TRABAJO PRÁCTICO INTEGRADOR II
Matemática y Programación

Alumnos - Comisión N°: 6

Alex Nahuel Austin
Leonel Jesús Aballay
Agustin Maldonado
Cristian Gabriel Aguirre
Gino Paolo Canevaro

Profesor:

Ethel Carina R. Jovanovich

Tutor:

Fernando Marti

INTRODUCCIÓN

Se pondrán en práctica todo lo aprendido del módulo 3 y 4 (Conjuntos y Lógica) en este trabajo, en donde todos los integrantes del grupo asumirá su responsabilidad a desarrollar para obtener un buen resultado, se fomentará el compañerismo, colaboración e integración para los próximos proyectos que nos espera como futuros profesionales en programación.

DIVISIÓN DE TAREAS

1. Alex Nahuel Austin: Desarrollo Conjuntos
2. Leonel Jesús Aballay: Lógica caso práctico y edición
3. Agustín Maldonado: Programa Py. DNI
4. Cristian Gabriel Aguirre Programa Py. Años de nacimientos
5. Gino Paolo Canevaro: Desarrollo Logica

OBJETIVOS

- Generar habilidades de comunicación, colaboración
- Asumir responsabilidades específicas en donde cada alumno aportará sus conocimientos
- Determinar roles y funciones en el trabajo.

NÚMERO DE DOCUMENTOS DE LOS INTEGRANTE

- **ALEX:** 42.209.093
- **LEONEL:** 36.185.972
- **GINO:** 47.527.332
- **CHRISTIAN:** 30.975.414
- **AGUSTÍN:** 44.347.575

LETRA QUE VA A REPRESENTAR A CADA INTEGRANTE

- **ALEX** = **A**
- **LEONEL** = **L**
- **GINO:** = **G**
- **CHRISTIAN** = **C**
- **AGUSTÍN** = **M**

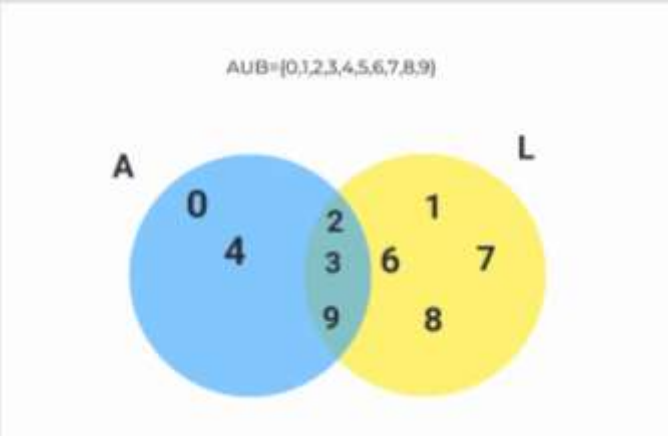
CONJUNTOS

DEFINICIÓN: un conjunto es una colección bien definida de elementos, utilizado para agrupar objetos o valores. Y se representan con letras mayúsculas.

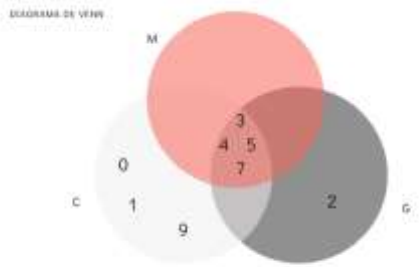
- ➔ **A** = {0,2,3,4,9}
- ➔ **L** = {1,2,3,5,6,7,8,9}
- ➔ **G** = {2,3,4,5,7}
- ➔ **C** = {0,1,3,4,5,7,9}
- ➔ **M** = {3,4,5,7}

