Agustin Martinez:

- Presentación y carga de los DNIs.
    - Generación de conjuntos únicos de dígitos por persona.
- Funciones implementadas:
    - union_conjuntos
    - interseccion_conjuntos
    - diferencia_entre_pares
    - diferencia_simetrica

Martìn Molina:

- Función contar_frecuencia_dni: cuenta repeticiones de dígitos.
- Función suma_digitos_dni: suma total de cada DNI.
- Función digitos_compartidos: detecta dígitos presentes en todos los DNIs.
- Función diversidad_numerica_alta: mide diversidad de los conjuntos.

Andrés Meshler:

- Diccionario con año de nacimiento por integrante.
- Conteo de años pares e impares.
- Clasificación: Grupo 'Z'(previo a 2000) vs 'Old School'.
- Función es_bisiesto: chequeo de años bisiestos.

Bruno mele:

- Cálculo de edad actual en base al año de nacimiento.
- Producto cartesiano: relaciona año ↔ edad.
- Impresión y análisis de resultados.

Agustin Martilotta:

- Realizar las 4 operaciones solicitadas (unión, intersección, diferencia (entre pares) y diferencia simétrica) con los DNI’s de los integrantes del grupo
- Diagrama de Venn.

Ayelen Masseroni:

- Después de plantear las expresiones lógicas realizadas en forma matemática, se implementaron mediante código en Python para verificar automáticamente las condiciones y obtener los resultados.
- Clasificación del grupo como “Old School”: Verificar que todos los integrantes nacieron antes del año 2000, por lo que el grupo cumple la condición y se lo considera ‘Old School’.
- Comparación de suma de dígitos de los DNI: Calcular la suma de los dígitos de cada DNI y que al menos dos integrantes tengan el mismo total, cumpliendo con la condición planteada.

- Compatibilidad entre conjuntos: Comprobar que los conjuntos de los dígitos únicos de los DNI de A y B tengan tres o más elementos en común, lo cual los define como altamente compatibles.
- Dígito universal: Analizar cual dígito está presente en todos los conjuntos de los DNIs, por lo que el numero 6 re repite en todos, por ende, es el digito universal

Relación entre las expresiones lógicas y el código implementado

En el presente trabajo se llevó a cabo un análisis y procesamiento de los números de Documento Nacional de Identidad (DNI) de un grupo de alumnos, abordando diferentes operaciones mediante conjuntos de dígitos y funciones de lógica matemática. Cada operación lógica tiene su representación directa en el código implementado en Python, estableciendo así una correspondencia clara entre teoría y práctica.

Unión (A ∪ B)

La unión entre dos conjuntos representa todos los elementos que pertenecen al conjunto A, al conjunto B, o a ambos. En el código, esta operación se implementa mediante la función union_conjuntos, que construye una lista con todos los dígitos únicos presentes en los dos DNIs comparados, evitando repeticiones. Por ejemplo, la unión de los conjuntos de Molina y Mele resulta en {1, 2, 3, 4, 5, 6, 7, 9}, lo cual se refleja correctamente en la salida del programa.

Intersección (A ∩ B)

La intersección identifica los elementos que son comunes entre dos conjuntos. Esto se implementa con la función interseccion_conjuntos, que recorre ambos conjuntos y guarda únicamente los dígitos compartidos. En el caso de Molina y Martinez, la intersección {3, 4, 6, 7, 9} muestra que estos alumnos comparten varios dígitos en sus respectivos DNIs.

Diferencia (A - B)

Esta operación devuelve los elementos que están en el conjunto A pero no en B. En el código, se utiliza diferencia_entre_pares, una función que compara ambos conjuntos y retiene solo los dígitos exclusivos del primero. Por ejemplo, al comparar a Molina con Meshler, la diferencia {1, 9} indica los dígitos únicos del DNI de Molina.

Diferencia Simétrica (A Δ B)

La diferencia simétrica obtiene los elementos que están en A o en B, pero no en ambos. El código lo refleja en la función diferencia_simetrica, que une ambas diferencias unidireccionales evitando repeticiones. Así, la diferencia simétrica entre Molina y Martilotta da como resultado {0, 1, 2, 3, 5, 7, 8}, reflejando fielmente el concepto teórico.