Trabajo Practico de Matemática: Semana de Integración 1

Integrantes:

- Lorenzo Blanco.
- Flavio Bravo.
- Tiziano Caamaño.
- Ignacio Cabrera.
- Matías Boyer.

Proceso de Selección del ejercicio:

Se realizo una encuesta grupal en la que decidimos que ejercicio llevaríamos a cabo como propuesta. Elegimos de manera democrática 2/3 ejercicios a realizar de forma individual a manera de comprensión del proyecto a realizar.

Mediante encuentros virtuales hacemos una puesta en común del ejercicio elegido para la entrega, cada integrante aporta ideas propias y distintas gracias a la amplitud de las soluciones propuestas. Procedimos a armar una estructura de proyecto en base a “Lo mejor” de cada proyecto individual. Una vez finalizado el programa adjuntamos la documentación requerida para llevar a cabo el trabajo practico (capturas de la utilización de IA, material propuesto por la catedra, etc.).

Para la grabación del video requerido dividimos el proyecto en 2 bloques principales:

1. **Explicación del programa realizado:**

Se detalla cual es el objetivo del programa y se explica cada segmento de la codificación, que representa cada instrucción / modulo del programa.

2. **Explicación de conceptos matemáticos aplicados:**

Se recopila la documentación utilizada y se hace una breve explicación de los conceptos matemáticos aplicados necesarios para entender la funcionalidad del programa.

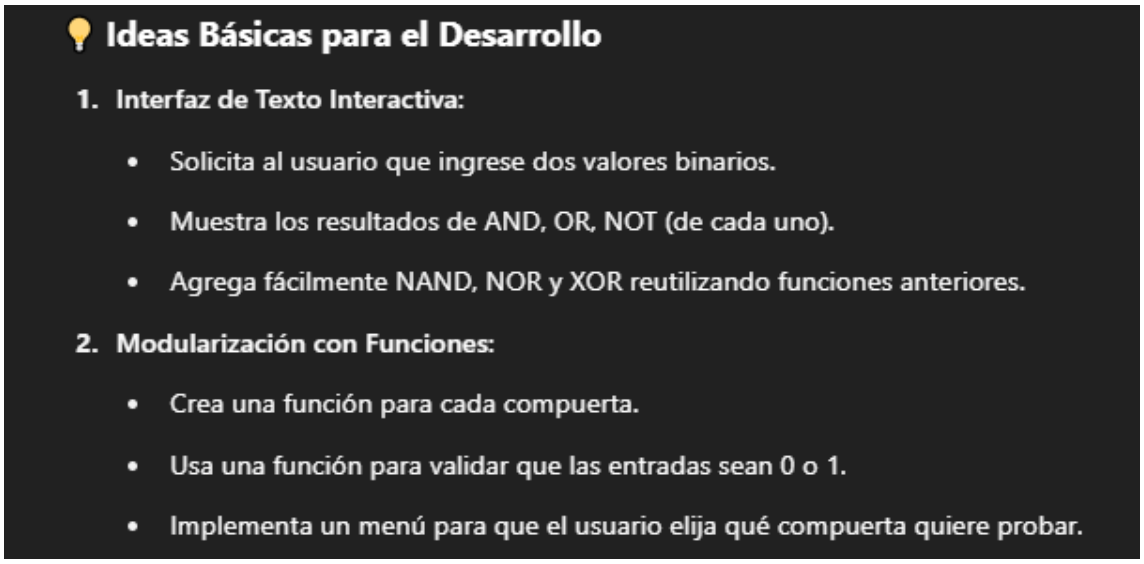
Documentación Utilizada para el desarrollo del programa:

El Álgebra de Boole es la base fundamental de la lógica digital, utilizada para representar y procesar información binaria (0 y 1) mediante reglas lógicas. Es esencial en la computación, ya que permite comprender cómo las computadoras toman decisiones lógicas a través de compuertas como AND, OR y NOT. Estas compuertas son construidas a nivel físico con transistores que actúan como interruptores, y a nivel teórico se modelan con operadores lógicos y tablas de verdad que determinan los resultados de distintas combinaciones de entrada.

El programa en Python pone en práctica estos conceptos simulando el funcionamiento de compuertas lógicas básicas. A través de funciones definidas para AND, OR, NOT, XOR, NAND y NOR, el código permite al usuario ingresar dos valores binarios (0 o 1) y obtener como resultado las salidas correspondientes a cada operación lógica. De esta manera, se crea un entorno de prueba que permite visualizar el comportamiento de circuitos lógicos en forma interactiva y comprensible.

Capturas de la utilizacion de la inteligencia artificial:

Ideas básicas para el desarrollo del ejercicio:



Ideas para el proceso de codificación:

- **Define el objetivo:** Simular compuertas lógicas básicas (AND, OR, NOT) y posiblemente extendidas (NAND, NOR, XOR).
- **Identifica entradas y salidas:**
  - Entradas: dos números binarios (o uno, para NOT).
  - Salidas: resultado de las operaciones lógicas.
- **Diseña una estructura modular:** Una función por compuerta, para mantener el código organizado.

- Crea funciones simples para cada compuerta:

```python
def compuerta_AND(a, b):
    return a * b
```

- Agrega comentarios que expliquen cada línea (especialmente útil si es para estudiantes o principiantes).
- Usa nombres de variables claros (`a` , `b` , `entrada1` , `entrada2`).

- Usa funciones para validar que el usuario solo introduzca 0 o 1:

```python
def validar_binario():
    while True:
        valor = input("Ingrese 0 o 1: ")
        if valor in ['0', '1']:
            return int(valor)
        print("Valor inválido.")
```

- También podés implementar un menú para que el usuario elija qué compuerta quiere probar.

- Pedí los valores de entrada.
- Llamá a las funciones de las compuertas y mostrales al usuario los resultados.
- Si vas a incluir todas las compuertas, ordená bien las salidas con `print()` para que el usuario entienda.

- Verificá que los nombres de variables y funciones sean coherentes.
- Comentá adecuadamente todo lo importante (pero no cada línea trivial).

Ideas para probar la funcionalidad del programa:

✅ 1. Pruebas Manuales (desde consola o interfaz)

Estas pruebas las hace el usuario escribiendo directamente en el programa.

- **Prueba de cada combinación posible:**
  - Para compuertas de dos entradas (AND, OR, NAND, etc.), probar con:
    - `(0, 0)`
    - `(0, 1)`
    - `(1, 0)`
    - `(1, 1)`
  - Para NOT (una sola entrada):
    - `0` → debería dar `1`
    - `1` → debería dar `0`
- **Verificar que los resultados coincidan con la tabla de verdad estándar.**