Trabajo Práctico Integrador II
Matemática

Alumnos:

German Dagatti - DNI: 31230293

Nicolás Gabriel Demiryi - DNI: 39717139

Leonardo Cortes - DNI: 34711991

Comisión: N°12

Profesor/a: Vanina Durruty

Tutor: Maria Teresa Brizzi

Fecha de Entrega: 13/06/2025

Consigna

Objetivo

Profundizar la integración entre los contenidos de Matemática (conjuntos y lógica) y Programación (estructuras condicionales, repetitivas y funciones), fortaleciendo también el trabajo en equipo, la comunicación clara y la responsabilidad individual en proyectos colaborativos.

Trabajo en grupo

El trabajo debe hacerse en grupo y todos los integrantes deben pertenecer a la misma comisión.

La conformación de grupos tiene como objetivo fomentar la colaboración entre pares, una habilidad fundamental que todo programador debe desarrollar para integrarse eficazmente en proyectos de gran envergadura.

Cada integrante debe asumir responsabilidades específicas dentro del proyecto, explicar su parte en el video y entregar por escrito una descripción de las tareas que realizó.

Parte 1 – Desarrollo Matemático (Conjuntos y Lógica)

1. Cada integrante debe anotar su número de DNI.
2. A partir de los DNIs, se deben formar tantos conjuntos de dígitos únicos como integrantes tenga el grupo.
3. Realizar entre esos conjuntos las siguientes operaciones: unión, intersección, diferencia (entre pares) y diferencia simétrica.
4. Para cada una de estas operaciones, se debe realizar un diagrama de Venn (a mano o digital), que debe incluirse en la entrega.
5. Redactar al menos dos expresiones lógicas en lenguaje natural, que puedan luego implementarse en Python y escribir en la documentación que van a presentar cual seria el resultado con los conjuntos que tienen.

Ejemplos de expresiones lógicas:

- Si todos los conjuntos tienen al menos 5 elementos, entonces se

considera que hay una alta diversidad numérica.·

- Si el conjunto A tiene más elementos que el conjunto B y el conjunto C contiene al menos un número impar, entonces se cumple la condición de combinación amplia.·
- Si ningún conjunto tiene el número 0, entonces se considera un grupo sin ceros.·
- Si algún dígito aparece en todos los conjuntos, se marca como dígito común.·
- Si hay más conjuntos con cantidad par de elementos que con cantidad impar, entonces se etiqueta como "grupo par".·
- Si la intersección entre todos los conjuntos tiene exactamente un elemento, se considera un dígito representativo del grupo.

Estas expresiones deben incluirse en el archivo PDF de la parte teórica y se espera que al menos una de ellas se implemente directamente como lógica en el programa Python.

Parte 2 – Desarrollo del Programa en Python

El programa debe implementar varias de las ideas trabajadas en papel. Debe incluir:

A. Operaciones con DNIs

· Ingreso de los DNIs (reales o ficticios).

· Generación automática de los conjuntos de dígitos únicos.

· Cálculo y visualización de: unión, intersección, diferencias y diferencia simétrica.

· Conteo de frecuencia de cada dígito en cada DNI utilizando estructuras repetitivas.

· Suma total de los dígitos de cada DNI.

· Evaluación de condiciones lógicas (condicionales), vinculadas con las expresiones escritas.

Ejemplos:

· Si un dígito aparece en todos los conjuntos, mostrar "Dígito compartido".

· Si algún conjunto tiene más de 6 elementos, mostrar "Diversidad numérica alta".

B. Operaciones con años de nacimiento

· Ingreso de los años de nacimiento (Si dos o mas integrantes del grupo tienen el mismo año, ingresar algún dato ficticio, según el caso).

· Contar cuántos nacieron en años pares e impares utilizando estructuras repetitivas.

· Si todos nacieron después del 2000, mostrar "Grupo Z".

· Si alguno nació en año bisiesto, mostrar "Tenemos un año especial".

· Implementar una función para determinar si un año es bisiesto.

· Calcular el producto cartesiano entre el conjunto de años y el conjunto de edades actuales.

Parte 3 – Video de Presentación

Duración estimada entre 5 y 10 minutos. Todos los integrantes deben presentarse en cámara, mostrar el programa funcionando y explicar la parte que realizaron. También deben comentar brevemente qué aprendieron al combinar matemática y programación.

Entrega final

1. Archivo PDF con: desarrollo de conjuntos y operaciones, todos los diagramas de Venn, expresiones lógicas redactadas, y tareas de cada integrante explicadas por escrito.

2. Archivo con extensión .py que contenga el programa en Python.

3. Video grupal subido en lo posible a YouTube.

4. Documento adicional con los nombres de los integrantes, descripción de lo que hizo cada uno y la relación entre las expresiones lógicas escritas y el código implementado.

Calificación

La calificación será numérica y para aprobar deberá ser mayor o igual a 6.

E J E M P L O

Supongamos que en el grupo hay 3 integrantes, con los siguientes DNIs:

- Persona A: 23456789
- Persona B: 22334455
- Persona C: 98765432

De cada DNI se toman los dígitos únicos y se arma un conjunto (sin repetir elementos).

