Tecnicatura en Programación

Trabajo Práctico de Integración

“Sistema de búsqueda y ordenamiento en Python”

Materia: Programación I

Docentes: Ariel Enferrel

Tutor: Martina Zabal

Integrantes:

- Carrion, Camila - carrioncamila790@gmail.com
- Cerrato Nahuel - nahuelcerrato@gmail.com

09 de Junio, 2025

ÍNDICE

1. INTRODUCCIÓN

El presente trabajo práctico tiene como objetivo desarrollar un programa en Python que simule el funcionamiento básico de una base de datos de productos, permitiendo la búsqueda por código y el ordenamiento de los registros por atributos clave.

A través de este ejercicio práctico, se abordan conceptos fundamentales de la programación estructurada y de la manipulación de estructuras de datos, como listas de diccionarios, algoritmos de búsqueda y técnicas de ordenamiento.

Este tipo de herramientas es común en sistemas de gestión de inventario, catálogos de productos o módulos administrativos simples, lo que brinda al estudiante una experiencia práctica alineada con escenarios del mundo real.

El proyecto fue desarrollado en equipo utilizando conocimientos adquiridos durante el cursado de la materia Programación I, integrando funciones, control de flujo, validación de entradas y diseño modular. Además, se aplicaron algoritmos como la búsqueda lineal para localizar productos específicos, y el clásico Bubble Sort para ordenar datos por criterios.

1.1 Objetivos del trabajo

- Implementar una base de datos de productos simulada mediante estructuras de datos en Python.
- Aplicar el algoritmo de búsqueda lineal para localizar productos por código.
- Utilizar el algoritmo Bubble Sort para ordenar productos por precio y por stock.
- Diseñar un menú interactivo que permita al usuario acceder a las distintas funcionalidades del sistema.
- Validar el correcto funcionamiento del sistema mediante pruebas con distintos conjuntos de datos.
- Integrar conocimientos teóricos con una aplicación práctica y didáctica para reforzar el aprendizaje de conceptos clave.

2. MARCO TEÓRICO:

El presente trabajo se enmarca en el estudio y aplicación de algoritmos de búsqueda y ordenamiento, conceptos fundamentales en el área de la programación y el manejo de datos. Estos algoritmos permiten organizar y acceder eficientemente a la información, y constituyen la base para numerosos sistemas informáticos, como catálogos, bases de datos, inventarios y motores de búsqueda. A continuación, se desarrollan los principales conceptos teóricos relacionados con esta temática.

2.1 Algoritmos de Búsqueda

Los algoritmos de búsqueda son procedimientos que permiten encontrar un elemento dentro de una estructura de datos. Su eficiencia depende del tamaño y tipo de estructura, así como del grado de orden en los datos.

Búsqueda Lineal: La búsqueda lineal (o secuencial) recorre la lista elemento por elemento, comparando cada uno con el valor buscado. Es simple y no requiere que la lista esté ordenada, aunque es poco eficiente en listas extensas.

- Ventajas: fácil de implementar; funciona en listas desordenadas.
- Desventajas: tiempo de ejecución proporcional al tamaño de la lista (O(n)).

Búsqueda Binaria: La búsqueda binaria es un algoritmo más eficiente, que requiere que los datos estén ordenados. Divide el espacio de búsqueda en mitades sucesivas, descartando en cada paso una mitad del conjunto.

- Ventajas: muy eficiente (O(log n)); útil para grandes volúmenes.
- Desventajas: requiere que los datos estén previamente ordenados.

Otros métodos (interpolación, hash, búsqueda en árbol): Existen métodos más avanzados, como la búsqueda por interpolación (que estima la posición del elemento) o el uso de funciones hash (búsqueda directa por clave), así como estructuras más complejas como árboles binarios. Estos métodos ofrecen gran eficiencia, pero requieren conocimientos adicionales y estructuras más sofisticadas.

2.2 Algoritmos de Ordenamiento

El ordenamiento de datos es el proceso de reacomodar elementos dentro de una colección según algún criterio (numérico, alfabético, por fecha, etc.). Esto permite mejorar la legibilidad, realizar comparaciones y facilitar otras operaciones como la búsqueda binaria o la clasificación. Dentro de estos algoritmos, se destacan:

Bubble Sort: Es uno de los algoritmos de ordenamiento más conocidos. Compara elementos adyacentes e intercambia sus posiciones si están en el orden incorrecto. El proceso se repite hasta que toda la lista queda ordenada.

