controla el flujo principal de la aplicación.

Una vez desarrollada la misma, se procedió a definir la función ***menu()*** para elegir las opciones del menú principal que se anidan dentro de la función principal:

manipular_lista(menu()). Este menú llama a las otras funciones más pequeñas y específicas que solucionan las tareas más puntuales: *imprimir_lista(lista)*, *agregar_articulo()*, *eliminar_articulo()*, *modificar_articulo()*, *buscar_articulo()*, *ordenar_lista()* o salir del programa.

- Herramientas y recursos utilizados (IDE, librerías, control de versiones, etc.).

Herramientas y recursos utilizados

- Entorno de desarrollo: Visual Studio Code (versión 1.100.3) con extensión Python.
- Lenguaje y dependencias: Python 3.13.2.
- Librerías: No se requirieron librerías externas; solo funciones nativas de Python.
- Control de versiones: Git con repositorio en GitHub
- DeepSeek Chat: para optimizar el formato de tablas y validar el algoritmo de ordenamiento.

- Trabajo colaborativo.

Para realizar el trabajo se realizó reparto de tareas. En relación a la parte escrita del informe, una persona se dedicó a redactar el marco teórico, la metodología y la bibliografía. La otra el caso práctico, los resultados obtenidos y las conclusiones. Por otro lado, en relación al código, una persona se dedicó a escribir la estructura general del programa, la función principal *manipular_lista()* y el *menu()* y las funciones secundarias como *imprimir_lista(lista)*, *agregar_articulo()*, *eliminar_articulo()*, *modificar_articulo()* y *buscar_articulo()*

La otra persona se dedicó a escribir la función de *ordenar_lista()* y *ordenar_selection_sort()*, además de las funciones de validación como *solicitar_dato()*, *leer_float_validado()*, *leer_numero_validado()*, *leer_opcion_menu()*.

5. Resultados obtenidos

El desarrollo de este trabajo integrador permitió investigar, conocer, comprender y aplicar algoritmos de búsqueda y ordenamiento en un caso práctico concreto. Se pudo resolver, a partir de la estrategia de “divide y vencerás”, el problema propuesto sobre una lista de productos de un comercio y diferentes acciones sobre la mismas. Para lograrlo se subdividió

en pequeños problemas a resolver que, paso a paso, se fueron abordando en el código a partir de las diferentes funciones desarrolladas. Las diversas funcionalidades se van integrando, combinando y colaborando entre sí para resolver el problema general. Se lograron programar las diferentes aplicaciones imaginadas al comienzo del proceso: imprimir lista de precios, agregar artículo, eliminar artículo, buscar un artículo (por ID, nombre o precio), ordenar lista por elemento (ID, nombre o precio), de manera creciente o decreciente.

La parte más compleja del algoritmo fue la parte de ordenamiento, ya que implica mayor complejidad en la comprensión de la lógica. Se evaluaron todos los métodos propuestos desde la cátedra, decantándose por utilizar uno de los métodos más sencillos y que no tiene tan mal rendimiento en listas acotadas: *Selection Sort*. Esta decisión implicó la siguiente evaluación: *Bubble Sort* requiere entender intercambios adyacentes repetidos, lo que es un poco más confuso inicialmente, además de que ambos (*Selection* y *Bubble*) tienen $O(n^2)$ en complejidad temporal (ineficientes para listas grandes), lo cual no significaba un gran cambio en una lista pequeña (16 elementos). Asimismo, *Quick Sort* $O(n \log n)$ era mucho más eficiente para listas grandes (que no es el caso de este proyecto), pero su implementación recursiva y el concepto de "pivote" complicaba un poco su implementación en listas anidadas. En función de este análisis, se decantó por *Selection Sort* que selecciona el elemento mínimo (o máximo) y lo coloca en su posición correcta. Para ello se utilizan bucles for y comparaciones, conocimientos accesibles acordes al nivel de conocimientos actuales a programación 1.

Los casos de prueba realizados están presentados anteriormente en el apartado de validación del funcionamiento. Realizar esto permite saber que el código funciona, además de demostrar cómo y por qué funciona en distintos escenarios. Esto incluyó probar diversos ejemplos de entradas/salidas, usar casos límite y validar entradas.

Se encontraron algunos errores al validar dos o tres de las funciones secundarias: *solicitar_dato()*; *leer_numero_validado()*; *leer_float_validado()*. Funcionan correctamente, si se ingresan números, pero genera un error en caso de ingresar otro tipo de carácter no numérico. Se preveía intentar arreglar esto pero, por falta de tiempo, no se logró solucionar completamente. Sólo se pudo solucionar parcialmente en los menu de tres opciones a partir de la función *leer_opcion_menu()*.

```
def solicitar_dato(variable):
    variable = input(f"\nIngrese {variable}: ")
    return variable

# Función para solicitar al usuario número float validado y configurar el
mensaje, minimo y máximo
def leer_float_validado(mensaje, min = float("-Inf"), max = float("Inf")):
    n = float(input(f"{mensaje}: "))
    while n < min or n > max:
        n = float(input(f"ERROR. {mensaje}: "))
    return n
```

```

# Función para solicitar al usuario número entero validado y configurar el
mensaje, mínimo y máximo
def leer_numero_validado(mensaje, min = float("-Inf"), max = float("Inf")):
    n = int(input(f"{mensaje}: "))
    while n < min or n > max:
        n = int(input(f"ERROR. {mensaje}: "))
    return n

# Función para solicitar al usuario la opción de menu entre 1, 2 o 3
def leer_opcion_menu():
    opcion = None
    valida = False
    while not valida:
        opcion = input("Seleccione opción (1/2/3): ")
        if opcion in {'1', '2', '3'}:
            valida = True
        else:
            print("ERROR: Debe ser 1, 2 o 3")
    return int(opcion)

```

Otro aspecto que quedó pendiente es la evaluación de rendimiento. La idea original era que, una vez se terminaba con el desarrollo del algoritmo, probar diferentes métodos de búsqueda y ordenamiento, por un lado para comprenderlos mejor desde la práctica y, por otro, poder comparar el tiempo de ejecución entre algoritmos para sacar nuestras propias conclusiones respecto del rendimiento en el uso de los diferentes métodos aplicados a una lista anidada.

