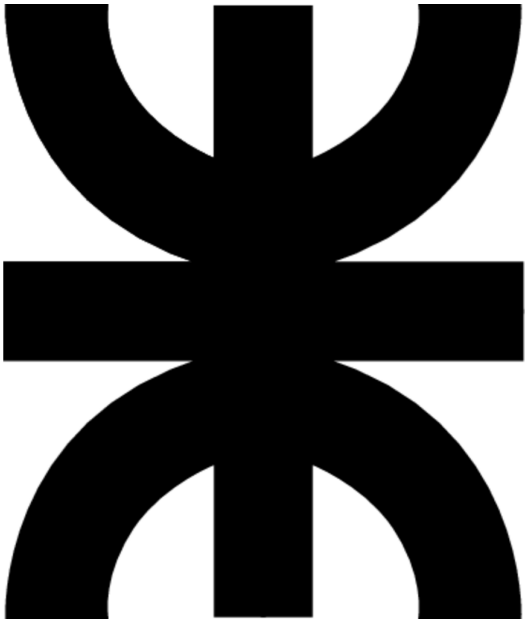
UNIVERSIDAD TECNOLÓGICA NACIONAL

TRABAJO INTEGRADOR 2 - Matemática y Programación

COMISIÓN 21 - GRUPO 4



INTEGRANTES

Nombre	E-mail
Mauro Gonzalo Santini	mgs.argentum@gmail.com
Cristian Serna	sernachristian700@gmail.com
Juan Martín Roques Zeballos	juanmartinroqueszeballos@gmail.com
Emanuel Facundo Ruidiaz	emafruidiaz@gmail.com

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.

Consignas

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.

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

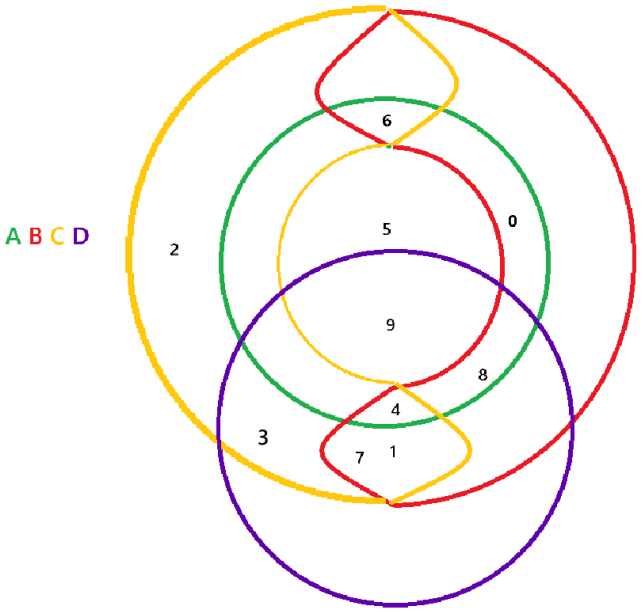
1. Archivo PDF con: desarrollo de conjuntos y operaciones, todos los diagramas de Venn, expresiones lógicas redactadas, y tareas de cada integrante explicadas por escrito.
2. Archivo con extensión .py que contenga el programa en Python.
3. Video grupal subido en lo posible a YouTube.
4. Documento adicional 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.

Desarrollo

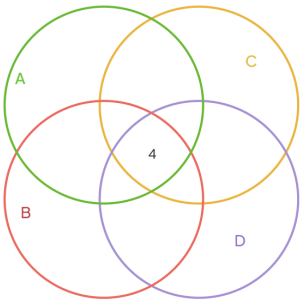
Parte 1 – Desarrollo Matemático (Conjuntos y Lógica)