- Conjunto A = {2, 3, 4, 5, 6, 7, 8, 9}
- Conjunto B = {2, 3, 4, 5}
- Conjunto C = {2, 3, 4, 5, 6, 7, 8, 9}

Ejemplo A y B:

- Unión (A ∪ B): todos los elementos de A y B sin repetir → {2, 3, 4, 5, 6, 7, 8, 9}

- Intersección (A ∩ B): elementos que están en ambos → {2, 3, 4, 5} ● Diferencia (A - B): elementos de A que no están en B → {6, 7, 8, 9}

- Diferencia simétrica (A Δ B): elementos que están en A o en B pero no en ambos → {6, 7, 8, 9} (En este caso, B no tiene ningún dígito que A no tenga, así que la diferencia simétrica es igual a la diferencia A - B.)

Diagrama de Venn: Hay herramientas online como https://venngage.com o pueden hacerlo en PowerPoint, Paint o a mano en papel.

Expresiones lógicas en lenguaje natural Estas son frases que expresan condiciones que se pueden traducir a código Python.

Ejemplo 1:Lenguaje natural: “Los dígitos que están en A y no están en B”

Resultado con los conjuntos del ejemplo: {6, 7, 8, 9}

Ejemplo 2: Lenguaje natural: “Los dígitos que están en A o en B, pero no en ambos”

Resultado: {6, 7, 8, 9}

Ejemplo 3: “Los dígitos que aparecen en los tres DNIs a la vez”

Resultado: {2, 3, 4, 5}

Parte 1 – Desarrollo Matemático (Conjuntos y Lógica)

1. Cada integrante debe anotar su número de DNI.

Alumno 1: 34711991
Alumno 2: 31230293
Alumno 3: 39717139

2. A partir de los DNIs, se deben formar tantos conjuntos de dígitos únicos como integrantes tenga el grupo.

Conjuntos de dígitos únicos por alumno

Alumno 1 (DNI: 34711991) → Conjunto A = {1, 3, 4, 7, 9}

Alumno 2 (DNI: 31230293) → Conjunto B = {0, 1, 2, 3, 9}

Alumno 3 (DNI: 39717139) → Conjunto C = {1, 3, 7, 9}

3. Realizar entre esos conjuntos las siguientes operaciones: unión, intersección, diferencia (entre pares) y diferencia simétrica.

4. Para cada una de estas operaciones, se debe realizar un diagrama de Venn (a mano o digital), que debe incluirse en la entrega.