6. Conclusiones

El desarrollo de este trabajo integrador permitió, además de investigar y comprender teóricamente, poder aplicar prácticamente algoritmos de búsqueda y ordenamiento. Al mismo tiempo permitió aplicarlos de manera integrada con el resto de conocimientos adquiridos en la asignatura a lo largo del cuatrimestre: Estructuras Secuenciales, Trabajo Colaborativo, Estructuras Condicionales, Estructuras Repetitivas, Listas, Funciones y Recursividad. Esto permitió no sólo comprender la importancia de los conocimientos adquiridos en estos meses, sino también brindar la posibilidad real de solucionar un problema concreto a partir del desarrollo de un algoritmo mucho más complejo, modular e integrado que los realizados hasta ahora en los ejercicios más pequeños y concretos.

Es importante destacar el aprendizaje adquirido acerca de la relevancia que poseen los algoritmos de búsqueda y ordenamiento y su gran utilidad en programación porque son esenciales para la gestión eficiente de la información. Permiten encontrar elementos específicos (búsqueda) y organizar datos según un criterio, lo cual mejora significativamente la eficiencia y el tiempo de ejecución de los programas, especialmente con grandes volúmenes de datos. Esta eficiencia es crucial para el rendimiento y la experiencia del usuario. Además, facilitan la organización y estructuración de datos para un análisis más sencillo, son escalables y se aplican en una vasta gama de contextos, incluyendo bases de



datos, sistemas de archivos, aplicaciones web y científicas, siendo pilares clave en el desarrollo de software.

Como posibles mejoras o extensiones futuras, hay que mejorar las funciones que permiten validar datos de una manera más completa, evitando cualquier tipo de error en el código. A medida que se adquieran más conocimientos, pueden mejorarse y perfeccionarse estos aspectos. Es también importante estar abiertos a posibles devoluciones de cómo podrían mejorarse este código, ya que, si bien funciona y cumple con los objetivos, seguramente existen miles de maneras de mejorarlo y perfeccionarlo, sobre todo para hacerlo más eficiente, menos repetitivo y más limpio y sencillo.

7. Bibliografía

- Codecademy Team (2025, 31 de enero). *How to Sort Lists of Lists in Python with Examples*. <https://codecademy.com/article/how-to-sort-lists-of-lists-in-python-with-examples>
- González, S., & Benítez, M. (2025). *Algoritmos de Búsqueda y Ordenamiento. Trabajo modelo* [Documento PDF]. TUPaD, UTN.
- Programación I (2025a). *Búsqueda y Ordenamiento en Programación* [Apunte de cátedra PDF]. TUPaD, UTN.
- Programación I (2025b). *Implementación de Algoritmos de Búsqueda Binaria en Python* [Presentación de diapositivas]. TUPaD, UTN.
- Tecnicatura (2025a, 17 de febrero). *BÚSQUEDA* [Video Youtube]. TUPaD, UTN. <https://youtu.be/gJlQTq80llg>
- Tecnicatura (2025b, 17 de febrero). *ORDENAMIENTO* [Video Youtube]. TUPaD, UTN. <https://youtu.be/xntUhrhtLaw>
- Tecnicatura (2025c, 19 de mayo). *Introducción Búsqueda y Ordenamiento- Integrador* [Video Youtube]. TUPaD, UTN. https://youtu.be/u1QuRbx-_x4

Recursos complementarios

- Charly Cimino (2024, 11 de abril). *ARRAYS en PROGRAMACIÓN*  *La base necesaria*  [Video Youtube]. <https://youtu.be/OHwl0RO4bac>
- Codecademy Team (2025, 31 de enero). *How to Sort Lists of Lists in Python with Examples*. <https://www.codecademy.com/article/how-to-sort-lists-of-lists-in-python-with-examples>
- DeepSeek. <https://chat.deepseek.com/>
- Google for Education (2024a, 29 de agosto). *Clase de Python. Orden de Python*. <https://developers.google.com/edu/python/sorting?hl=es-419>
- Google for Education (2024b, 6 de noviembre). *Clase de Python. Listas de Python*. <https://developers.google.com/edu/python/lists?hl=es-419>

- La Geekipedia De Ernesto (2022, 21 de octubre). *Curso Python 3 desde cero #62, Listas anidadas* [Video Youtube]. <https://youtu.be/8tXEia5ZcCE>
- Python Software Foundation (2025). *documentación de Python - 3.13.3: Guía para principiantes: Listas*. <https://docs.python.org/es/3.13/tutorial/introduction.html#lists>



8. Anexos

- [Enlace al repositorio en GitHub >>](#)
- [Enlace al video de presentación >>](#)
- [Enlace a la presentación >>](#)
- [Enlace al código Python >>](#)
- [Enlace al código Python con las funciones documentadas >>](#)
- [Enlace al README.MD](#)