1. DNI’s
    a. 38969450
    b. 40067818
    c. 32743164
    d. 94731883
2. Conjuntos
    a. A = {3,8,9,6,4,5,0}
    b. B = {4,0,6,7,8,1}
    c. C = {3,2,7,4,1,6}
    d. D = {9,4,7,3,1,8}
3. unión, intersección, diferencia (entre pares) y diferencia simétrica.
    a. Unión
    $A \cup B \cup C \cup D = \{0,1,2,3,4,5,6,7,8,9\}$
    b. Intersección
    $A \cap B \cap C \cap D = \{4\}$
    c. Diferencia entre pares
    A-A = B-B = C-C = D-D = {}
    A-B = {3,9,5}
    A-C = {8,9,5,0}
    A-D= {6,5,0}
    B-A={7,1}
    B-C={0,8}
    B-D={0,6}
    C-A={2,7,1}
    C-B={3,2}
    C-D={2,6}
    D-A={9,7,1}
    D-B={9,3}
    D-C={9,8}
    d. Diferencia simétrica
    A-B = {1,7,3,9,5}
    A-C = {8,9,5,0,1,2,7}
    A-D = {0,1,5,6,7}
    B-C = {0,2,3,8}
    B-D = {0,3,6,9}
    C-D = {2,6,8,9}
4. Diagramas de Venn