Operaciones entre conjuntos

- A ∪ B ∪ C (Unión) = {1,3,4,7,9,2}

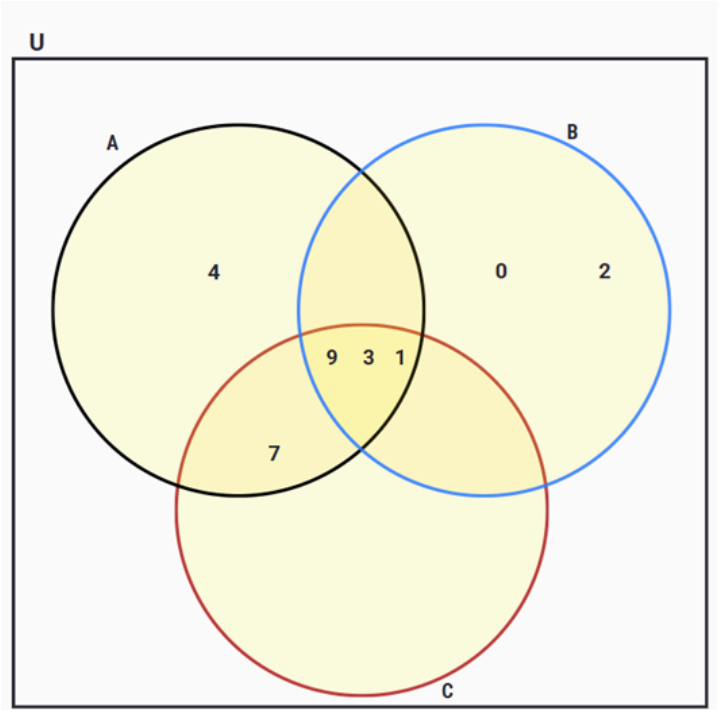


Gráfico 1: Elaboración propia con la herramienta venngage y Paint

- A ∩ B ∩ C (Intersección) → {1, 3, 9}

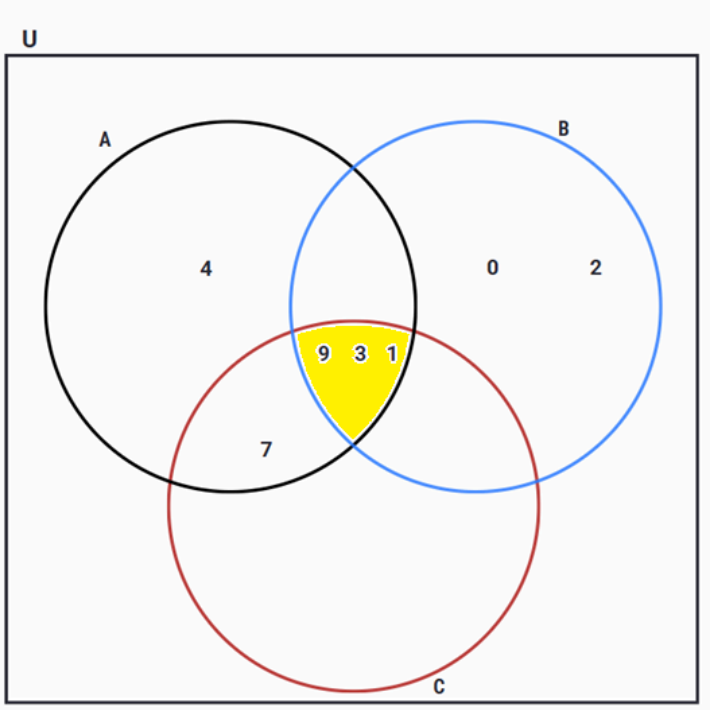


Gráfico 2: Elaboración propia con la herramienta venngage y Paint

- A - B (Diferencia) → {'4', '7'}

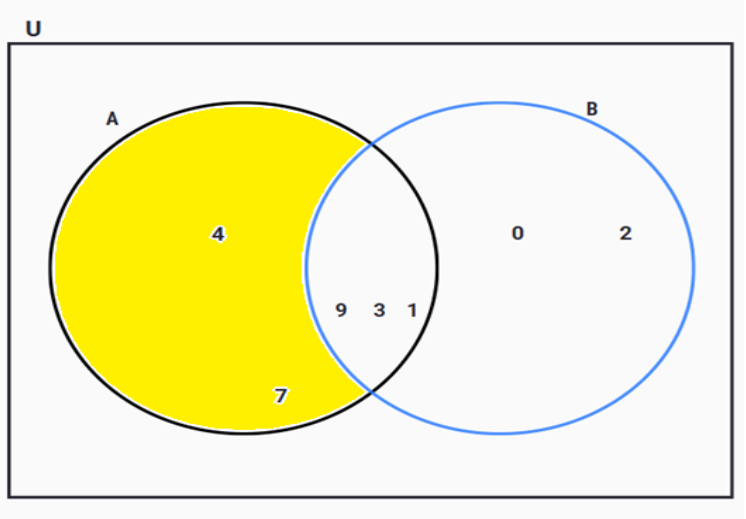


Gráfico 3: Elaboración propia con la herramienta venngage y Paint

- A Δ B (Diferencia simétrica) → {'0', '2', '4', '7'}

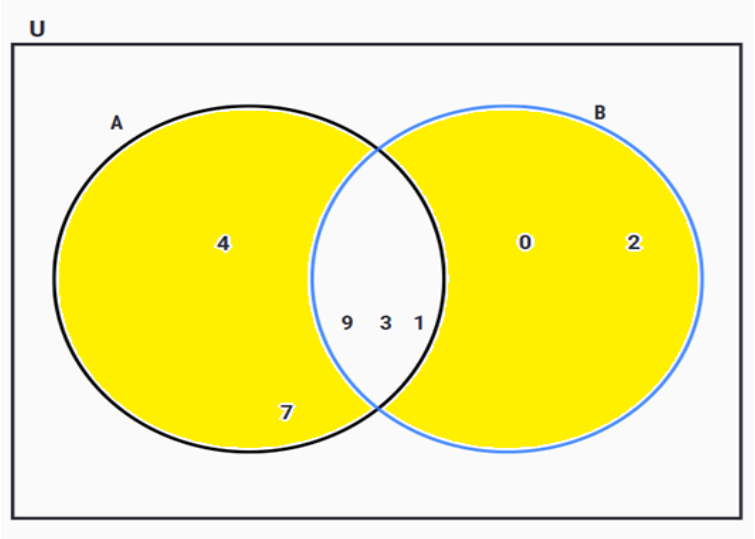


Gráfico 4: Elaboración propia con la herramienta venngage y Paint

- A - C → {'4'}

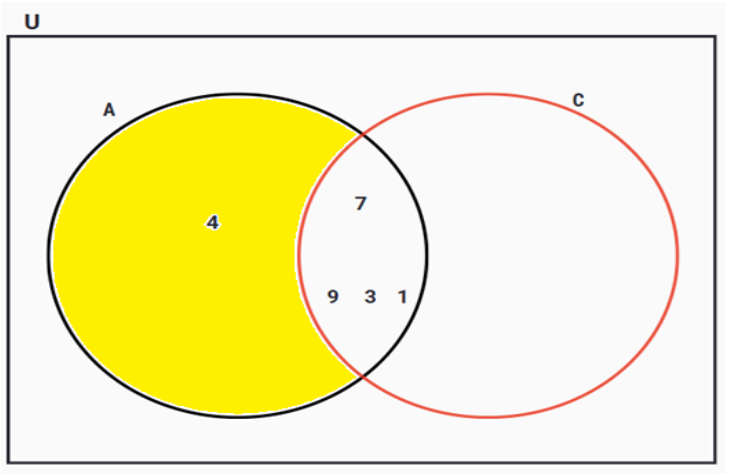


Gráfico 5: Elaboración propia con la herramienta venngage y Paint

- A Δ C → {'4'}

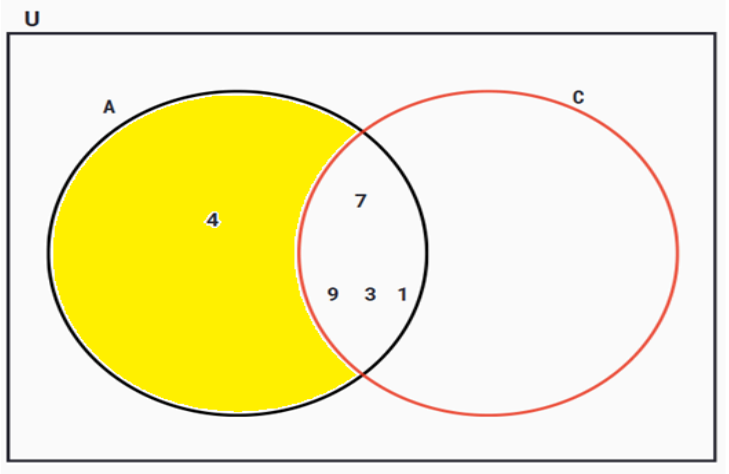


Gráfico 6: Elaboración propia con la herramienta venngage y Paint

- B - C → {'0', '2'}

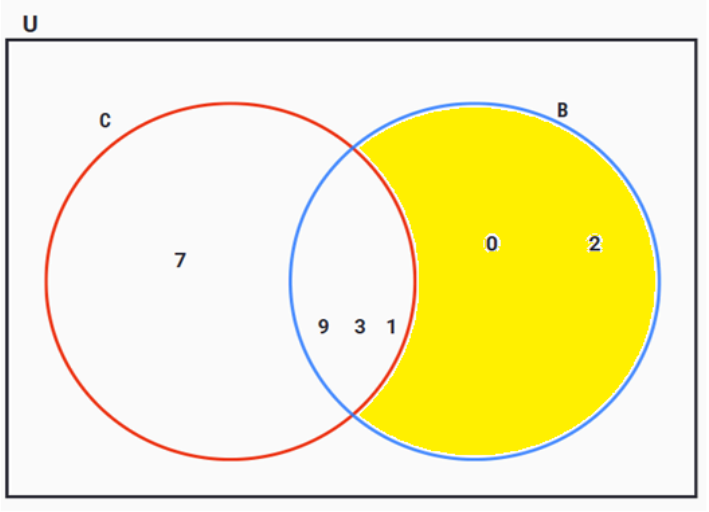


Gráfico 7: Elaboración propia con la herramienta venngage y Paint

