Trabajo Integrador 2: Matemática y Programación

Integrantes del grupo:

Bruno Pighin- DNI: 35940327;

brunopighin@hotmail.com

Aldo Manfredi - DNI: 27028093;
manfredialdo.1979@gmail.com

Profesor: Ernesto Klimovsky

Tutor: Andrea Comerci

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 cuál sería el resultado con los conjuntos que tienen.
==========
==========

1. DNIs:

Bruno Pighin: 35940327

Aldo Manfredi: 27028093

2. Conjuntos de dígitos únicos:

Conjunto A = {0, 2, 3, 4, 5, 7, 9}

Conjunto B = {0, 2, 3, 7, 8, 9}

3. Operaciones entre los conjuntos:

Siendo A = {0, 2, 3, 4, 5, 7, 9} y B = {0, 2, 3, 7, 8, 9}

Unión (A ∪ B) = {0, 2, 3, 4, 5, 7, 8, 9}

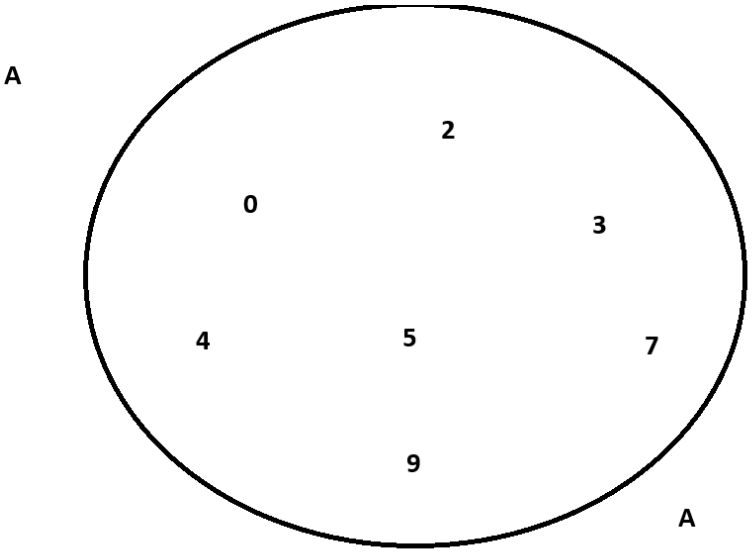
Intersección (A ∩ B) = {0, 2, 3, 7, 9}

Diferencia (A - B) = {4, 5}

Diferencia simétrica (A Δ B) = {4, 5, 8}

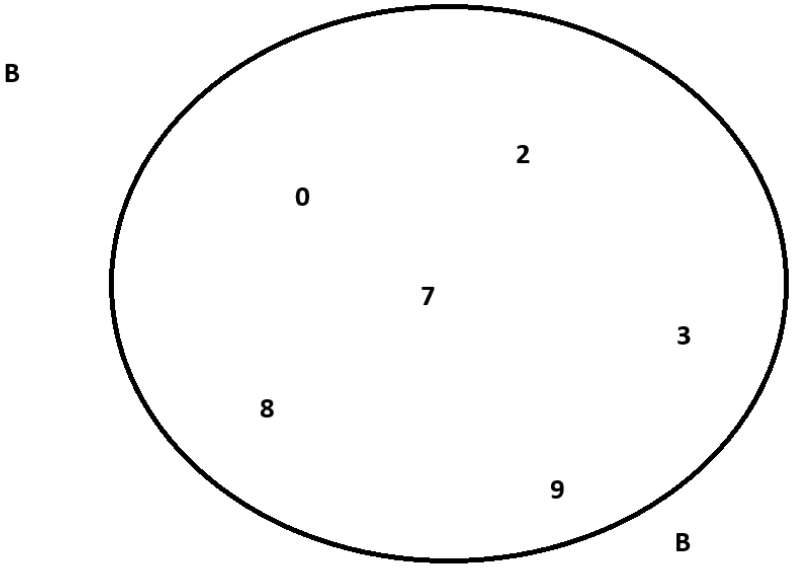
4. Diagramas de Venn:

Conjunto A = {0, 2, 3, 4, 5, 7, 9}



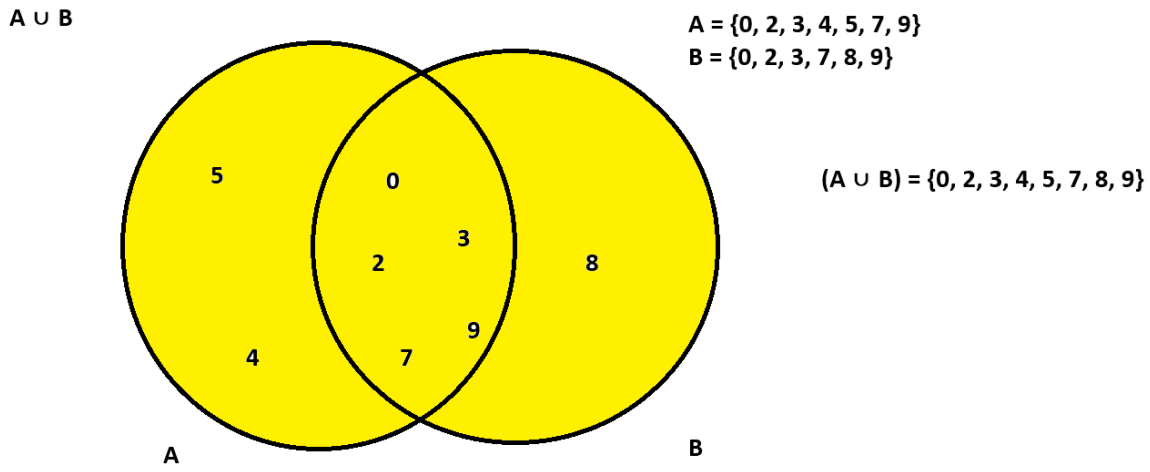
A = {0, 2, 3, 4, 5, 7, 9}

Conjunto B = {0, 2, 3, 7, 8, 9}

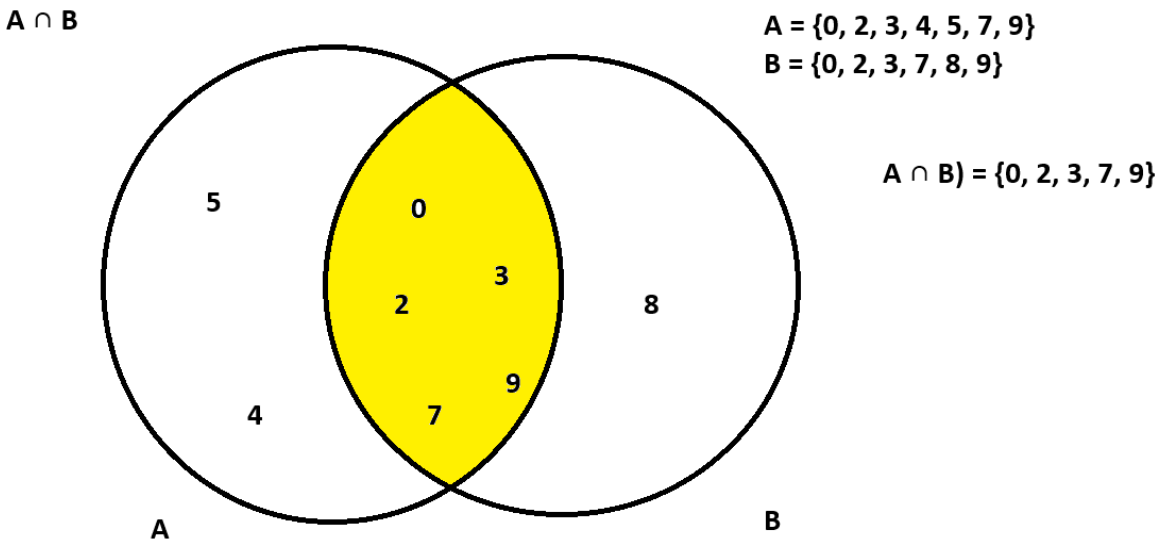


B = {0, 2, 3, 7, 8, 9}

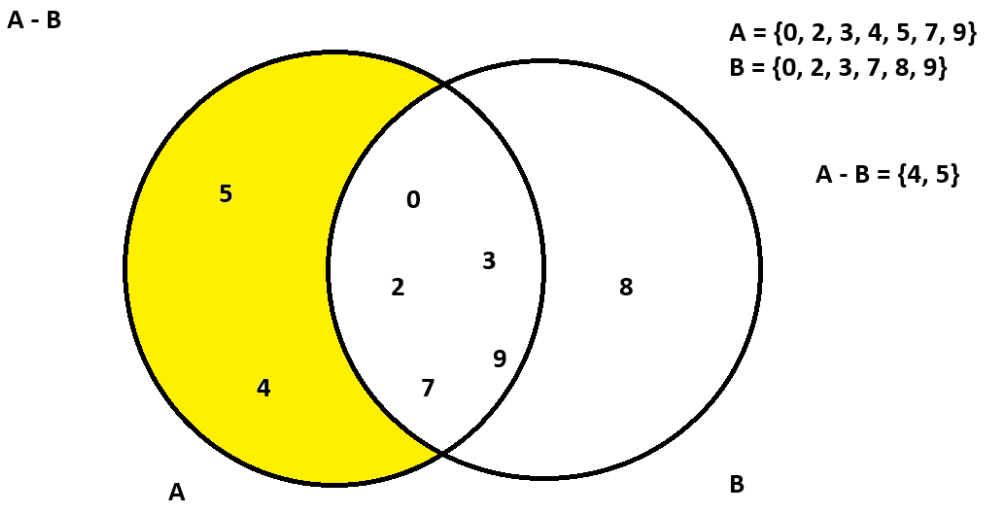
Unión (A ∪ B) = {0, 2, 3, 4, 5, 7, 8, 9}



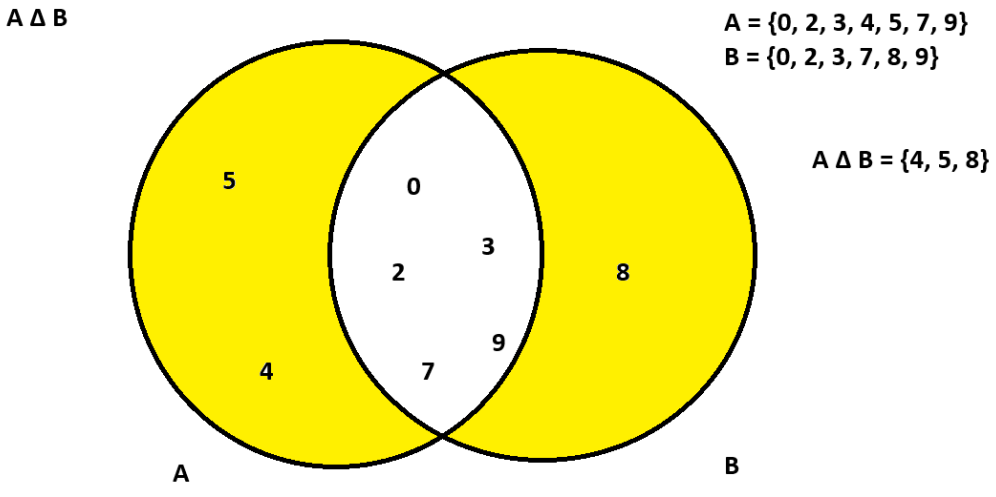
Intersección (A ∩ B) = {0, 2, 3, 7, 9}



Diferencia (A - B) = {4, 5}



Diferencia simétrica (A Δ B) = {4, 5, 8}



5. Expresiones lógicas en lenguaje natural:

- Si la suma de los elementos obtenidos de la diferencia A - B es impar y ambos conjuntos comparten al menos 5 elementos.. imprimir en pantalla el destino del día “estudiar”, caso contrario el imprimir “ver películas”... El resultado sería en nuestro caso: “estudiar”