OPERACIONES CON CONJUNTOS

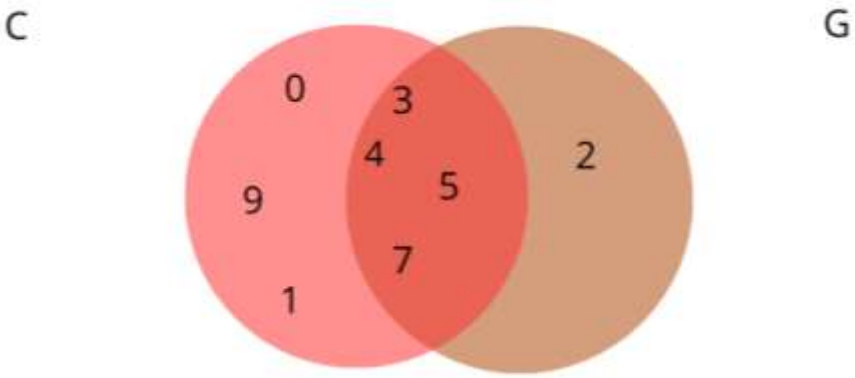
- **UNIÓN:**Es la unión de todos los elementos de dos o más conjuntos.
  - ➔ A={0,2,3,4,9}
  - ➔ L = {1,2,3,5,6,7,8,9}
  - ➔ A U L={0,1,2,3,4,5,6,7,8,9}



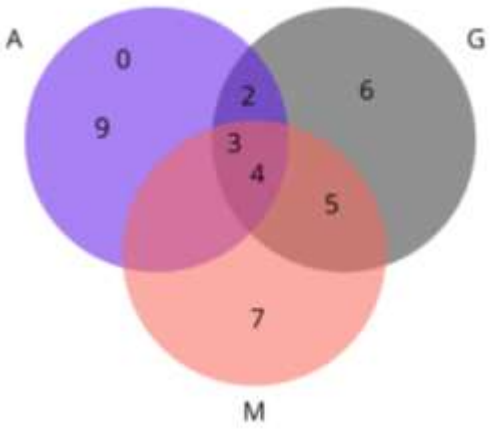
- ➔ G = {2,3,4,5,7}
- ➔ C = {0,1,3,4,5,7,9}
- ➔ M = {3,4,5,7}
- ➔ G U C U M = {0,1,2,3,4,5,7,9}



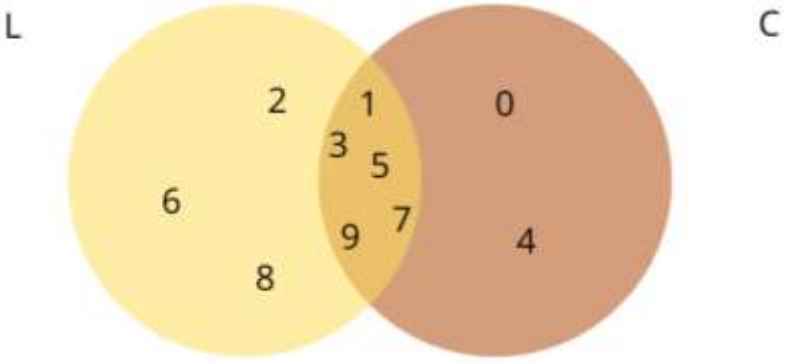
- **INTERSECCIÓN:** Son dos o más conjuntos que comparten los mismos elementos.
  - ➔ G = {2,3,4,5,7}
  - ➔ C = {0,1,3,4,5,7,9}
  - ➔ G ∩ C ={3,4,5,7}



  - ➔ A = {0,2,3,4,9}
  - ➔ G = {2,3,4,5,7}
  - ➔ M = {3,4,5,7}
  - ➔ A ∩ G ∩ M = {3,4}



- **DIFERENCIA:** La diferencia de dos o más conjuntos son los elementos que encuentra solo en un conjunto.
  - ➔ L = {1,2,3,5,6,7,8,9}
  - ➔ C = {0,1,3,4,5,7,9}
  - ➔ L - C ={2,6,8}



- **DIFERENCIA SIMÉTRICA:** La diferencia simétrica entre dos conjuntos es el conjunto de elementos que están en uno de los dos conjuntos, pero no en ambos.

  ➔ A = {0,2,3,4,9}

  ➔ G = {2,3,4,5,7}

  ➔ A Δ G = {0,5,7,9}

LÓGICA

INTEGRANTE	NOMBRE	DNI	CONJUNTOS ÚNICOS
A	Alex	42209093	{0, 2, 3, 4, 9}
L	Leonel	36185972	{1, 2, 3, 5, 6, 7, 8, 9}
G	Gino	47527332	{2, 3, 4, 5, 7}
C	Christian	30975414	{0, 1, 3, 4, 5, 7, 9}
M	Agustín	44347575	{3, 4, 5, 7}