a. Unión

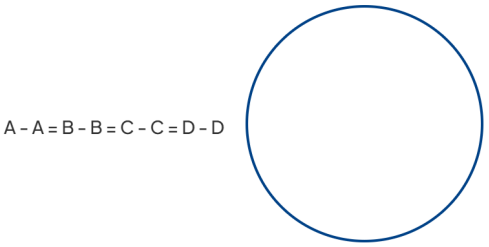


b. Intersección



c. Diferencia

i. A - A = B - B = C - C = D - D = {}

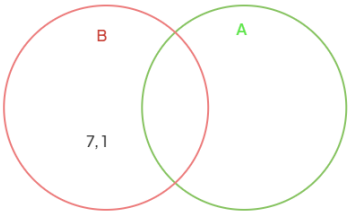


ii. A - B

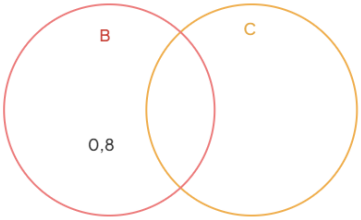
iii. A - C

iv. A - D

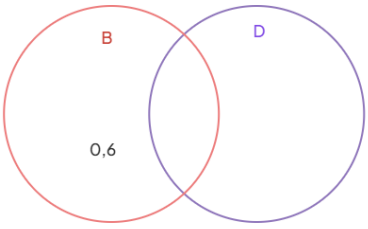
v. B - A



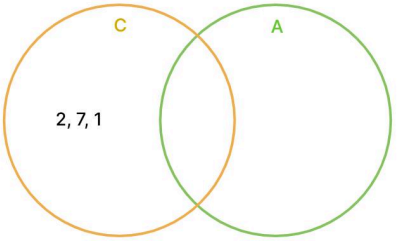
vi. B - C



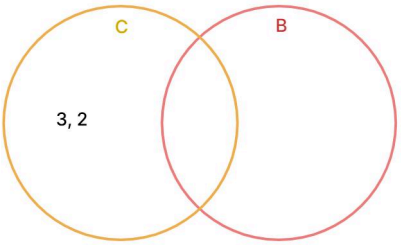
vii. B - D



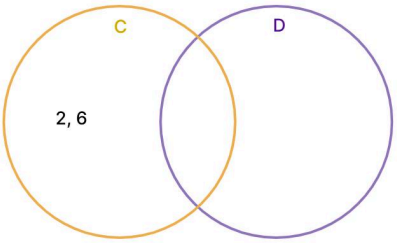
viii. C - A



ix. C - B

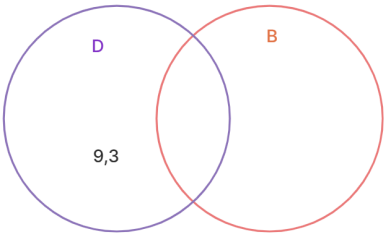


x. C - D

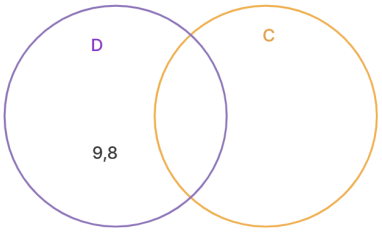


xi. D - A

xii. D - B

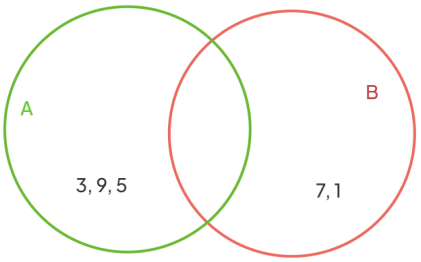


xiii. D - C

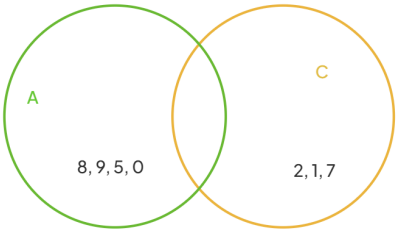


d. Diferencia simétrica

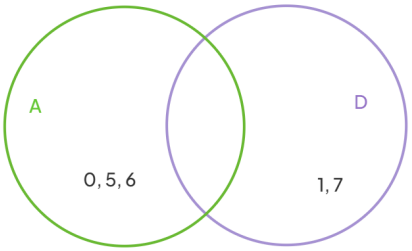
i. A-B



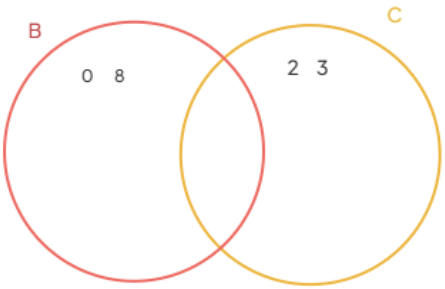
ii. A-C



iii. A-D

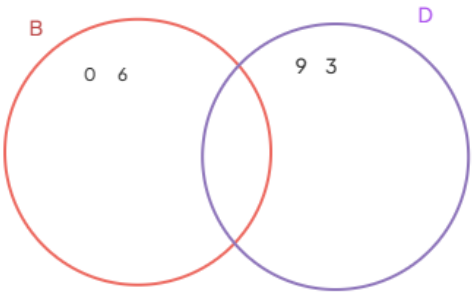


iv. B-C

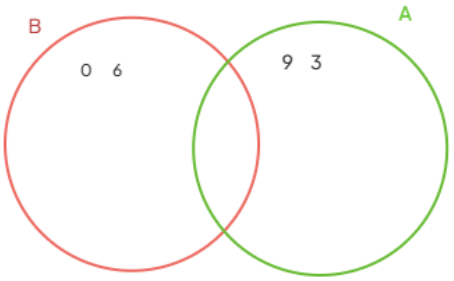


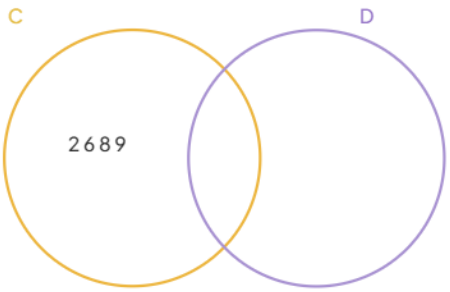
v. B-D

vi.