- Si el número de elementos en la unión de A y B es par, y el número de elementos en la intersección de A y B es mayor que 3, y el conjunto A contiene el número 0, entonces modificar la variable estado del sistema con el texto "Alerta, se detectaron algunos fallos", caso contrario modificar la misma variable con el texto "óptimo". El resultado en nuestro caso sería "óptimo"

SCRIPTS PARTE1

Para la parte1 realizamos dos scripts en python… Ambos scripts en el anexo..

scriptparte1a.py y scriptparte1b.py

IMPLEMENTACIÓN scriptparte1a.py

función dni_a_conjunto: toma los dnis, y nos devuelve el conjunto de los elementos de un dni;

función condicion1 va tomar los dos conjuntos y devuelve True si la suma de los elementos de la diferencia es impar

función condicion2 a partir de dos conjuntos y devuelve True si la insterseccion tiene al menos 5 elementos

función inicio que es la encargada de verificar la lógica e imprimir resultado

IDEA LÓGICA: si se cumplen ambas condiciones mostrar en pantalla “ a estudiar” sino “ver pelis”

IMPLEMENTACIÓN scriptparte1b.py (ver anexo)

Necesitamos que verifiquen el resultado de estas condiciones.. parte del ejemlo consigna…

condicion3: Si todos los conjuntos tienen al menos 5 elementos, entonces se # considera que hay una alta diversidad numérica.·

condicion4: Si algún dígito aparece en todos los conjuntos, se marca como # dígito común.·

condicion5: Si la intersección entre todos los conjuntos tiene exactamente un elemento, se considera un dígito representativo del grupo.

Funciones implementadas:

dni_a_conjunto: toma un dni y devuelve un conjunto

CÁLCULOS

condicion3(a, b): toma dos dni y nos devuelve si hay alta diversidad numérica

condicion4(a,b): toma los dos dni y devuelve True si la intersección tmas de un elemento

condicion5(a,b) toma dos dnis y devuelve True si la ila intersección tiene exactamente un elemento

LÓGICA

inicio: es la encargada de tomar los dnis e imprimir los resultados.

==========
==========

Parte 2A – Desarrollo del Programa en Python
El programa debe implementar varias de las ideas trabajadas en papel. Debe incluir:

A**. Operaciones con DNIs**
Parte 2Aa· Ingreso de los DNIs (reales o ficticios).
Parte 2Ab· Generación automática de los conjuntos de dígitos únicos.
Parte 2Ac· Cálculo y visualización de: unión, intersección, diferencias y diferencia simétrica.
Parte 2Ad· Conteo de frecuencia de cada dígito en cada DNI utilizando estructuras repetitivas.
Parte 2Ae· Suma total de los dígitos de cada DNI.
Parte 2Af· Evaluación de condiciones lógicas (condicionales), vinculadas con las expresiones escritas.
Ejemplo:

scriptparte2a.py

Documentacion scriptparte2a.py (para codigo ver anexo)

Operaciones con DNIs

- Ingreso de los DNIs (reales o ficticios)....

- opción de generación automática de los conjuntos de dígitos únicos. #

El script scriptparte2a.py da la opción del ingreso manual de dnis y la generación aleatoria de dnis… luego realiza todos los siguientes cálculos y su posterior visualización,, y además evalúa si se cumplen algunas condiciones y muestra el resultado,,

CÁLCULO Y VISUALIZACIÓN DE

- unión, intersección, diferencias y diferencia simétrica.

- la frecuencia de cada dígito en cada DNI utilizando estructuras repetitivas.

- suma total de los dígitos de cada DNI.

