**Curso**  
Tecnicatura en Programación  
**Alumnos**  
Barroso, Nikolas  
Berger, Natasha  
Huarcaya, Ivan  
**Materia**  
Matematica  
**Docentes**  
Prof. Carina Jovanovich – Tutor Daniel Mosqueda

Temas

Conjuntos, lógica. Estructuras condicionales, repetitivas y funciones

MAtematica y programacion

Trabajo integrador 2

Contenido

[1. Introducción 2](#_Toc200660739)

[2. Parte 1 Desarrollo Matemático 2](#_Toc200660740)

[3. Parte 2 con Código 4](#_Toc200660741)

## Introducción

Se parte de una base matemática (teoría de conjuntos y lógica) para luego trasladar sus conceptos a un entorno computacional a través del lenguaje Python. Esta vinculación permite comprender cómo ideas abstractas pueden representarse mediante estructuras lógicas y algoritmos.

## Parte 1 – Desarrollo Matemático

* 1. **Elemento y Pertenencia**

Un elemento es cada uno de los objetos que forman parte de un conjunto. La pertenencia se denota con el símbolo ∈. Por ejemplo, si A = {3, 5, 7}, entonces 5 ∈ A. En este trabajo, los elementos de los conjuntos son los dígitos únicos de los DNIs de los integrantes.

* 1. **Igualdad e Inclusión de Conjuntos**

Dos conjuntos son iguales si tienen exactamente los mismos elementos: A = B. Un conjunto A es subconjunto de B si todos los elementos de A también pertenecen a B (A ⊆ B). En este trabajo, al comparar los conjuntos de dígitos se puede observar si algún conjunto es subconjunto de otro.

* 1. **Conjunto Universal y Conjunto Vacío**

El conjunto universal (U) contiene todos los elementos posibles en un contexto dado. En nuestro caso, podría ser U = {0,1,2,3,4,5,6,7,8,9}. El conjunto vacío (∅) es aquel que no contiene ningún elemento. Por ejemplo, si dos conjuntos no comparten dígitos, su intersección será el conjunto vacío.

* 1. **Operaciones con Conjuntos**

Las operaciones permiten combinar o comparar conjuntos de diferentes maneras:

• Unión (A ∪ B): Todos los elementos que están en A o B.

• Intersección (A ∩ B): Elementos comunes a A y B.

• Diferencia (A − B): Elementos en A que no están en B.

• Diferencia Simétrica (A Δ B): Elementos que están en A o en B, pero no en ambos.

Estas operaciones se aplicaron entre los conjuntos formados a partir de los DNIs.

* 1. **Diagramas de Venn**

Los diagramas de Venn permiten representar gráficamente las relaciones entre conjuntos. En este trabajo se representan las operaciones entre los tres conjuntos generados, mostrando visualmente la unión, intersección y diferencias.

* 1. **Producto Cartesiano**

Dado dos conjuntos A y B, el producto cartesiano A × B es el conjunto de todos los pares ordenados (a, b) con a ∈ A y b ∈ B. En la parte 2 del trabajo, se utilizó el producto cartesiano entre los años de nacimiento y las edades actuales.

* 1. **Relaciones Binarias y Propiedades**

Una relación binaria es un subconjunto del producto cartesiano entre dos conjuntos. Tiene un dominio (elementos del primer conjunto que se relacionan) y una imagen (elementos del segundo conjunto relacionados). Las propiedades de las relaciones incluyen:

• Reflexiva: todo elemento se relaciona consigo mismo.

• Simétrica: si (a, b) pertenece, entonces (b, a) también.

• Transitiva: si (a, b) y (b, c) pertenecen, entonces (a, c) también.