vii. C-D





5. Expresiones lógicas
    a. Si la intersección entre todos los conjuntos tiene exactamente un elemento, se considera un dígito representativo del grupo.
    b. Si algún dígito aparece en todos los conjuntos, se marca como dígito común

```
# Definición de los conjuntos
A = {3, 8, 9, 6, 4, 5, 0}
B = {4, 0, 6, 7, 8, 1}
C = {3, 2, 7, 4, 1, 6}
D = {9, 4, 7, 3, 1, 8}

# 1. Verificar si hay un dígito representativo (intersección con
exactamente 1 elemento)
interseccion_total = A & B & C & D

if len(interseccion_total) == 1:
    digito_representativo = interseccion_total.pop()
    print(f"El dígito representativo del grupo es:
{digito_representativo}")
```

```
else:
    print("No hay un único dígito representativo en todos los
conjuntos")

# 2. Encontrar todos los dígitos comunes (que aparecen en todos los
conjuntos)
digitos_comunes = A & B & C & D

if digitos_comunes:
    print(f"Dígitos comunes en todos los conjuntos: {digitos_comunes}")
else:
    print("No hay dígitos comunes en todos los conjuntos")
```

Parte 2 – Desarrollo del Programa en Python

A. Operaciones con DNIs

```
# Este programa utiliza varios métodos y funciones de Python para
realizar análisis sobre DNIs ingresados:
# - set(): Permite crear conjuntos de elementos únicos, útil para
operaciones como unión, intersección y diferencia.
# - enumerate(): Facilita la iteración sobre listas,
proporcionando un índice junto con cada elemento.
# - strip(): Elimina espacios en blanco o caracteres específicos
al inicio y al final de una cadena.
# - sum(): Calcula la suma de los elementos de un iterable, como
una lista o conjunto.

# Ingreso de DNIs
dnis = []
while True:
    dni = input("Ingrese un DNI (8 dígitos, sin puntos) o 'fin'
para terminar: ").strip()
    if dni.lower() == 'fin':
        break
    if not dni.isdigit() or len(dni) != 8:
        print("Error: El DNI debe tener exactamente 8 dígitos
numéricos.")
        continue
    dnis.append(dni)
```

```
if not dnis:
    print("No se ingresaron DNIs. Programa terminado.")
else:
    # Mostrar DNIs ingresados y sus dígitos únicos
    print("\nDNIs ingresados:")
    for i, dni in enumerate(dnis, 1):
        print(f"DNI {i}: {dni}")
        print(f"Dígitos únicos: {set(dni)}")

    # Operaciones con conjuntos
    conjuntos = [set(dni) for dni in dnis]

    # Verifica si hay al menos 2 conjuntos para realizar las
operaciones
    if len(conjuntos) >= 2:
        print("\nOperaciones de conjuntos:")
        # Itera sobre cada conjunto
        for i in range(len(conjuntos)):
            # Compara el conjunto actual con los siguientes
conjuntos
            for j in range(i+1, len(conjuntos)):
                # Imprime los DNIs que se están comparando
                print(f"\nDNI {i+1}: {dnis[i]}, DNI {j+1}:
{dnis[j]}")
                # Calcula y muestra la unión de los conjuntos
                print(f"Unión: {conjuntos[i] | conjuntos[j]}")
                # Calcula y muestra la intersección de los
conjuntos
                print(f"Intersección: {conjuntos[i] &
conjuntos[j]}")
                # Calcula y muestra la diferencia entre el primer
y segundo conjunto
                print(f"Diferencia (DNI{i+1} - DNI{j+1}):
{conjuntos[i] - conjuntos[j]}")
                # Calcula y muestra la diferencia entre el
segundo y primer conjunto
                print(f"Diferencia (DNI{j+1} - DNI{i+1}):
{conjuntos[j] - conjuntos[i]}")
                # Calcula y muestra la diferencia simétrica entre
los conjuntos
                print(f"Diferencia simétrica: {conjuntos[i] ^
conjuntos[j]}")
    else:
```

```
        print("\nSe necesitan al menos 2 DNIs para realizar
operaciones de conjuntos.")

    # Frecuencia de dígitos
    print("\nFrecuencia de dígitos:")
    # Itera sobre cada DNI ingresado
    for i, dni in enumerate(dnis, 1):
        # Inicializa un diccionario para contar la frecuencia de
cada dígito
        frecuencia = {}
        # Recorre cada dígito del DNI
        for digito in dni:
            # Incrementa el contador de frecuencia para el dígito
actual
            frecuencia[digito] = frecuencia.get(digito, 0) + 1

        # Imprime el DNI actual y su frecuencia de dígitos
        print(f"\nDNI {i}: {dni}")
        # Ordena los dígitos por clave y muestra cuántas veces
aparece cada uno
        for digito, count in sorted(frecuencia.items()):
            print(f"Dígito {digito}: {count} vez/veces")

    # Calcula y muestra la suma de los dígitos de cada DNI
    print("\nSuma de dígitos:")
    for i, dni in enumerate(dnis, 1):
        # Convierte cada dígito a entero y calcula la suma total
        suma = sum(int(d) for d in dni)
        # Imprime el DNI y la suma de sus dígitos
        print(f"DNI {i}: {dni} - Suma: {suma}")

    # Evalúa condiciones específicas para cada DNI
    print("\nEvaluación de condiciones:")
    for i, dni in enumerate(dnis, 1):
        # Obtiene el primer y último dígito del DNI
        primer_digito = dni[0]
        ultimo_digito = dni[-1]
        # Calcula la suma de los dígitos del DNI
        suma = sum(int(d) for d in dni)
        # Obtiene la longitud del DNI
        longitud = len(dni)
        # Verifica si el DNI contiene al menos un cero
        tiene_cero = '0' in dni
```

```
        # Verifica si todos los dígitos del DNI son pares
        todos_pares = all(int(d) % 2 == 0 for d in dni)

        # Imprime las condiciones evaluadas para el DNI actual
        print(f"\nDNI {i}: {dni}")
        print(f"- El primer dígito es {primer_digito}")
        print(f"- El último dígito es {ultimo_digito}")
        print(f"- La suma de todos sus dígitos es {suma}")
        print(f"- {'Tiene' if tiene_cero else 'No tiene'} al
menos un cero")
        print(f"- {'Todos' if todos_pares else 'No todos'} los
dígitos son pares")

        # Condiciones adicionales
        if primer_digito == ultimo_digito:
            print("- El primer y último dígito son iguales")
        if suma > 40:
            print("- La suma de dígitos es mayor que 40")
        if longitud == 8 and not tiene_cero:
            print("- Tiene exactamente 8 dígitos y ninguno es
cero")
```