CONDICIONES Tomando los ejemplos de la consigna sobre evaluación de condiciones, vinculadas con las expresiones escritas…

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

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

Una función para cada tarea… Se realiza la operación y visualización de cada cálculo, y demás condiciones lógicas.,,,

==========
==========

Parte 2B – Desarrollo del Programa en Python El programa debe implementar varias de las ideas trabajadas en papel. Debe incluir:
B. Operaciones con años de nacimiento
Parte2Bb· Contar cuántos nacieron en años pares e impares utilizando estructuras repetitivas.
Parte2Bc· Si todos nacieron después del 2000, mostrar "Grupo Z".
Parte2Bd· Si alguno nació en año bisiesto, mostrar "Tenemos un año especial". · Implementar una función para determinar si un año es bisiesto.
Parte 2Be· Calcular el producto cartesiano entre el conjunto de años y el conjunto de edades actuales.

PARTE2B: scriptparte2b.py (para el codigo, ver anexo)

Operaciones con años de nacimiento ·

Ingreso de los años de nacimiento

Implementar una función para determinar si un año es bisiesto

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

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

Si todos nacieron después del 2000, mostrar en pantalla "Grupo Z"

Si dos o más integrantes del grupo tienen el mismo año,.. mostrar “los dos nacieron el mismo año”

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

Una función para cada tarea… Se realiza la operación y visualización de cada cálculo, y se evalúan y muestra resultado de las condiciones lógicas.,,,

ANEXO:

scripts en https://github.com/manfredialdo/TUPADtpi2mate.git

video explicativo: https://www.youtube.com/watch?v=Sxsgc6xGZ-Q

SCRIPTS

PARTE1:

scriptparte1a.py

```
def dni_a_conjunto():
    dni = input("Por favor, ingrese un DNI (8 dígitos numéricos): ")
    if not dni.isdigit() or len(dni) != 8:
            raise ValueError("DNI inválido. Debe contener exactamente 8 dígitos numéricos.")
    return set(map(int, dni))


def condicion1(a,b):
    if sum(a.difference(b)) % 2!=0:
        return True


def condicion2(a, b):
    if len(a.intersection(b))>=5:
```

```
        return True


def inicio():
    # pedir dos dnis
    dniA = dni_a_conjunto()
    dniB = dni_a_conjunto()
    # las condiciones de nuestro caso practico
    if condicion1(dniA, dniB) == True and condicion2(dniA, dniB)==True:
        return "a estudiar"
    else:
        return "ver pelis"
# TEST dni A = Bruno Pighin- DNI: 35940327; y dni Aldo Manfredi - DNI: 27028093;
inicio()
```

PARTE1

scriptparte1b.py

```
def dni_a_conjunto():
    dni = input("Por favor, ingrese un DNI (8 dígitos numéricos): ")
    if not dni.isdigit() or len(dni) != 8:
            raise ValueError("DNI inválido. Debe contener exactamente 8 dígitos numéricos.")
    return set(map(int, dni))


def condicion3(a, b):
    if len(a)>=5 and len(b)>=5:
        return True
```

```
def condicion4(a,b):
    if len(a.intersection(b))>1:
        return True


def condicion5(a,b):
    if len(a.intersection(b))==1:
        return True


def inicio():
    dniA = dni_a_conjunto()
    dniB = dni_a_conjunto()
    # se evaluan condiciones
    if condicion3(dniA,dniB)==True:
        print("hay alta diversidad numerica")
    if condicion4(dniA,dniB)==True:
        print(f"los digitos comunes de ambos conjuntos son{dniA.intersection(dniB)}")
    if condicion5(dniA,dniB)==True:
          print(f"{dniA.intersection(dniB)} es el digito representativo compartidos por de ambos dnis")


inicio()
# TEST dni A = Bruno Pighin- DNI: 35940327; y dni Aldo Manfredi - DNI: 27028093;
```