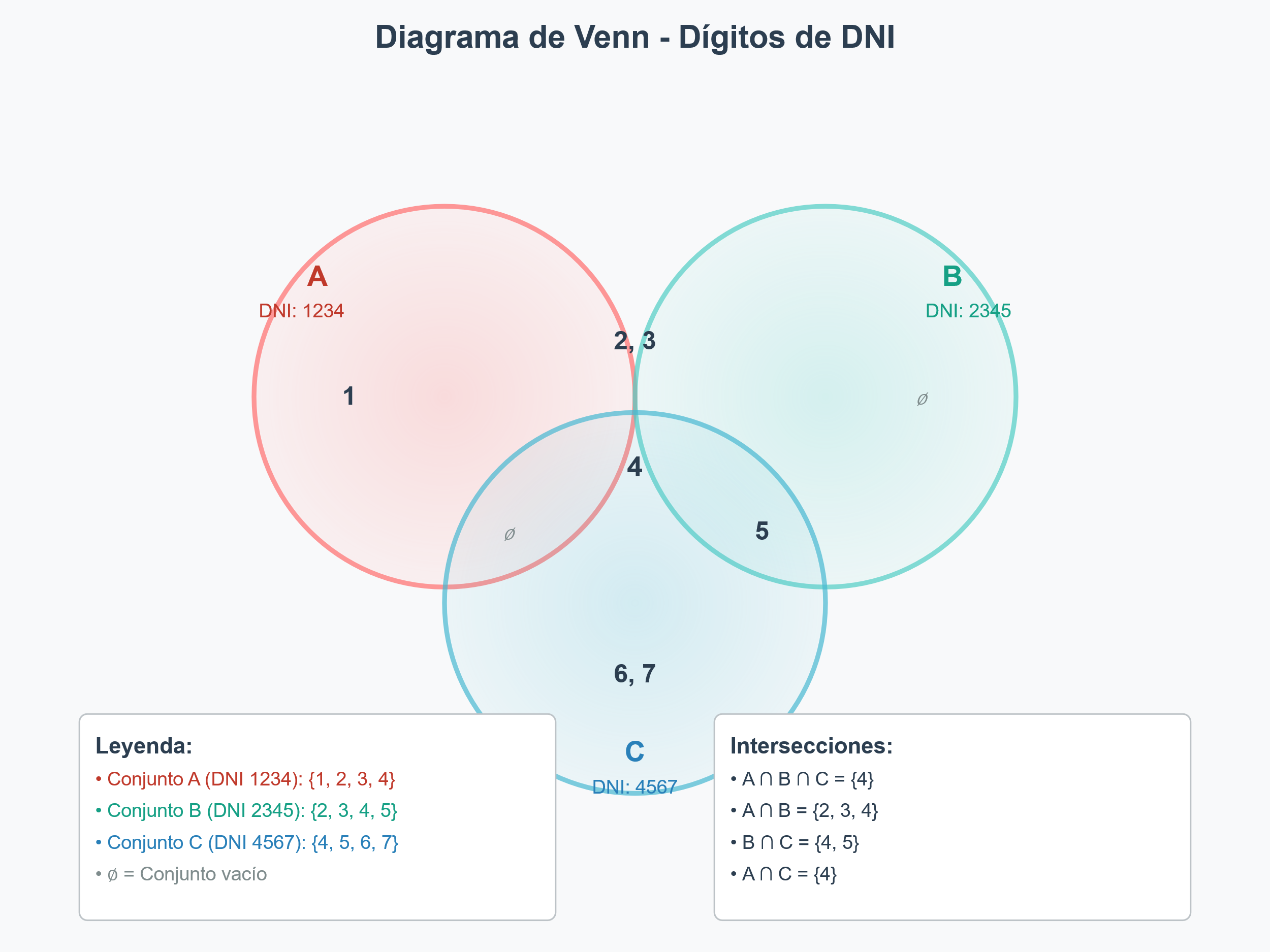
• Antisimétrica: si (a, b) y (b, a) pertenecen, entonces a = b.

Estas propiedades pueden analizarse sobre relaciones definidas entre los conjuntos de dígitos o entre años y edades.

* 1. **Visualizaciones: Diagramas y Tablas de Verdad**

Diagrama de Venn de los Dígitos de los DNI

El siguiente diagrama representa la intersección y unión entre los conjuntos formados por los dígitos de tres DNIs distintos.