- Ventajas: muy fácil de implementar; útil en contextos educativos.
- Desventajas: poco eficiente en listas grandes ($O(n^2)$).

Insertion Sort: Construye una lista ordenada uno a uno, insertando cada nuevo elemento en su lugar correspondiente. Es más eficiente que Bubble Sort en listas parcialmente ordenadas.

Quick Sort y Merge Sort: Estos algoritmos utilizan el enfoque “divide y vencerás” para dividir la lista en partes, ordenarlas y combinarlas. Tienen mejor eficiencia ($O(n \log n)$) y se usan en sistemas reales.

Timsort: Es el algoritmo por defecto que utiliza Python cuando se llama a la función sorted(). Es un híbrido de Merge Sort e Insertion Sort, altamente optimizado para listas reales.

2.3 Estructuras de Datos Simples

En el contexto de la programación con Python, una de las formas más accesibles para representar una “base de datos” es utilizar listas de diccionarios. Cada diccionario puede representar un objeto con múltiples atributos, como un producto con nombre, precio y stock.

Esta estructura permite almacenar y manipular información de manera flexible y comprensible para estudiantes que se inician en la programación, sin necesidad de utilizar bases de datos relacionales ni bibliotecas externas.

- Ventajas: sintaxis clara, fácil de recorrer y manipular.
- Desventajas: limitada eficiencia para grandes volúmenes de datos.

2.4 Importancia de la búsqueda y el ordenamiento en sistemas reales

La capacidad de buscar y ordenar datos correctamente es fundamental para el desarrollo de sistemas robustos, eficientes y escalables. Desde catálogos de productos hasta sistemas de gestión de usuarios o inventarios, los algoritmos de búsqueda y ordenamiento permiten:

- Reducir tiempos de respuesta
- Facilitar el acceso a la información
- Mejorar la experiencia del usuario
- Preparar datos para análisis posteriores

Por este motivo, el estudio y aplicación de estos algoritmos representa un paso clave en la formación técnica de programadores y desarrolladores de software.

3. CASO PRÁCTICO:

Con el objetivo de aplicar los conceptos teóricos desarrollados en el marco de este trabajo, se diseñó e implementó un programa en Python que simula una base de datos de productos. Esta aplicación permite realizar búsquedas por código de producto y ordenar la lista por precio o stock disponible, a través de un menú interactivo en consola.

El sistema fue desarrollado con herramientas básicas del lenguaje Python, sin utilizar bibliotecas externas, priorizando la comprensión del flujo lógico, el uso de estructuras de datos y la implementación de algoritmos fundamentales.

3.1 Estructura de la base de datos simulada

La base de datos fue representada mediante una lista de diccionarios, donde cada diccionario almacena la información de un producto. Los atributos utilizados para cada producto fueron los siguientes:

- CÓDIGO: un identificador único del producto (por ejemplo, "REM001")
- NOMBRE: descripción del producto (por ejemplo, "Remera Negra")
- PRECIO: precio en pesos argentinos (entero)
- STOCK: cantidad disponible. En algunos productos este valor es None si no se controla stock (como en planes de suscripción).

Este diseño sencillo permite recorrer, buscar y ordenar los elementos de forma flexible y didáctica, sin requerir conocimientos avanzados de bases de datos.

3.2 Funcionalidades implementadas

- Búsqueda de producto por código:

Se implementó la función busqueda_lineal(), que recorre la lista de productos uno por uno hasta encontrar coincidencia con el código ingresado por el usuario. Este método fue elegido por su simplicidad y porque no se requiere que la lista esté ordenada. Dado que la cantidad de productos es reducida y el código es único, la búsqueda lineal resulta efectiva y coherente con el propósito educativo del trabajo.

```python
# -------- FUNCIÓN DE BÚSQUEDA LINEAL --------
# Busca un producto por su código dentro de la lista
def busqueda_lineal(lista, codigo_buscado):
    for producto in lista:  # Recorremos todos los productos
        if producto["CODIGO"] == codigo_buscado:  # Si encontramos coincidencia
            return producto  # Lo devolvemos
    return None  # Si no se encuentra, devolvemos None
```