1. **Dígito común en el grupo**

   Todo integrante del grupo, el 3 pertenece a todos los integrantes

   Proposición lógica: $\forall x \in \{A, L, G, C, M\}, 3 \in x$

(Para todo x en el conjunto {A, L, G, C, M}, el número 3 pertenece a x)

2. **Que DNI terminen en número Impar**

Integrante	Último Dígito	Par/Impar
A	3	Impar
L	2	par
G	2	par
C	4	par
M	5	impar

- Par = L,G,C
- IMPAR = A,M

Proposición Lógica: ∃x ∈ {A,M} | DNI es impar

(Existe al menos un x en el conjunto {A,M} tal que su DNI termina en impar

3. **Integrantes con el dígito 0 en su DNI**

INTEGRANTE	VALOR
A	Verdadero
L	Falso
G	Falso
C	Verdadero
M	Falso

Proposición Lógica: Tiene 0 (x) **<->** 0∈X

(x tiene 0 si y sólo si el 0 pertenece a x)

Propiedades definidas

Símbolo Significado

1. P(x) = Ejercicio 1
2. Q(x) = Ejercicio 2
3. R(x) = Ejercicio 3

TABLA DE VERDAD

Integrante	P(x):3∈ x	Q(x): DNI Impar	R(x): **0 ∈ x**
A	V	V	V
L	V	F	F
G	V	F	F
C	V	F	V
M	V	V	F

EJEMPLO EJERCICIO 2 LÓGICA: NÚMEROS PARES O IMPARES

```
# Diccionario que almacena las claves y sus valores
alumnos = {
    "ALEX": 42209093,
    "LEONEL": 36185972,
    "GINO": 47527332,
```

```
    "CHRISTIAN": 30975414,
    "AGUSTÍN": 44347575
}

# la función que determina si es par o impar
def es_impar(dni):
    return dni % 2 != 0

# Ciclo for para evaluar cada DNI de los alumnos y determinar cuales son par e impar
for nombre, dni in alumnos.items(): # items devuelve pares clave-valor (nombre y el dni)
    print(f"{nombre}: DNI = {dni} > > >  {'Impar (True)' if es_impar(dni) else 'Par (False)'}") # imprime si es par o impar usando el formato F-strings, llama a es_impar
    # si es impar devuelve True, sino False
```

EJERCICIO OPERACIONES CON DNI

```
# Ingreso de los DNIs (reales o ficticios).
def ingreso_dni():
    lista = []
    continuar = "s"
    while continuar == "s" or continuar == "S":
        documento = int(input("Ingrese un DNI: "))
        lista.append(documento)
        continuar = input("¿Desea continuar? S o N: ").strip()
```

```
    return lista

# documentos= [42209093, 36185972, 47527332, 30975414, 44347575]

# Generación automática de los conjuntos de dígitos únicos.
def digitos_unicos(lista):
    lista_conjuntos=[]
    for i in lista:
        conjunto = set()
        for d in str(i):
            conjunto.add(int(d))
        lista_conjuntos.append(conjunto)
    return lista_conjuntos

# Cálculo y visualización de: unión, intersección, diferencias y diferencia simétrica.
def vista_conjuntos(listadeconjuntos):
    conjunto_resultados = {"union": set(),
                           "interseccion": set(),
                           "diferencias": set(),
                           "diferencia simetrica": set()
                           }
    conjunto_resultados["union"] = set.union(*listadeconjuntos)
    conjunto_resultados["interseccion"] = set.intersection(*listadeconjuntos)
    conjunto_resultados["diferencias"] = diferencia(listadeconjuntos)
    conjunto_resultados["diferencia simetrica"] =
diferencia_simetrica(listadeconjuntos)
    print("Unión:", conjunto_resultados["union"])
    print("Intersección:", conjunto_resultados["interseccion"])
```

```
    print("Diferencias:", conjunto_resultados["diferencias"])
    print("Diferencia Simétrica:", conjunto_resultados["diferencia simetrica"])
    print("_" * 30)

def diferencia(listaconjuntos):
    acumulador = listaconjuntos[0]
    for i in range(1, len(listaconjuntos)):
        acumulador = acumulador - listaconjuntos[i]
    return acumulador

def diferencia_simetrica(listaconjuntos):
    acumulador = listaconjuntos[0]
    for i in range(1, len(listaconjuntos)):
        acumulador = acumulador ^ listaconjuntos[i]
    return acumulador


# Conteo de frecuencia de cada dígito en cada DNI utilizando estructuras repetitivas.
def frecuencia(lista_dnis):
    for dni in lista_dnis:
        print(f"DNI: {dni}")
        print("Frecuencia de dígitos:")
        conteo = {}  # diccionario para contar los dígitos
        for digito in str(dni):
            if digito in conteo:
                conteo[digito] += 1
            else:
                conteo[digito] = 1
```

```
        for digito, cantidad in conteo.items():
            if cantidad == 1:
                veces = "vez"
            else:
                veces = "veces"
            print(f"   Dígito {digito} → {cantidad} {veces}")
        print("-" * 30)

# Suma total de los dígitos de cada DNI.
def suma_digitos(lista_dnis):
    for dni in lista_dnis:
        suma = 0
        for digito in str(dni):
            suma += int(digito)
        print(f"Suma de los dígitos del DNI {dni}: {suma}")
    print("_" * 30)

# Evaluación de condiciones lógicas (condicionales), vinculadas con las expresiones escritas.
def condiciones_logicas(listaconjuntos):
    digitos_compartidos = set.intersection(*listaconjuntos)

    if digitos_compartidos:
        print(f"Dígito(s) compartido(s) en todos los conjuntos: {digitos_compartidos}")
    else:
        print("No hay dígitos compartidos en todos los conjuntos.")

    hay_diversidad_alta = False
```

```
    for conjunto in listaconjuntos:
        if len(conjunto) > 6:
            hay_diversidad_alta = True

    if hay_diversidad_alta:
        print("Diversidad numérica alta: algún conjunto tiene más de 6 elementos.")
    else:
        print("No hay diversidad numérica alta.")
    if len(digitos_compartidos) == 1:
        print("Hay un dígito representativo del grupo (intersección con un solo elemento).")



dnis_ingresados = ingreso_dni()
conjuntos_generados = digitos_unicos(dnis_ingresados)
vista_conjuntos(conjuntos_generados)
frecuencia(dnis_ingresados)
suma_digitos(dnis_ingresados)
condiciones_logicas(conjuntos_generados)
```