- B Δ C → {'0', '2', '7'}

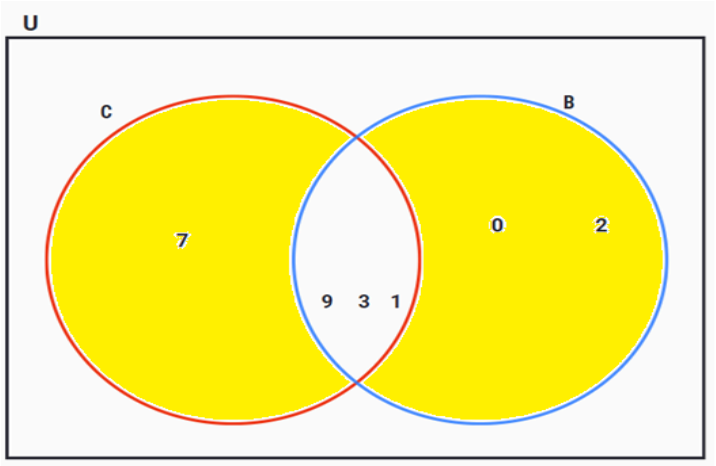


Gráfico 8: Elaboración propia con la herramienta venngage y Paint

- A Δ B Δ C (Diferencia Simétrica total) → {'0', '1', '2', '3', '4', '9'}

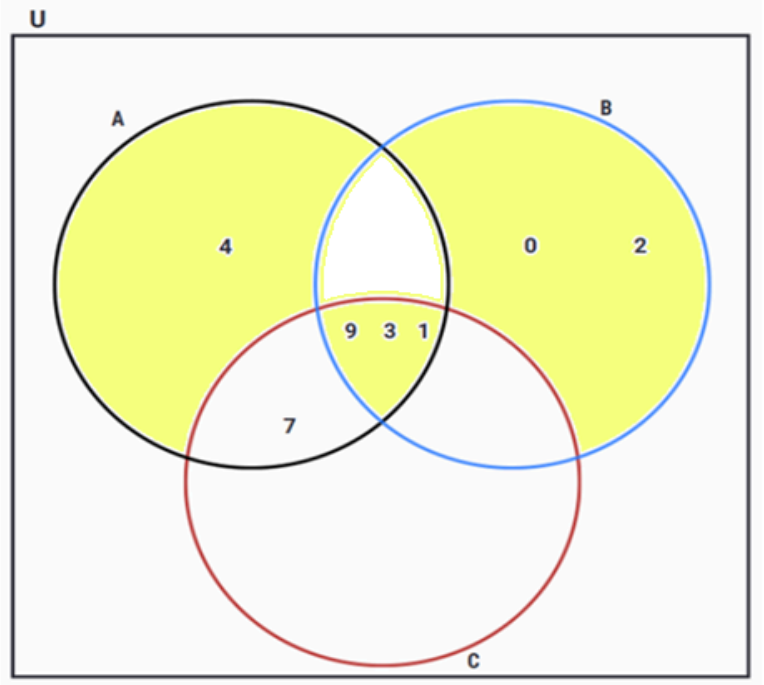


Gráfico 9: Elaboración propia con la herramienta venngage y Paint

5. Redactar al menos dos expresiones lógicas en lenguaje natural, que puedan luego implementarse en Python y escribir en la documentación que van a presentar cuál sería el resultado con los conjuntos que tienen.

Expresión lógica 1:
“Un número natural pertenece al conjunto A si es mayor que 10 y es par”

Implementación en Python (lógica):
A = [x for x in conjunto if x > 10 and x % 2 == 0]

Ejemplo de conjunto:

conjunto = [4, 11, 12, 15, 18, 21, 8

Resultado esperado:

A = [12, 18]

Porque son los únicos números que cumplen ambas condiciones: mayores que 10 y pares.

Expresión lógica 2:

Lenguaje natural:

"Un número pertenece al conjunto B si es menor que 5 o es múltiplo de 3."

Implementación en Python (lógica):

B = [x for x in conjunto if x < 5 or x % 3 == 0]

Ejemplo de conjunto:

conjunto = [2, 3, 5, 6, 7, 9, 12, 14]

Resultado esperado:

B = [2, 3, 6, 9, 12]

Porque cumplen al menos una de las condiciones: ser menores que 5 o múltiplos de 3.

Parte 2 – Desarrollo del Programa en Python

El programa debe implementar varias de las ideas trabajadas en papel. Debe incluir:

A. Operaciones con DNIs

- Ingreso de los DNIs (reales o ficticios).
- Generación automática de los conjuntos de dígitos únicos.
- Cálculo y visualización de: unión, intersección, diferencias y diferencia simétrica.
- Conteo de frecuencia de cada dígito en cada DNI utilizando estructuras repetitivas.
- Suma total de los dígitos de cada DNI.
- Evaluación de condiciones lógicas (condicionales), vinculadas con las expresiones escritas.

Ejemplos:

- Si un dígito aparece en todos los conjuntos, mostrar "Dígito compartido".
- Si algún conjunto tiene más de 6 elementos, mostrar "Diversidad numérica alta".