PARTE2

scriptparte2a.py

```
from random import randint
```

```
def dni_aleatorio():
    nrodni = randint(10000000, 65000000)
    return nrodni, set(map(int, str(nrodni)))

#CALCULOS
def calcular_union(conjunto1, conjunto2):
    return conjunto1.union(conjunto2)

def calcular_interseccion(conjunto1, conjunto2):
    return conjunto1.intersection(conjunto2)

def calcular_diferencia(conjunto1, conjunto2):
    return conjunto1.difference(conjunto2)

def calcular_diferencia_simetrica(conjunto1, conjunto2):
    return conjunto1.symmetric_difference(conjunto2)

def sumar_digitos_dni(numero_dni):
    return sum(map(int, str(numero_dni)))

def contar_frecuencia_digitos(numero_dni):
    frecuencias = {}
    for digito_c in str(numero_dni):
        digito = int(digito_c)
        if digito in frecuencias:
            frecuencias[digito] += 1
```

```
        else:
            frecuencias[digito] = 1
    return frecuencias


# VISUALIZACION
def mostrar_resultados_conjuntos(conjunto_A_digitos, conjunto_B_digitos, original_dniA, original_dniB):
    """
    Parte2Ac· Visualización de unión, intersección, diferencias y diferencia simétrica.
    """
     print(f"Conjunto A (Dígitos DNI 1): {conjunto_A_digitos} \nConjunto B (Dígitos DNI 2): {conjunto_B_digitos}")

    union_result = calcular_union(conjunto_A_digitos, conjunto_B_digitos)
    print(f"Union (A U B): {union_result}")

    interseccion_result = calcular_interseccion(conjunto_A_digitos, conjunto_B_digitos)
    print(f"Interseccion (A ∩ B): {interseccion_result}")

    diferencia_ab_result = calcular_diferencia(conjunto_A_digitos, conjunto_B_digitos)
    print(f"Diferencia (A - B): {diferencia_ab_result}")

    diferencia_ba_result = calcular_diferencia(conjunto_B_digitos, conjunto_A_digitos)
    print(f"Diferencia (B - A): {diferencia_ba_result}")
```

```
    diferencia_simetrica_result = calcular_diferencia_simetrica(conjunto_A_digitos, conjunto_B_digitos)
    print(f"Diferencia Simétrica (A Δ B): {diferencia_simetrica_result}")
    print("-" * 30)

    # usamos 'original_dniA' y 'original_dniB' para las sumas
                print(f"Suma de dígitos del DNI 1 ({original_dniA}): {sumar_digitos_dni(original_dniA)}")
                print(f"Suma de dígitos del DNI 2 ({original_dniB}): {sumar_digitos_dni(original_dniB)}")
    print("-" * 30)

    # usamos 'original_dniA' y 'original_dniB' para las frecuencias
    print(f"Frecuencia de dígitos para DNI 1 ({original_dniA}):")
    frecuencias_dniA = contar_frecuencia_digitos(original_dniA)
    for digito, cuenta in sorted(frecuencias_dniA.items()):
        print(f"   Dígito {digito}: {cuenta} vez/veces")

    print(f"\nFrecuencia de dígitos para DNI 2 ({original_dniB}):")
    frecuencias_dniB = contar_frecuencia_digitos(original_dniB)
    for digito, cuenta in sorted(frecuencias_dniB.items()):
        print(f"   Dígito {digito}: {cuenta} vez/veces")
    print("-" * 30)

    # --- CONDICIONES ---
    print("\n--- Evaluación de Condiciones ---")
    # Condición1: Si un dígito aparece en todos los conjuntos
    if interseccion_result:
```

```
        print(f"Dígito(s) compartido(s): {interseccion_result}")
    else:
        print("No hay dígitos compartidos entre ambos DNIs.")

    # Condición2: Si algún conjunto tiene más de 6 elementos
    if len(conjunto_A_digitos) > 6 or len(conjunto_B_digitos) > 6:
        print("Diversidad numérica alta en al menos un DNI.")
    else:
        print("Ningún DNI tiene diversidad numérica alta (menos de 7 dígitos únicos).")
    print("-" * 30)


# la función 'inicio'
def inicio():
    print("0 - salir")
    print("1 - ingresar dos dni modo manual")
    print("2 - ingresar dos dni aleatorio")
    opcion = int(input("Elija opcion: "))

    dni_num1 = None # Inicializamos a None
    dni_num2 = None # Inicializamos a None
    conjunto_dni1 = None
    conjunto_dni2 = None

    if opcion == 1:
        str_dni1 = input("Ingrese DNI 1: ")
        str_dni2 = input("Ingrese DNI 2: ")
```

```
            dni_num1 = int(str_dni1) # Convertimos a int para las funciones de suma/frecuencia
        dni_num2 = int(str_dni2)
        conjunto_dni1 = set(map(int, str_dni1))
        conjunto_dni2 = set(map(int, str_dni2))
    elif opcion == 2:
        # dni_aleatorio devuelve (número, conjunto_de_dígitos)
        dni_num1, conjunto_dni1 = dni_aleatorio()
        dni_num2, conjunto_dni2 = dni_aleatorio()
        print(f"DNI 1 aleatorio generado: {dni_num1}")
        print(f"DNI 2 aleatorio generado: {dni_num2}")
    elif opcion == 0:
        return
    else:
        print("Opción no válida. Por favor, elija 1 o 2.")
        inicio()

      # Pasamos tanto los conjuntos de digitos unicos como los numeros de DNI originales
    if dni_num1 is not None and dni_num2 is not None:
            mostrar_resultados_conjuntos(conjunto_dni1, conjunto_dni2, dni_num1, dni_num2)

# TEST dni A = Bruno Pighin- DNI: 35940327; y dni Aldo Manfredi - DNI: 27028093;
inicio()
```