OPERACIONES CON AÑO DE NACIMIENTO

```
from datetime import datetime
from itertools import product

# Función que determina si un año es bisiesto
def es_bisiesto(anio):
    return (anio % 4 == 0 and anio % 100 != 0) or (anio % 400 == 0)
```

```
# Pedimos cuántas personas hay en el grupo
cantidad = int(input("¿Cuántas personas hay en el grupo? "))

# Creamos una lista para guardar los años de nacimiento
anios = []

# Ingresamos los años uno por uno
for i in range(cantidad):
    anio = int(input(f"Ingresá el año de nacimiento del integrante {i + 1}: "))
    anios.append(anio)

# Contamos pares e impares
pares = 0
impares = 0
for anio in anios:
    if anio % 2 == 0:
        pares += 1
    else:
        impares += 1

print(f"\nCantidad de años pares: {pares}")
print(f"Cantidad de años impares: {impares}")

# Verificamos si todos nacieron después del 2000
if all(anio > 2000 for anio in anios):
    print("Grupo Z")

# Verificamos si alguno nació en un año bisiesto
if any(es_bisiesto(anio) for anio in anios):
```

```
    print("Tenemos un año especial")

# Calculamos edades actuales
anio_actual = datetime.now().year
edades = [anio_actual - anio for anio in anios]

# Mostramos las edades
print(f"\nEdades actuales: {edades}")

# Producto cartesiano entre años y edades
print("\nProducto cartesiano (año, edad):")
for combinacion in product(anios, edades):
    print(combinacion)
```

CONCLUSIÓN

Las matemáticas son fundamentales en la programación, ya que nos ayudan a pensar de forma lógica y nos dan herramientas para desarrollarlas.

Muchos algoritmos, como el que usamos para verificar si un número es par o impar, se basan en operaciones matemáticas básicas, las matemáticas nos permiten que a la hora de trabajar en programación, nuestros proyectos sean precisos y eficientes. Nos ayudan con datos, fórmulas, manejar condiciones y tomar decisiones dentro de un programa.

En resumen, sin matemáticas, la programación perdería gran parte de su poder, porque ambas disciplinas se complementan. las matemáticas nos enseñan a resolver problemas y con la programación a automatizar las soluciones.