**MATEMÁTICA**

**TRABAJO PRACTICO INTEGRADOR N° 2**

**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.**

**Luca:** 43028752

**Exequiel:** 45881752

**Yoel:** 42695051

**Fabián:** 22219807

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

Por lo que se define a

Luca como **L={0,2,3,4,5,7,8}**

Exequiel como **E={1,2,4,5,7,8}**

Yoel como **Y={0,1,2,4,5,6,9}**

Fabián como **F={0,1,2,7,8,9}**

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

**UNION** Son todos los elementos sin repetir de los conjuntos definidos

**L={0,2,3,4,5,7,8}**

**E={1,2,4,5,7,8}**

**L** U **E ={0,1,2,3,4,5,7,8}**

y si hacemos los 4 conjuntos**:**

**L={0,2,3,4,5,7,8}**

**E={1,2,4,5,7,8}**

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

Entonces

**L** U **E** U **Y** U **F ={0,1,2,3,4,5,6,7,8,9}**

**INTERSECCION** Son todos los elementos que se repiten o elementos comunes de ambos conjuntos

Sean los conjuntos

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

**Y F = {0,1,2,9}**

**DIFERENCIA** Son los elementos no comunes entre ambos conjuntos

Sean los conjuntos

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

**La diferencia entre Y - F es:**

**Y - F = {4,5,6}**

**F -Y={7,8}**

**DIFERENCIA SIMETRICA** Son todos los elementos que pertenecen a uno de los conjuntos, sacando los compartidos

Sean los conjuntos definidos por

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

**La diferencia simétrica se define como:**

**Y Δ F = (Y - F)** U **(F -Y)**

**Y - F = {4,5,6}**

**F - Y = {7,8}**

**Por lo tanto**

**Y Δ F = {4,5,6,7,8}**

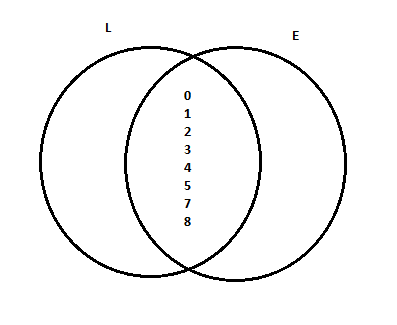
**4. Para cada una de estas operaciones, se debe realizar un diagrama de Venn (a mano o digital), que debe incluirse en la entrega.**

**UNION**

**L={0,2,3,4,5,7,8}**

**E={1,2,4,5,7,8}**

**L** U **E ={0,1,2,3,4,5,7,8}**

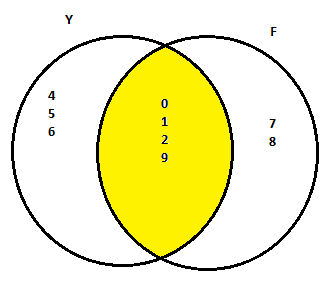


**INTERSECCION**

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

**Y F = {0,1,2,9}**

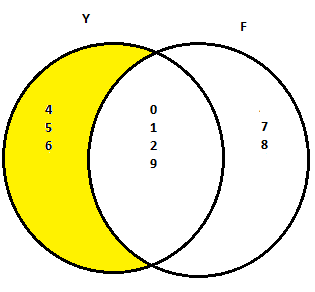


**DIFERENCIA**

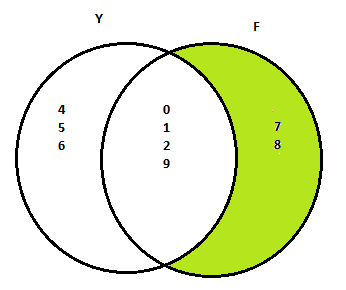
**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

**Y - F = {4,5,6}**



**F -Y={7,8}**

****

**DIFERENCIA SIMETRICA**

**Y={0,1,2,4,5,6,9}**

**F={0,1,2,7,8,9}**

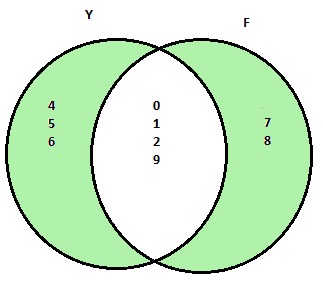
**Y Δ F = (Y - F)** U **(F -Y)**

**Y - F = {4,5,6}**

**F - Y = {7,8}**

**Por lo tanto**

**Y Δ F = {4,5,6,7,8}**

****

**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**

"Unir todos los elementos que están en el conjunto L o en el conjunto E, sin repetir ningún elemento."