* 1. **Tabla de Verdad de la Proposición (p ∧ q) → r**

La tabla de verdad muestra los posibles valores de verdad de la proposición compuesta (p ∧ q) → r, donde p, q y r representan propiedades relacionadas con los conjuntos definidos por los dígitos.

|  |  |  |  |
| --- | --- | --- | --- |
| p | q | r | (p ∧ q) → r |
| True | True | True | True |
| True | True | False | False |
| True | False | True | True |
| True | False | False | True |
| False | True | True | True |
| False | True | False | True |
| False | False | True | True |
| False | False | False | True |

## Parte 2 con Código

**Operaciones con DNIs**

* 1. **Ingreso de DNIs**

Código para ingresar y transformar los DNIs en conjuntos de dígitos únicos:

dnis = [35966752, 42469802, 42561321]

conjuntos = [set(str(dni)) for dni in dnis]

* 1. **Operaciones entre conjuntos**

Código para realizar operaciones entre conjuntos:

A, B, C = conjuntos

union = A | B | C

interseccion = A & B & C

diferencia\_AB = A - B

diferencia\_sim\_AB = A ^ B

* 1. **Visualización de resultados**
  2. **Conteo de frecuencia de dígitos**

Código para contar la frecuencia de cada dígito en los DNIs:

from collections import Counter

for dni in dnis:

contador = Counter(str(dni))

print(f"Frecuencia en {dni}: {dict(contador)}")

* 1. **Suma total de los dígitos**

Código para calcular la suma total de los dígitos:

for dni in dnis:

suma = sum(int(d) for d in str(dni))

print(f"Suma de dígitos en {dni}: {suma}")

* 1. **Evaluación de condiciones lógicas**

Código para evaluar condiciones lógicas según las expresiones definidas:

if len(A & B & C) >= 1:

print("Dígito compartido")

if any(len(c) > 6 for c in conjuntos):

print("Diversidad numérica alta")

* 1. **Operaciones con Años de Nacimiento**

**Ingreso de años**

anios = [2003, 2001, 1999]

* 1. **Contar pares e impares**

pares = sum(1 for anio in anios if anio % 2 == 0)

impares = len(anios) - pares

print(f"Años pares: {pares}, impares: {impares}")

* 1. **Verificar si todos nacieron después del 2000**

if all(anio > 2000 for anio in anios):

print("Grupo Z")

* 1. **Verificar si alguno nació en un año bisiesto**

Función para detectar años bisiestos:

def es\_bisiesto(anio):

return (anio % 4 == 0 and anio % 100 != 0) or (anio % 400 == 0)

Uso de la función:

if any(es\_bisiesto(anio) for anio in anios):

print("Tenemos un año especial")

**Producto cartesiano: años × edades**

from itertools import product

from datetime import datetime

anios\_actuales = [datetime.now().year - anio for anio in anios]

cartesiano = list(product(anios, anios\_actuales))

print("Producto cartesiano (año, edad):", cartesiano)

## Descripción Individual de Tareas

## Iván

• Realizó el análisis y desarrollo teórico de la Parte 1, incorporando los conceptos de conjuntos, lógica, relaciones y operaciones matemáticas.

• Redactó expresiones lógicas y propuso condiciones computables basadas en los conjuntos formados por los dígitos de los DNIs.

• Implementó el código en Python para las operaciones de conjuntos, cálculos de frecuencias y suma de dígitos.

• Participó en el diseño general del enfoque metodológico que conecta los contenidos de Matemática con Programación.

## Nikolas

• Colaboró en la implementación de las operaciones con años de nacimiento, identificación de años pares/impares y lógica condicional.

• Desarrolló la función para verificar si un año es bisiesto e integró el producto cartesiano entre años y edades.

• Contribuyó con la revisión del código Python y pruebas de ejecución.

• Redactó parte de la documentación relacionada con las funciones y estructuras utilizadas en el desarrollo del programa.

## Natasha

• Participó en la construcción y comparación de los conjuntos de dígitos únicos a partir de los DNIs.

• Realizó las operaciones de diferencia, intersección y diferencia simétrica y su interpretación en el contexto del problema.

• Creó los diagramas de Venn digitales basados en las operaciones realizadas.

• Apoyó la redacción final y el formato del informe para su presentación en PDF.

## Relación entre Expresiones Lógicas y Código

Las expresiones lógicas redactadas en la Parte 1 se tradujeron en condiciones evaluadas mediante estructuras condicionales en Python. Por ejemplo:

• "Si algún dígito aparece en todos los conjuntos" se implementó usando intersección entre conjuntos.

• "Si algún conjunto tiene más de 6 elementos" se evaluó utilizando la función `len()` dentro de un bucle.

• "Si todos nacieron después del 2000" se resolvió con `all()` sobre la lista de años.

Estas condiciones permitieron generar respuestas automáticas basadas en la información cargada, integrando razonamiento lógico con programación estructurada.