Programa utilizado

```
def Genera_Conjuntos(Aux):
    str_DNI = str(Aux)
    Conjunto_DNI = {int(digito) for digito in str_DNI}
    return Conjunto_DNI

def Union_Conjuntos(A,B):
    C = A.union(B)
```

```
    return C

def Union_total(A, B, C):
    U_total = A | B | C
    return U_total


def Inter_Conjuntos(A,B):
    C = A.intersection(B)
    return C

def Inter_Total(A,B,C):
    I_total = A & B & C
    return I_total

def Diferencia_Conjuntos(A,B):
    C = A - B
    return C

def Diferencia_simetrica(A,B):
    C = A.symmetric_difference(B)
    return C

def DSimetrica_total(A, B, C):
    Ds_total= A ^ B ^ C
    return Ds_total

def contar_frecuencia_digitos(dni):
    frecuencia = {str(i): 0 for i in range(10)}
    suma_total = 0
    for digito in dni:
        if digito in frecuencia:
            frecuencia[digito] += 1
            suma_total += int(digito)
    return frecuencia, suma_total

def mostrar_elementos(conjunto):
    elementos = ""
    for i, digito in enumerate(conjunto):
        if i > 0:
            elementos += ", "
        elementos += str(digito)
    return elementos


# DNI de los Integrantes de Grupo
# Generación de Conjuntos a partir de los DNI con los que se va a operar

DNI_1 = 34711991
conjunto_DNI1 = Genera_Conjuntos(DNI_1)
```


```
DNI_2 = 31230293
conjunto_DNI2 = Genera_Conjuntos(DNI_2)

DNI_3 = 39717139
conjunto_DNI3 = Genera_Conjuntos(DNI_3)

# Aquí mostramos los conjunto con los que se va a operar
print(f"Conjunto 1: {conjunto_DNI1}")
print(f"Conjunto 2: {conjunto_DNI2}")
print(f"Conjunto 3: {conjunto_DNI3}")

# Visualización de Operaciones de conjuntos unión, intersección, diferencias y
diferencia simétrica
print("----------------------Union de Conjuntos--------------------")

print(f"{conjunto_DNI1} Union {conjunto_DNI2} =
{Union_Conjuntos(conjunto_DNI1,conjunto_DNI2)}")
print(f"{conjunto_DNI1} Union {conjunto_DNI3} =
{Union_Conjuntos(conjunto_DNI1,conjunto_DNI3)}")
print(f"{conjunto_DNI2} Union {conjunto_DNI3} =
{Union_Conjuntos(conjunto_DNI2,conjunto_DNI3)}")

print(f"Union Total de los Conjuntos =
{Union_total(conjunto_DNI1,conjunto_DNI2,conjunto_DNI3)}")

print("--------------------Interseccion de Conjuntos------------------")

print(f"{conjunto_DNI1} Interseccion {conjunto_DNI2} =
{Inter_Conjuntos(conjunto_DNI1,conjunto_DNI2)}")
print(f"{conjunto_DNI1} Interseccion {conjunto_DNI3} =
{Inter_Conjuntos(conjunto_DNI1,conjunto_DNI3)}")
print(f"{conjunto_DNI2} Interseccion {conjunto_DNI3} =
{Inter_Conjuntos(conjunto_DNI2,conjunto_DNI3)}")


print(f"Interseccion Total de los Conjuntos =
{Inter_Total(conjunto_DNI1,conjunto_DNI2,conjunto_DNI3)}")

print("-------------------Diferencia de Conjuntos por Pares-----------------")
print(f"{conjunto_DNI1} Diferencia {conjunto_DNI2} =
{Diferencia_Conjuntos(conjunto_DNI1,conjunto_DNI2)}")
print(f"{conjunto_DNI2} Diferencia {conjunto_DNI1} =
{Diferencia_Conjuntos(conjunto_DNI2,conjunto_DNI1)}")

print(f"{conjunto_DNI1} Diferencia {conjunto_DNI3} =
{Diferencia_Conjuntos(conjunto_DNI1,conjunto_DNI3)}")
print(f"{conjunto_DNI3} Diferencia {conjunto_DNI1} =
{Diferencia_Conjuntos(conjunto_DNI3,conjunto_DNI1)}")

print(f"{conjunto_DNI2} Diferencia {conjunto_DNI3} =
{Diferencia_Conjuntos(conjunto_DNI2,conjunto_DNI3)}")
```

```
print(f"{conjunto_DNI3} Diferencia {conjunto_DNI2} =
{Diferencia_Conjuntos(conjunto_DNI3,conjunto_DNI2)}")


print("----------Diferencia Simetrica de Conjuntos----------")

print(f"{conjunto_DNI1} Diferencia Simetrica {conjunto_DNI2} =
{Diferencia_simetrica(conjunto_DNI1,conjunto_DNI2)}")
print(f"{conjunto_DNI1} Diferencia Simetrica {conjunto_DNI3} =
{Diferencia_simetrica(conjunto_DNI1,conjunto_DNI3)}")
print(f"{conjunto_DNI2} Diferencia Simetrica {conjunto_DNI3} =
{Diferencia_simetrica(conjunto_DNI2,conjunto_DNI3)}")

print(f"Diferencia Simetrica Total =
{DSimetrica_total(conjunto_DNI1,conjunto_DNI2,conjunto_DNI3)}")


print("----------Conteo de frecuencia de cada dígito en cada
DNI----------")
print("----------Suma total de los dígitos de cada DNI----------")
dnis = [str(DNI_1), str(DNI_2), str(DNI_3)]
for dni in dnis:
    frecuencia, suma_total = contar_frecuencia_digitos(dni)
    print(f"Frecuencia de dígitos en el DNI {dni}:")
    for digito, count in frecuencia.items():
        if count > 0:
            print(f"Dígito {digito}: {count} veces")
    print(f"Suma total de los dígitos en el DNI {dni}: {suma_total}")
    print("")


print("---------- Evaluación de condiciones lógicas (condicionales), vinculadas
con las expresiones escritas ----------")

A = Inter_Total(conjunto_DNI1,conjunto_DNI2,conjunto_DNI3)
B = Union_total(conjunto_DNI1,conjunto_DNI2,conjunto_DNI3)
if len(A) != 0:
    elementos = mostrar_elementos(A)
    if len(A) > 1:
        print(f"Los digitos compartidos entre los tres DNIs son {elementos}.")
    else:
        print(f"El digito compartido entre los tres DNIs es {elementos}.")
else:
    print("No existe dígito común a todos los conjuntos")

if len(B)>0:
    elementos_union = mostrar_elementos(B)
    print(f"Los dígitos presente en los tres DNIs son {elementos_union}")

Conjunto_DNIs=[conjunto_DNI1,conjunto_DNI2, conjunto_DNI3]
digitos = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
for i in range(0, len(Conjunto_DNIs)):
    elementos_dni = Inter_Conjuntos(Conjunto_DNIs[i],{0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
    if len(elementos_dni) >=5:
        print(f"El DNI {dnis[i]} tiene 5 dígitos. Podemos decir que tiene 'Diversidad numérica alta'")
```