**UNION**

**L={0,2,3,4,5,7,8}**

**E={1,2,4,5,7,8}**

**L** U **E ={0,1,2,3,4,5,7,8}**

union = L | E

# También se puede usar: union = L.union(E)

Dado:

L = {0, 2, 3, 4, 5, 7, 8}

E = {1, 2, 4, 5, 7, 8}

La unión de L y E (L | E) da como resultado el conjunto: {0, 1, 2, 3, 4, 5, 7, 8}

Este conjunto contiene todos los elementos presentes en L o en E, sin duplicados.

**Expresión lógica 2**

**INTERSECCION**

"Obtener los elementos que están tanto en el conjunto Y como en el conjunto F."

interseccion = Y & F

# También válido: interseccion = Y.intersection(F)

Dado:

Y = {0, 1, 2, 4, 5, 6, 9}

F = {0, 1, 2, 7, 8, 9}

La intersección entre Y y F (Y & F) es el conjunto {0, 1, 2, 9}, ya que estos elementos se encuentran en ambos conjuntos simultáneamente.

**Parte 2 - Desarrollo del programa en Python**

**Código fuente escrito en Python**

**# TPI 2 MATEMATICA: Análisis de DNIs y Años de nacimiento**

**from datetime import datetime**

**from itertools import product**

**# PARTE A: Análisis de DNIs:**

**# 1. Solicita los DNIs al usuario, valida que sean numéricos y los almacena en una lista.**

**def pedir\_dnis(): # Solicita DNIs al usuario y los almacena en una lista**

**cantidad = 0 # Almacena la cantidad de DNIs a ingresar**

**while cantidad < 1: # Asegura que se ingrese al menos un DNI y se inicia el bucle**

**try:**

**cantidad = int(input("Cantidad de DNIs a ingresar: ")) # Solicita la cantidad de DNIs**

**except ValueError: # Captura el error si no se ingresa un número válido**

**print("Debe ingresar un número válido.") # Si no es un número, solicita nuevamente**

**lista\_dnis = [] #Se crea la lista de DNIs**

**for i in range(cantidad): # Bucle para solicitar cada DNI, se asegura que el DNI sea un número y no esté vacío y si no es válido, solicita nuevamente**

**while True:**

**dni = input(f"DNI {i + 1}: ").strip() # Con .strip() eliminamos espacios al inicio y al final**

**if dni.isdigit(): # Verifica si el DNI es un numero**

**lista\_dnis.append(dni) # Si es un numero, lo agrega a la lista**

**break**

**else:**

**print("Solo se aceptan números.") # Si el DNI no es un número, solicita nuevamente**

**return lista\_dnis**

**# 2. Genera una lista de conjuntos, donde cada conjunto contiene los dígitos únicos de cada DNI.**

**def conjuntos\_digitos\_unicos(dnis):**

**return [set(map(int, dni)) for dni in dnis]**

**# 3. Muestra la unión, intersección, diferencia y diferencia simétrica entre los conjuntos de dígitos únicos de los DNIs.**

**def mostrar\_operaciones\_conjuntos(conjuntos):**

**# Unión de todos los conjuntos**

**union\_total = set().union(\*conjuntos)**

**print("\nUnión total:", union\_total)**

**# Intersección de todos los conjuntos**

**interseccion\_total = set.intersection(\*conjuntos)**

**print("Intersección total:", interseccion\_total)**

**# Diferencia entre pares de conjuntos**

**print("\nDiferencias entre conjuntos:") # Imprime un título para la sección de diferencias**

**for i, ci in enumerate(conjuntos): # Itera sobre cada conjunto con su índice**

**for j, cj in enumerate(conjuntos): # Itera nuevamente sobre cada conjunto con su índice**

**if i < j: # Solo compara cada par una vez y evita comparar un conjunto consigo mismo**

**diferencia = ci - cj # Calcula la diferencia de conjuntos (elementos en ci que no están en cj)**

**print(f"Conjunto {i+1} ({ci}) - Conjunto {j+1} ({cj}): {diferencia if diferencia else ' - (conjunto vacío)'}") # Muestra el resultado**