B. Operaciones con años de nacimiento

```
# Función para determinar si un año es bisiesto
def es_bisiesto(year):
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

# Entrada de los años de nacimiento
cant_integrantes = int(input("Ingrese la cantidad de integrantes del
grupo: "))
anios = []

for i in range(cant_integrantes):
    anio = int(input(f"Ingrese el anio de nacimiento del integrante
{i + 1}: "))
    anios.append(anio)

# Contar años pares e impares
pares = 0
```

```python
impares = 0
anio_bisiesto_en_lista = False
todos_genz = True

for anio in anios:
    if anio % 2 == 0:
        pares += 1
    else:
        impares += 1
    if es_bisiesto(anio):
        anio_bisiesto_en_lista = True
    if anio <= 2000:
        todos_genz = False

# Mostrar resultados
print(f"Cantidad de anios pares: {pares}")
print(f"Cantidad de anios impares: {impares}")

# Muestro si son todos GenZ
if todos_genz:
    print("Grupo Z")

# Ver si hay algún año bisiesto
if anio_bisiesto_en_lista:
    print("Tenemos un anio especial")

# Calcular edades actuales asumiendo el año 2025
edades = [2025 - anio for anio in anios]

# Producto cartesiano entre años y edades
producto_cartesiano = []
for anio in anios:
    for edad in edades:
        producto_cartesiano.append((anio, edad))

print("Producto cartesiano entre anios y edades:")
print("{", end=" ")
for i in range(len(producto_cartesiano)):
    print(producto_cartesiano[i], end="")
    if i != len(producto_cartesiano) - 1:
        print(", ", end="")
print(" }")
```

Parte 3 – Video de Presentación

Video de presentación:

- TP Integrador 2 - Matemática y Programación