Salida del programa

```
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Matematica I\TPMatematica> pythc
mpCodeRunnerFile.py"
Conjunto 1: {1, 3, 4, 7, 9}
Conjunto 2: {0, 1, 2, 3, 9}
Conjunto 3: {9, 3, 1, 7}
----------Union de Conjuntos----------
{1, 3, 4, 7, 9} Union {0, 1, 2, 3, 9} = {0, 1, 2, 3, 4, 7, 9}
{1, 3, 4, 7, 9} Union {9, 3, 1, 7} = {1, 3, 4, 7, 9}
{0, 1, 2, 3, 9} Union {9, 3, 1, 7} = {0, 1, 2, 3, 7, 9}
Union Total de los Conjuntos = {0, 1, 2, 3, 4, 7, 9}
----------Interseccion de Conjuntos----------
{1, 3, 4, 7, 9} Interseccion {0, 1, 2, 3, 9} = {1, 3, 9}
{1, 3, 4, 7, 9} Interseccion {9, 3, 1, 7} = {9, 3, 1, 7}
{0, 1, 2, 3, 9} Interseccion {9, 3, 1, 7} = {9, 3, 1}
Interseccion Total de los Conjuntos = {1, 3, 9}
----------Diferencia de Conjuntos por Pares----------
{1, 3, 4, 7, 9} Diferencia {0, 1, 2, 3, 9} = {4, 7}
{0, 1, 2, 3, 9} Diferencia {1, 3, 4, 7, 9} = {0, 2}
{1, 3, 4, 7, 9} Diferencia {9, 3, 1, 7} = {4}
{9, 3, 1, 7} Diferencia {1, 3, 4, 7, 9} = set()
{0, 1, 2, 3, 9} Diferencia {9, 3, 1, 7} = {0, 2}
{9, 3, 1, 7} Diferencia {0, 1, 2, 3, 9} = {7}
----------Diferencia Simetrica de Conjuntos----------
{1, 3, 4, 7, 9} Diferencia Simetrica {0, 1, 2, 3, 9} = {0, 2, 4, 7}
{1, 3, 4, 7, 9} Diferencia Simetrica {9, 3, 1, 7} = {4}
{0, 1, 2, 3, 9} Diferencia Simetrica {9, 3, 1, 7} = {0, 2, 7}
Diferencia Simetrica Total = {0, 1, 2, 3, 4, 9}
----------Conteo de frecuencia de cada dígito en cada DNI----------
```

```
----------Conteo de frecuencia de cada dígito en cada DNI----------
----------Suma total de los dígitos de cada DNI----------
Frecuencia de dígitos en el DNI 34711991:
Dígito 1: 3 veces
Dígito 3: 1 veces
Dígito 4: 1 veces
Dígito 7: 1 veces
Dígito 9: 2 veces
Suma total de los dígitos en el DNI 34711991: 35

Frecuencia de dígitos en el DNI 31230293:
Dígito 0: 1 veces
Dígito 1: 1 veces
Dígito 2: 2 veces
Dígito 3: 3 veces
Dígito 9: 1 veces
Suma total de los dígitos en el DNI 31230293: 23

Frecuencia de dígitos en el DNI 39717139:
Dígito 1: 2 veces
Dígito 3: 2 veces
Dígito 7: 2 veces
Dígito 9: 2 veces
Suma total de los dígitos en el DNI 39717139: 40

---------- Evaluación de condiciones lógicas (condicionales), vinculadas con las expresiones escritas --
Los digitos compartidos entre los tres DNIs son 1, 3, 9.
Los dígitos presente en los tres DNIs son 0, 1, 2, 3, 4, 7, 9
El DNI 34711991 tiene 5 dígitos. Podemos decir que tiene 'Diversidad numérica alta'
El DNI 31230293 tiene 5 dígitos. Podemos decir que tiene 'Diversidad numérica alta'
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Matematica I\TPMatematica>
```

B. Operaciones con años de nacimiento

- Ingreso de los años de nacimiento (Si dos o más integrantes del grupo tienen el mismo año, ingresar algún dato ficticio, según el caso).
- Contar cuántos nacieron en años pares e impares utilizando estructuras repetitivas.
- Si todos nacieron después del 2000, mostrar "Grupo Z".
- Si alguno nació en año bisiesto, mostrar "Tenemos un año especial".
- Implementar una función para determinar si un año es bisiesto.
- Calcular el producto cartesiano entre el conjunto de años y el conjunto de edades actuales.

Programa Utilizado

Definición de función bisiesto

```
def es_bisiesto(año):
    """
    Función para determinar si un año es bisiesto.
    :param año: Año a verificar.
    :return: True si el año es bisiesto, False de lo contrario.
    """
    if año % 400 == 0:
        return True
    if año % 100 == 0:
        return False
    if año % 4 == 0:
        return True
    return False

# Ejemplo de uso
if __name__ == "__main__":
    año = int(input("Ingrese un año: "))
    if es_bisiesto(año):
        print(f"{año} es un año bisiesto.")
    else:
        print(f"{año} no es un año bisiesto.")
```