**# Diferencia simétrica entre pares de conjuntos**

**print("\nDiferencias simétricas:") # Imprime un título para la sección de diferencias simétricas**

**for i, ci in enumerate(conjuntos): # Itera sobre cada conjunto con su índice**

**for j, cj in enumerate(conjuntos): # Itera nuevamente sobre cada conjunto con su índice**

**if i < j: # Solo compara cada par una vez y evita comparar un conjunto consigo mismo**

**diferencia\_sim = ci ^ cj # Calcula la diferencia simétrica (elementos en ci o cj pero no en ambos)**

**print(f"Conjunto {i+1} ({ci}) - Conjunto {j+1} ({cj}): {diferencia\_sim if diferencia\_sim else ' - (conjunto vacío)'}") # Muestra el resultado**

**# 4. Cuenta y muestra la frecuencia de cada dígito (0-9) en todos los DNIs ingresados.**

**def frecuencia\_digitos(dnis): # Define una función que recibe la lista de DNIs**

**conteo = [0]\*10 # Inicializa una lista de 10 ceros para contar cada dígito del 0 al 9**

**for dni in dnis: # Itera sobre cada DNI en la lista**

**for d in dni: # Itera sobre cada dígito del DNI (como string)**

**conteo[int(d)] += 1 # Convierte el dígito a entero y suma 1 en la posición correspondiente**

**print("\nFrecuencia de dígitos:") # Imprime un título para la sección de frecuencias**

**for i, cant in enumerate(conteo): # Itera sobre cada dígito y su cantidad**

**print(f"Dígito {i}: {cant} veces") # Muestra cuántas veces apareció cada dígito**

**# 5. Calcula y muestra la suma total de todos los dígitos de todos los DNIs.**

**def suma\_total\_digitos(dnis): # Define una función que recibe la lista de DNIs**

**total = sum(int(d) for dni in dnis for d in dni) # Suma todos los dígitos de todos los DNIs**

**print("\nSuma total de todos los dígitos:", total) # Imprime el resultado**

**# 6. Evalúa si hay valores comunes en todos los conjuntos y si el grupo es equilibrado entre conjuntos de tamaño par e impar.**

**def evaluar\_valores\_y\_equilibrio(conjuntos): # Define una función que recibe la lista de conjuntos**

**# Intersección entre todos los conjuntos**

**comunes = set.intersection(\*conjuntos) # Calcula la intersección de todos los conjuntos (elementos comunes)**

**if comunes: # Si hay elementos comunes**

**print("\nValores presentes en todos los conjuntos:", comunes)**

**else:**

**print("\nNo hay valores comunes en todos los conjuntos.")**

**# Comparación entre cantidad de conjuntos con tamaño par e impar**

**pares = sum(1 for c in conjuntos if len(c) % 2 == 0)**

**impares = len(conjuntos) - pares**

**if pares == impares:**

**print("Grupo equilibrado entre pares e impares.")**

**else:**

**print("Grupo NO equilibrado.")**

**# PARTE B: Años de nacimiento**

**# 7. Determina si un año dado es bisiesto.**

**def es\_bisiesto(anio):**

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

**# 8. Solicita los años de nacimiento, calcula estadísticas y muestra el producto cartesiano entre años y edades.**

**def procesar\_anios():**

**cantidad = int(input("\nCantidad de integrantes: "))**

**anios = []**

**for i in range(cantidad):**

**anio = int(input(f"Año de nacimiento {i+1}: "))**

**anios.append(anio)**

**# Estadísticas**

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

**impares = cantidad - pares**

**hay\_bisiesto = any(es\_bisiesto(a) for a in anios)**

**todos\_z = all(a > 2000 for a in anios)**

**# Resultados**

**print(f"\nNacidos en año par: {pares}")**

**print(f"Nacidos en año impar: {impares}")**

**if todos\_z:**

**print("Grupo Z")**

**if hay\_bisiesto:**

**print("Al menos un año bisiesto presente")**

**# Cálculo de edades**

**edades = [datetime.now().year - a for a in anios]**

**# Producto cartesiano entre años y edades**

**print("\nProducto cartesiano entre años y edades:")**

**for par in product(anios, edades):**

**print(par)**

**# --- Ejecución principal --- #**

**if \_\_name\_\_ == "\_\_main\_\_":**

**# Parte A: análisis de DNIs**

**dnis = pedir\_dnis()**

**conjuntos = conjuntos\_digitos\_unicos(dnis)**

**mostrar\_operaciones\_conjuntos(conjuntos)**

**frecuencia\_digitos(dnis)**

**suma\_total\_digitos(dnis)**

**evaluar\_valores\_y\_equilibrio(conjuntos)**

**# Parte B: análisis de años de nacimiento**

**procesar\_anios()**