scriptparte2b.py

```
import datetime
def es_bisiesto(anio):
    return (anio % 4 == 0) and (anio % 100 != 0 or anio % 400 == 0)

def ingresar_anios_nacimiento():
    while True:
        anio1 = int(input("Ingrese año de nacimiento del integrante 1: "))
        anio2 = int(input("Ingrese año de nacimiento del integrante 2: "))
        return anio1, anio2

def verificar_bisiesto(anio1, anio2):
    if es_bisiesto(anio1) or es_bisiesto(anio2):
        print("\n---Tenemos un año especial, hay un bisiesto")

def calcular_producto_cartesiano(anios):
    anio_actual = datetime.datetime.now().year
    edades = [anio_actual - anio for anio in anios]

    print("\n--- Producto Cartesiano (Años x Edades) ---")
    for anio in anios:
        for edad in edades:
            print(f"({anio}, {edad})")

def verificar_grupo_z(anio1, anio2):
    if anio1 > 2000 and anio2 > 2000:
        print("\n---pertenece al Grupo Z")
```

```
def verificar_anios_iguales(anio1, anio2):
    if anio1 == anio2:
        print(f"\n---los dos nacieron mismo año... año: {anio1}")


def contar_anios_pares_impares(anio1, anio2):
    pares = 0
    impares = 0


    anios = [anio1, anio2]
    for anio in anios:
        if anio % 2 == 0:
            pares += 1
        else:
            impares += 1
    print(f"\n---Cantidad de años de nacimiento pares: {pares}")
    print(f"\n---Cantidad de años de nacimiento impares: {impares}")


def inicio():
    anio_integrante1, anio_integrante2 = ingresar_anios_nacimiento()
    anios_ingresados = [anio_integrante1, anio_integrante2]
    print(f"\nAños de nacimiento ingresados: {anio_integrante1}, {anio_integrante2}")
    verificar_bisiesto(anio_integrante1, anio_integrante2)
    calcular_producto_cartesiano(anios_ingresados)
    verificar_grupo_z(anio_integrante1, anio_integrante2)
    verificar_anios_iguales(anio_integrante1, anio_integrante2)
    contar_anios_pares_impares(anio_integrante1, anio_integrante2)
```

```
inicio()

# test:

# (2004, 2008) -> tres condiciones OK ...

# (2004, 1999) -> bisiesto y par (2 condiciones)...

# (1823, 1999) -> ninguna condicion
```