- Ordenamiento por precio y stock

El programa permite ordenar los productos por PRECIO o STOCK, en ambos casos de menor a mayor. Para lograrlo, se implementó el algoritmo Bubble Sort en una función genérica que recibe como parámetro el campo a ordenar. Antes de ordenar, se realiza una copia de la lista original para evitar modificar la base de datos original. También se contempla el caso de productos con valores None, que son ignorados o enviados al final.

```python
# -------- FUNCIÓN DE ORDENAMIENTO (BUBBLE SORT) --------
# Ordena la lista por el atributo que le pasemos (clave)
def bubble_sort(lista, clave):
    lista_ordenada = lista.copy()  # Hacemos una copia para no modificar la original
    n = len(lista_ordenada)  # Cantidad de elementos en la lista

    for i in range(n):  # Realizamos n pasadas
        for j in range(0, n - i - 1):  # Comparamos elementos consecutivos
            a = lista_ordenada[j][clave]  # Valor actual (por la clave indicada)
            b = lista_ordenada[j + 1][clave]  # Valor siguiente

            if a is None:
                continue  # Si el valor actual es None, lo salteamos

            # Si el siguiente es None o el actual es mayor, intercambiamos
            if b is None or a > b:
                lista_ordenada[j], lista_ordenada[j + 1] = lista_ordenada[j + 1], lista_ordenada[j]

    return lista_ordenada  # Devolvemos la lista ordenada
```

- Mostrar productos en pantalla

Para presentar los resultados al usuario, se diseñó una función mostrar_productos() que recorre una lista de productos e imprime sus campos principales de forma ordenada y legible.

```python
# -------- FUNCIÓN PARA MOSTRAR LOS PRODUCTOS --------
# Imprime cada producto de la lista en una línea
def mostrar_productos(lista):
    for p in lista:
        print(f"{p['CODIGO']} - {p['NOMBRE']} | ${p['PRECIO']} | STOCK: {p['STOCK']}")
```

3.3 Interfaz de usuario (menú interactivo)

El programa se ejecuta a través de un menú interactivo que se muestra en consola. Cada opción llama a una función distinta, y el sistema se mantiene activo hasta que el usuario selecciona “salir”.

Opciones del menú:

- Buscar producto por código
- Ordenar productos por PRECIO
- Ordenar productos por STOCK
- Salir

Este diseño de menú refuerza el uso de ciclos, condicionales, y modularización, promoviendo buenas prácticas en el desarrollo de programas interactivos.

3.4 Validación y manejo de datos nulos

Un aspecto particular de la implementación fue el manejo de productos con STOCK = None. Esto representa productos sin control de stock, como planes de gimnasio o servicios. Durante el ordenamiento, se consideró:

- Evitar errores al comparar con valores nulos (None)
- Ubicar estos productos al final de la lista para claridad visual
- Mostrar mensajes específicos cuando el stock no se controla

Este tipo de validación refleja casos reales en aplicaciones de gestión de productos o catálogos híbridos (productos físicos y digitales).

3.5 Estructura de datos utilizada

Para representar la base de datos de productos, se optó por utilizar una lista de diccionarios, una estructura flexible y accesible en Python que permite organizar información compleja de manera clara. Cada producto fue modelado como un diccionario con claves bien definidas: CÓDIGO, NOMBRE, PRECIO y STOCK. Esta elección permite almacenar múltiples atributos por producto y acceder a ellos fácilmente a través de funciones personalizadas.

Este enfoque resulta ideal en proyectos de nivel inicial o intermedio, donde no se requiere un sistema de base de datos formal como SQLite o PostgreSQL, y permite simular el funcionamiento de catálogos o inventarios.

3.6 Manejo de valores nulos (None)

Una particularidad de la base de datos fue la inclusión de productos sin control de stock, como los planes de gimnasio. En estos casos, el valor del campo STOCK fue establecido como None. Este detalle implicó usar validaciones específicas en varias partes del programa:

- Durante el ordenamiento por stock, se ignoraron temporalmente los valores None, o bien se reubicaron al final de la lista.
- En la búsqueda y visualización de productos, se indicó explícitamente cuando un producto no tenía control de stock.

Este tratamiento especial refleja un comportamiento realista y demuestra cómo anticipar situaciones que podrían causar errores si no se gestionan adecuadamente.

3.7 Menú interactivo como interfaz del sistema

La interacción con el usuario fue resuelta mediante un menú de opciones en consola, que se ejecuta dentro de un bucle while. Cada opción representa una funcionalidad del sistema y permite una navegación simple e intuitiva. Este tipo de diseño es especialmente útil en prácticas educativas, ya que permite:

- Organizar el flujo del programa en bloques lógicos
- Utilizar condicionales, funciones y bucles de forma integrada
- Simular un entorno de sistema real sin interfaz gráfica

3.9 Justificación de los algoritmos elegidos

La elección de búsqueda lineal y Bubble Sort se fundamenta en el contexto del trabajo:

Criterio	Búsqueda Lineal	Bubble Sort
Tamaño de la base de datos	Reducido	Reducido
Orden previo de la lista	No	No
Facilidad de implementación	Alta	Alta
Claridad didáctica	Muy clara	Muy clara
Eficiencia esperada	Aceptable para este caso	Suficiente para este caso

Aunque existen algoritmos más eficientes, como la búsqueda binaria o el ordenamiento rápido (Quick Sort), estos requerirían estructuras adicionales o una preparación previa de los datos. Por este motivo, se optó por algoritmos básicos pero funcionales, que permitieran enfocarse en la lógica, validación y presentación clara de resultados.

A continuación, adjuntamos el enlace al video realizado, donde se puede ver el desarrollo del trabajo práctico: [https://youtu.be/fUeUuRUoOpE?si=YqqgDWQjXBXI3EsX]

4. METODOLOGÍA UTILIZADA:

El desarrollo del presente trabajo integrador se llevó a cabo mediante una secuencia de etapas bien definidas, que permitieron avanzar desde la planificación inicial hasta la validación final del sistema. La metodología adoptada combinó investigación teórica, diseño modular de funciones, implementación en lenguaje Python y pruebas con distintos conjuntos de datos.

El enfoque seguido fue práctico y progresivo, permitiendo aplicar de forma efectiva los conocimientos adquiridos en la materia Programación I, y reflejando un modelo de trabajo colaborativo.

4.1 Etapas de desarrollo

Etapa 1: Investigación y planificación

Objetivo de la etapa: Definir el enfoque del proyecto, los algoritmos a implementar, y la estructura de datos más adecuada según el nivel y alcance del trabajo.

Tareas realizadas:

- Revisión de conceptos teóricos sobre algoritmos de búsqueda y ordenamiento.
- Comparación entre métodos disponibles en Python y algoritmos clásicos.
- Evaluación de estructuras de datos posibles para representar productos.
- División inicial de tareas entre integrantes del grupo.

Etapa 2: Diseño de la base de datos simulada

Tareas realizadas:

- Creación de una lista de diccionarios con productos reales (ropa deportiva, accesorios, planes de gimnasio).
- Inclusión de campos variables: CODIGO, NOMBRE, PRECIO, STOCK.
- Consideración de casos particulares como valores None para productos sin stock.

Decisiones clave: Simular una base de datos lo más cercana posible a un sistema de gestión real, manteniendo una estructura simple y comprensible.

Etapa 3: Implementación de algoritmos

Tareas realizadas:

- Programación de la función de búsqueda lineal por código.
- Desarrollo del algoritmo Bubble Sort para ordenar por precio y por stock.
- Validación de funcionamiento con productos que tenían valores nulos (None).

Se prioriza la claridad y funcionalidad por sobre la eficiencia algorítmica, para facilitar el aprendizaje y asegurar la comprensión del proceso lógico.

Etapa 4: Construcción del menú interactivo

Objetivo de la etapa: Garantizar una experiencia fluida y clara para el usuario, integrando las distintas funciones del sistema en un entorno controlado.

Tareas realizadas:

- Diseño de un menú de opciones con input del usuario.
- Asignación de cada funcionalidad a una opción del menú.
- Validación de entradas erróneas o fuera de rango.

Etapa 5: Pruebas y validación del sistema

Tareas realizadas:

- Pruebas individuales de cada función con diferentes entradas.
- Validación cruzada entre opciones del menú.
- Comprobación de salidas esperadas y casos límite (por ejemplo, producto inexistente, stock None, lista vacía).

Etapa 6: Documentación y presentación

Tareas realizadas:

- Redacción del informe integrador con estilo académico.
- Preparación de un guión explicativo para el video de presentación.
- Desarrollo de una guía de diapositivas como apoyo visual.

- Organización de roles y tiempos para grabar la presentación final.

Trabajo colaborativo:

Las tareas fueron distribuidas entre los integrantes del equipo según afinidades: desarrollo, documentación, revisión teórica y presentación oral. La coordinación se realizó mediante reuniones virtuales y documentos compartidos.

5. RESULTADOS:

A lo largo del desarrollo del presente trabajo integrador se logró implementar exitosamente un sistema funcional en Python, que simula una base de datos de productos y permite aplicar algoritmos de búsqueda y ordenamiento.

Los resultados fueron consistentes con los objetivos propuestos y permitieron comprobar el correcto funcionamiento de las funcionalidades principales, tanto en términos lógicos como de interacción con el usuario.

Funcionalidades implementadas correctamente

Búsqueda por código: Se validó que la función busqueda_lineal() localiza correctamente los productos existentes por su código único. También se verificó que el sistema respondiera con un mensaje adecuado en caso de códigos inexistentes.

Ordenamiento por precio y stock: La función bubble_sort() ordena correctamente la lista de productos de menor a mayor según el campo elegido. Los productos sin stock (None) se ubican al final de la lista, sin generar errores de comparación.

Visualización clara de los resultados: Se mostró correctamente la información de los productos a través de la función mostrar_productos(), lo que facilitó la interpretación de los resultados por parte del usuario.

Menú interactivo robusto: El sistema permitió al usuario seleccionar distintas opciones, procesarlas adecuadamente y responder con salidas coherentes, incluyendo validación de entradas incorrectas.

Pruebas realizadas: Se realizaron múltiples pruebas con diferentes entradas de usuario y combinaciones de productos para validar:

- La correcta respuesta a búsquedas válidas e inválidas.
- La estabilidad del ordenamiento incluso con valores nulos.
- El mantenimiento de la base de datos original sin alteraciones destructivas.
- El comportamiento del programa ante errores o entradas no contempladas.

Dificultades encontradas:

- Comparación con valores None: Durante la etapa de ordenamiento fue necesario implementar lógica adicional para evitar errores al comparar campos cuyo valor era None. Esto se resolvió excluyendo temporalmente esos casos de la comparación directa.
- Manejo de input del usuario: Se diseñó lógica defensiva para evitar que el programa se rompa ante entradas inválidas (como letras donde se espera un número).

Posibles mejoras del proyecto:

- Implementar búsqueda binaria en listas previamente ordenadas, para mejorar la eficiencia de las consultas.
- Agregar filtrado combinado, permitiendo buscar productos por múltiples campos (por ejemplo: por nombre y por stock).
- Ordenamiento ascendente o descendente según elección del usuario, para mejorar la experiencia de análisis de datos.
- Control de errores más robusto, incluyendo validación de tipos de datos, manejo de excepciones, y uso de pruebas automatizadas.

6. CONCLUSIÓN:

El desarrollo de este trabajo práctico permitió aplicar de forma concreta los conocimientos adquiridos sobre algoritmos de búsqueda y ordenamiento, estructuras de datos y lógica de programación.

A través de la implementación de un sistema funcional en Python, se logró simular una base de datos de productos con operaciones básicas de consulta y organización de información. Esta experiencia no solo reforzó los conceptos teóricos abordados durante el cursado, sino que también favoreció el trabajo colaborativo, la toma de decisiones técnicas y el diseño de soluciones orientadas al usuario.

El uso de algoritmos sencillos como la búsqueda lineal y Bubble Sort fue suficiente para resolver las necesidades del sistema, demostrando que incluso con herramientas básicas es posible construir aplicaciones funcionales, claras y útiles para resolver problemas concretos.

Aprendizajes personales:

- Se comprendió la diferencia entre algoritmos de búsqueda y su aplicabilidad según el tipo de datos.
- Se aprendió a manipular listas de diccionarios como estructuras representativas de bases de datos simples.
- Se reforzó la capacidad de diseñar programas modulares con funciones independientes y menús interactivos.
- Se valoró la importancia de validar correctamente los datos de entrada para evitar errores lógicos o de ejecución.
- Se experimentó el trabajo en equipo desde la planificación, implementación y documentación del proyecto.

Este trabajo nos permitió consolidar habilidades esenciales para la formación en programación, y nos brindó una visión más clara de cómo aplicar la lógica computacional para resolver problemas reales de forma estructurada, modular y efectiva.

7. BIBLIOGRAFÍA:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3.ª ed.). MIT Press. (Capítulos sobre algoritmos de búsqueda y ordenamiento).

Python Software Foundation. (2024). Documentación oficial de Python https://docs.python.org/es/3/

Khan Academy. (2024). Algoritmos de búsqueda y ordenamiento. https://es.khanacademy.org/computing/computer-science/algorithms

W3Schools. (2024). Python Data Structures – Lists and Dictionaries https://www.w3schools.com/python/

UTN – Facultad Regional San Nicolás. (2025). Material de Programación I – Plataforma Educativa. (Apuntes y recursos de apoyo proporcionados por la cátedra).

Real Python. (2023). Sorting Algorithms in Python https://realpython.com/sorting-algorithms-python/

GeeksforGeeks. (2023). Linear Search and Bubble Sort in Python https://www.geeksforgeeks.org/