Programa inciso B

```
"""Operaciones con años de nacimiento
```

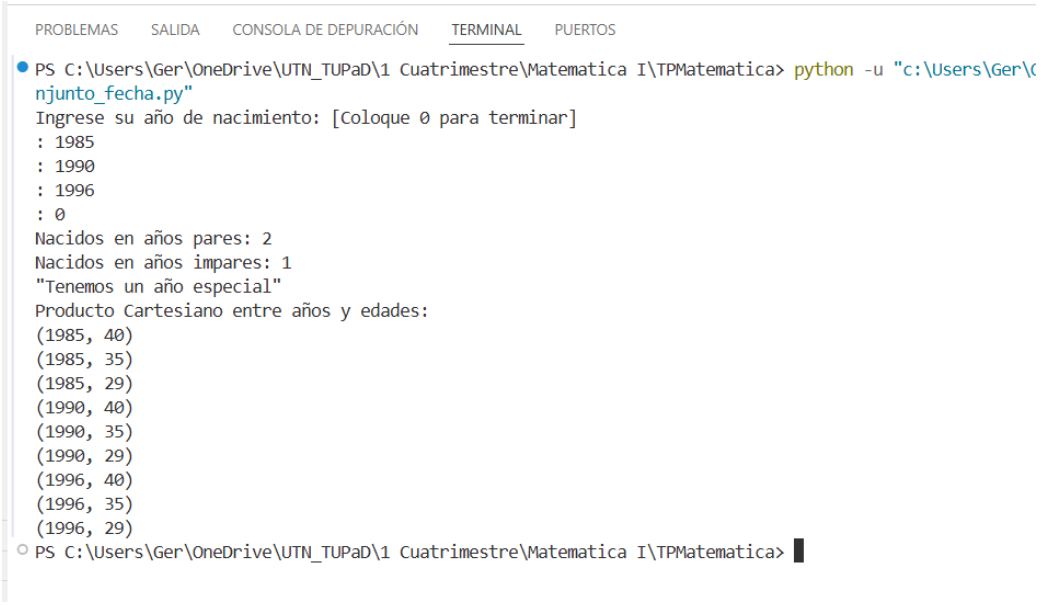
UTN
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

```
· Ingreso de los años de nacimiento (Si dos o mas integrantes del grupo tienen el mismo año,

  ingresar algún dato ficticio, según el caso).

· Contar cuántos nacieron en años pares e impares utilizando estructuras repetitivas.

· Si todos nacieron después del 2000, mostrar "Grupo Z".

· Si alguno nació en año bisiesto, mostrar "Tenemos un año especial".

· Implementar una función para determinar si un año es bisiesto.

· Calcular el producto cartesiano entre el conjunto de años y el conjunto de edades actuales.

"""

# Importamos la paquete bisiesto

import bisiesto as bi

import itertools



#Ingreso de los años de nacimiento

print("Ingrese su año de nacimiento: [Coloque 0 para terminar]")

a = 1 # generamos un contador

anios = []

edades = []

while a != 0:

    a = int(input(": "))

    if a != 0:

        anios.append(a)

        edades.append(2025-a)



#Contar cuántos nacieron en años pares e impares utilizando estructuras repetitivas.

#Si todos nacieron después del 2000, mostrar "Grupo Z".



contador_pares = 0   # Inicializar contadores
```

```
contador_impares = 0
contador_2000= 0
contador_bisiestro = 0
i=0
while i < len(anios):
    anio=anios[i]
    # Contar cuantos años son pares y cuantos impares
    if anio % 2 == 0: # Verifica si el resto de la división por 2 es igual a 0
        contador_pares += 1
    else:
        contador_impares += 1

    # Cuenta cuantos son mayores que 2000
    if anio > 2000:
        contador_2000 +=1

    # Verificar si hay años bisiestos
    if bi.es_bisiesto(anio):
        contador_bisiestro +=1

    i+=1

# Calcular el producto cartesiano entre el conjunto de años y el conjunto de edades actuales
producto_cartesiano = []
for año in anios:
    for edad in edades:
        producto_cartesiano.append((año, edad))

#Muestra resultados
```

```
print(f"Nacidos en años pares: {contador_pares}")
print(f"Nacidos en años impares: {contador_impares}")
if contador_2000 == len(anios):
    print('"Grupo Z"')
if contador_bisiestro > 0:
    print('"Tenemos un año especial"')
print("Producto Cartesiano entre años y edades:")
for par in producto_cartesiano:
    print(par)
```

Salida del programa



División de Tareas del Grupo

En este proyecto, cada integrante del grupo asumió responsabilidades específicas, aunque todos colaboramos activamente en las distintas etapas del trabajo. La distribución principal de tareas fue la siguiente:

Germán Dagatti: Se encargó principalmente del desarrollo de las operaciones relacionadas con los años de nacimiento en el programa Python. Esto incluyó la implementación de la función para detectar años bisiestos, el conteo de años pares e impares, y el cálculo del producto cartesiano entre años y edades.

Leonardo: Fue responsable de las operaciones con los DNI. Desarrolló la lógica

para generar los conjuntos de dígitos únicos, realizar las operaciones de conjuntos (unión, intersección, diferencia y diferencia simétrica), y aplicar condiciones lógicas basadas en los dígitos.

Nicolás: Se ocupó de los aspectos formales y de redacción de la Parte 1 del trabajo, incluyendo la elaboración de las expresiones lógicas en lenguaje natural, la documentación teórica y la edición del video grupal.

A pesar de esta división, todos los integrantes participaron activamente en la revisión del código, la elaboración de los diagramas de Venn, la grabación del video y la integración de las distintas partes del proyecto. El trabajo fue verdaderamente colaborativo y nos permitió combinar conocimientos de matemática y programación de manera efectiva.


UTN
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA


UTN
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA


UTN
TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA