“Evaluación del estado académico de un grupo de estudiantes utilizando algoritmos de búsqueda y ordenamiento en Python”

Alumnas

Antonela Noeli Menna - antonelanmenna@gmail.com

Sandra Martinez - sandymartinez.186@gmail.com

Materia: Programación I

Profesora:Julieta Trapé

Tutor: Ángel David López

Fecha de Entrega: 09-06-2025

Índice

1. INTRODUCCIÓN

El objetivo de este trabajo es el desarrollo de un programa en lenguaje **Python** que permita gestionar el estado académico de alumnos, calculando su promedio a partir de las notas obtenidas. Para llegar a ello, se aplican algoritmos de búsqueda y ordenamiento para operar con los datos obtenidos.

La meta principal es **elaborar una solución práctica a un problema real**, en este caso en el ámbito educativo.

La elección de este tema surge de la intención de aprender a **enlazar** temas como: **algoritmos**, **búsqueda y ordenamiento** y **estructura de datos** para así fortalecer las habilidades en el uso de Python, lenguaje que hemos aprendido a lo largo de la cursada, ya que el correcto manejo de la información es importante para garantizar eficiencia y funcionalidad.

2. MARCO TEÓRICO

Python es un lenguaje de scripting, es decir, se utiliza para crear scripts que permiten resolver tareas específicas. Se destaca por su sintaxis clara y por contar con muchas funciones integradas, lo que lo hace sencillo de aprender y usar.

Python soporta múltiples paradigmas de programación, incluyendo la programación orientada a objetos, imperativa y funcional, y posee una extensa biblioteca estándar que amplía sus capacidades para múltiples áreas (Lutz, 2013).

Estructuras de datos (listas)

Las listas son secuencias que se pueden modificar, ya que permiten cambiar, eliminar o agregar elementos. Se pueden guardar tantos valores como se necesiten, como en este caso las calificaciones de los alumnos e incluso es posible tener listas vacías. Se definen usando corchetes y sus elementos se separan con comas. como en este caso, las calificaciones de los alumnos.

Búsqueda y Ordenamiento

La **búsqueda** es una operación clave en programación que se usa para localizar un elemento dentro de un conjunto de datos. Es una tarea muy común en distintos tipos de aplicaciones, como bases de datos, sistemas de archivos y algoritmos de inteligencia artificial.

Puntos clave sobre los métodos de búsqueda:

- **Búsqueda lineal**: examina uno por uno todos los elementos de la lista hasta encontrar el que se busca.
  Búsqueda binaria: es más rápida, pero solo se aplica a listas ordenadas. Consiste en ir dividiendo el conjunto en mitades para reducir el rango de búsqueda.

Funcionamiento de la búsqueda binaria:

1. Se compara el valor del medio con el que se desea encontrar.
2. Si coinciden, se devuelve la posición del elemento.
3. Si el valor buscado es menor, se continúa buscando en la mitad izquierda.
4. Si es mayor, en la mitad derecha.

El **Ordenamiento**, por su parte, consiste en organizar los datos según un criterio, como de menor a mayor o en orden alfabético. Los algoritmos de ordenamiento son esenciales porque permiten estructurar la información de forma eficiente. Una vez ordenados, los datos pueden analizarse, buscarse y procesarse más fácilmente.

Algoritmos de ordenamiento más comunes

- **Bubble Sort**: compara elementos vecinos y los intercambia si están en el orden equivocado, repitiendo el proceso hasta que toda la lista esté ordenada.
- **Insertion Sort**: recorre la lista agregando cada elemento en la posición correcta dentro de una nueva lista ordenada.
- **Selection Sort**: busca el valor mínimo en cada pasada y lo ubica en su lugar definitivo, avanzando posición por posición.

3. CASO PRÁCTICO

Vamos a simular la situación real de algún docente que necesite organizar y acceder de manera sencilla a la información académica de los alumnos.

Implementamos un programa que permite registrar los datos de alumnos, ingresando sus notas (podrían ser de parciales), calcular el promedio de cada uno, ordenarlos según el valor de éste y poder buscar a los alumnos mediante distintos criterios (nombre, apellido o DNI).

Por ejemplo, si ingresamos tres alumnos con sus respectivas notas, el programa calcula sus promedios, los ordena de mayor a menor, y le permite al usuario buscar a cualquier alumno para conocer su promedio y estado académico, en este caso aprobado o desaprobado..

A continuación se muestra el código implementado:

```
# Calculamos el promedio de una lista de notas
def calcular_promedio(notas):
    return sum(notas) / len(notas)

# Cargamos los datos de los alumnos desde el teclado
def ingresar_alumnos():
```

```
    alumnos = []
    cantidad = int(input("¿Cuántos alumnos vas a ingresar?: "))

    for i in range(cantidad):
        print("\nIngresá los siguientes datos:")
        nombre = input("Nombre: ")
        apellido = input("Apellido: ")
        dni = input("DNI: ")
        notas = []

        for j in range(3):  # Pedimos 3 notas por alumno
            nota = float(input(f"Ingresá la nota del parcial n° {j + 1}: "))
            notas.append(nota)

        alumno = {
            "nombre": nombre,
            "apellido": apellido,
            "dni": dni,
            "notas": notas
        }
        alumnos.append(alumno)

    return alumnos


# BÚSQUEDA SECUENCIAL

def buscar_alumno(clave, lista_alumnos):
    for alumno in lista_alumnos:
        if (alumno["nombre"].lower() == clave.lower() or
            alumno["apellido"].lower() == clave.lower() or
            alumno["dni"] == clave):
            return alumno
    return None


# ORDENAMIENTO BURBUJA

def ordenar_por_promedio(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n - i - 1):
```

```
            if calcular_promedio(lista[j]["notas"]) < calcular_promedio(lista[j
+ 1]["notas"]):
                # Intercambia si el siguiente promedio es mayor
                lista[j], lista[j + 1] = lista[j + 1], lista[j]


# PROGRAMA PRINCIPAL

# 1. Ingreso de datos
alumnos = ingresar_alumnos()

# 2. Ordenamiento por promedio descendente
ordenar_por_promedio(alumnos)

# 3. Mostrar alumnos ordenados
print("\n📋 Lista de alumnos ordenada por promedio (mayor a menor):")
for alumno in alumnos:
    promedio = calcular_promedio(alumno["notas"])
    print(f"{alumno['apellido']}, {alumno['nombre']} - Promedio:
{promedio:.2f}")

# 4. Búsqueda interactiva
while True:
    print("\n🔍 Búsqueda de alumno")
    clave = input("Ingresá nombre, apellido o DNI (o escribí 'salir' para
terminar): ")
    if clave.lower() == "salir":
        break

    resultado = buscar_alumno(clave, alumnos)

    if resultado:
        promedio = calcular_promedio(resultado["notas"])
        estado = "APROBADO ✅" if promedio >= 6 else "DESAPROBADO ❌"
        print(f"\n✅ Alumno encontrado:")
        print(f"Nombre: {resultado['nombre']} {resultado['apellido']}")
        print(f"DNI: {resultado['dni']}")
        print(f"Notas: {resultado['notas']}")
        print(f"Promedio: {promedio:.2f}")
        print(f"Estado: {estado}")
    else:
          print("❌ Alumno no encontrado.")
```

4. METODOLOGÍA UTILIZADA

Para resolverlo seguimos los siguientes pasos:

Identificamos la necesidad de gestionar alumnos y sus notas, y definimos las funcionalidades básicas requeridas.

Diseñamos el programa que incluye la carga de datos, el cálculo de promedios, el ordenamiento y la búsqueda.

Implementación en Python:
Utilizamos:

-Listas y diccionarios para almacenar los datos.
-Búsqueda secuencial
-Bubble Sort para ordenar los alumnos por su promedio, de mayor a menor.

Pruebas: se ejecutó el programa con distintos valores para verificar el correcto funcionamiento del código

5. RESULTADOS OBTENIDOS

Se logró un programa funcional que permite:

>>>Cargar múltiples alumnos con sus tres notas.
>>>Calcular correctamente el promedio de cada uno.
>>>Ordenar la lista de alumnos de forma descendente según el promedio.
>>>Buscar alumnos por nombre, apellido o DNI, mostrando toda su información y estado académico.
>>>La búsqueda secuencial fue efectiva para encontrar rápido elementos en una lista corta.

6. CONCLUSIONES

El desarrollo de este trabajo nos permitió aplicar algunos de los conceptos teóricos de búsqueda y ordenamiento. Implementamos la búsqueda secuencial para localizar alumnos dentro de una lista, aunque sabemos que esto fue eficiente ya que se trataba de una cantidad pequeña de datos. También utilizamos el algoritmo de ordenamiento burbuja para organizar a los alumnos por promedio, esto nos facilitó la visualización de los resultados de mayor o mejor resultado, a menor.

Pudimos almacenar y manejar la información de forma clara con el uso de estructuras como listas y diccionarios. Y por último organizar el código en funciones mejoró su legibilidad lo que nos permitió separar cada tarea específica.

7. BIBLIOGRAFÍA

- **EBAC. (s.f.). *Cómo aprender Python*.** Recuperado de **https://ebac.mx/blog/como-aprender-python**
- **Python Software Foundation. (2024). *Documentación oficial de Python 3***. Recuperado de **https://docs.python.org/3/**
- **Unidad 5 - Actividad II. (s.f.). *Introducción a las listas de Python*** [Material de lectura].
- Material de Lectura - Integración Búsqueda y Ordenamiento. (s.f.). ***Teoría sobre Búsqueda y Ordenamiento*.**
- Material de Apoyo *Ejemplo de Búsquedas* [Presentación PPT]
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). ***Introduction to Algorithms* (3.ª ed.).** MIT Press.
- Knuth, D. E. (1997). ***The Art of Computer Programming: Volume 3 – Sorting and Searching*** (2.ª ed.). Addison-Wesley.
- Downey, A. (2015). ***Think Python: How to Think Like a Computer Scientist*** (2.ª ed.). O'Reilly Media.
- **OpenAI. (2025). *ChatGPT* (versión GPT-4).** Recuperado de **https://chat.openai.com**

8. ANEXOS

Anexo – Uso de herramientas complementarias

1. **ChatGPT (OpenAI, 2025) fue utilizado como recurso de apoyo durante el desarrollo del trabajo para:**

- Aclarar dudas conceptuales sobre temas como listas, algoritmos de búsqueda y ordenamiento.
- Reformular y parafrasear textos del trabajo, manteniendo el sentido original pero con mayor claridad y fluidez utilizando sinónimos .
- Verificar la comprensión de contenidos del material de lectura.

El uso de esta herramienta se enmarca dentro de un enfoque de aprendizaje autónomo, como complemento al material brindado en la cursada.

2. Adjuntamos imágenes en la PPT obtenidas de diferentes sitios:

- https://www.youtube.com/watch?v=8hVuD0ng364
- https://medium.com/@mise/algoritmos-de-b%C3%BAsqueda-y-ordenamiento-7116bcea03d0
- **Nisfari, H. M. (2022).** *Sorting Algorithm: Bubble Sort, Selection Sort, Insertion Sort*. Medium.