Tecnicatura Universitaria en Programación

Matemática

Profesora: Martina Wallace

Trabajo Semana de Integración II

Alumnos:

Agustín Emiliano Sotelo Carmelich (agustinemiliano22@gmail.com)

Bruno Giuliano Vapore (brunogvapore@gmail.com)

Fecha de Entrega: 13/06/2025

Índice

Parte 1 – Desarrollo Matemático (Conjuntos y Lógica)

1. Cada integrante debe anotar su número de DNI.

DNI de Agustin: **38.416.852**
DNI de Bruno: **35.829.366**

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

Conjunto de Agustin: **A = { 1, 2, 3, 4, 5, 6, 8 }**
Conjunto de Bruno: **B = { 2, 3, 5, 6, 8, 9 }**

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

- Unión
  A ∪ B = { 1, 2, 3, 4, 5, 6, 8, 9 }

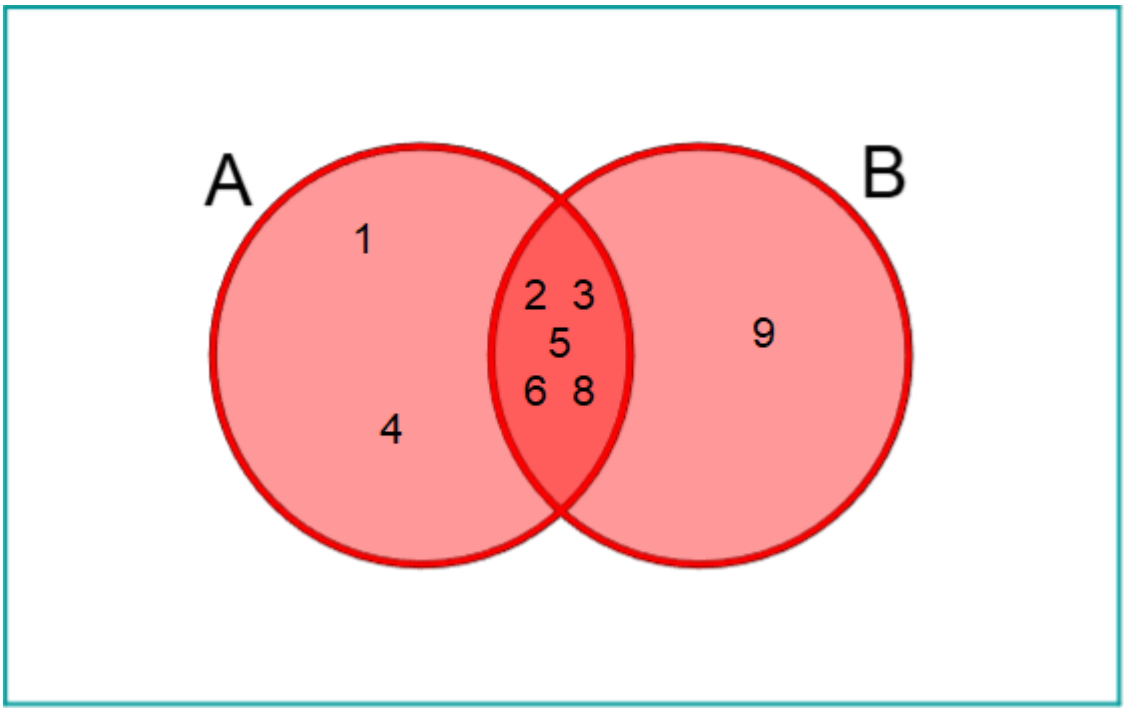
- Intersección
  A ∩ B = { 2, 3, 5, 6, 8 }

- Diferencia
  A - B = { 1, 4 }
  B - A = { 9 }

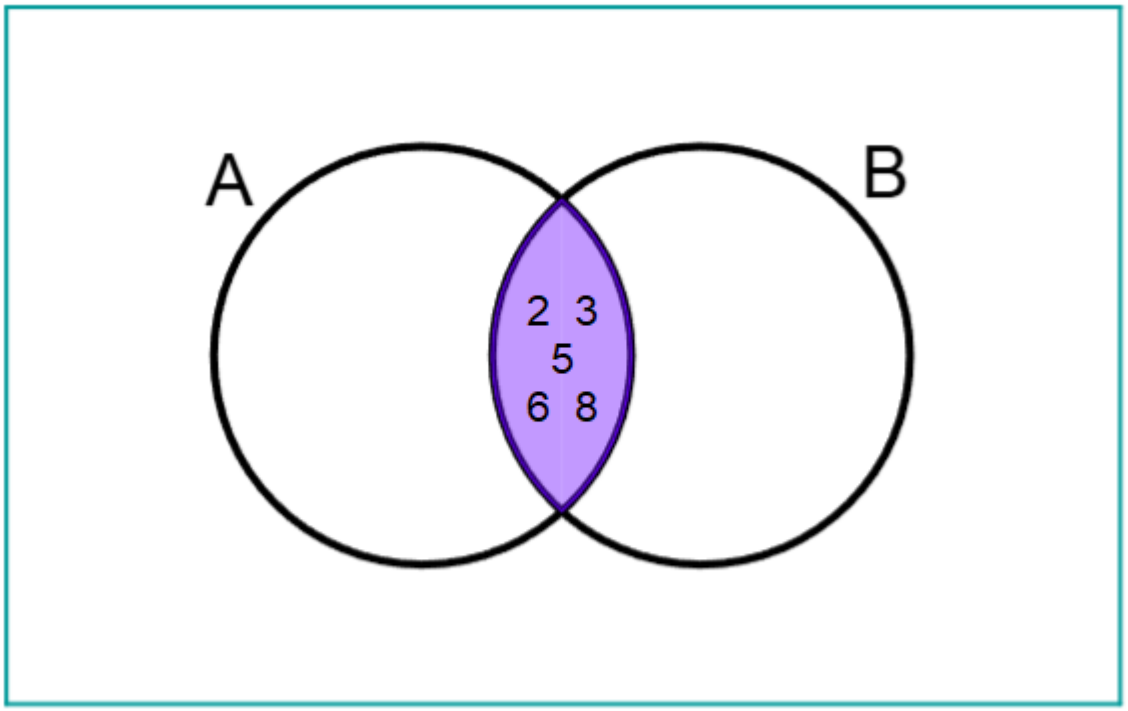
- Diferencia simétrica
  A Δ B = { 1, 4, 9 }

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

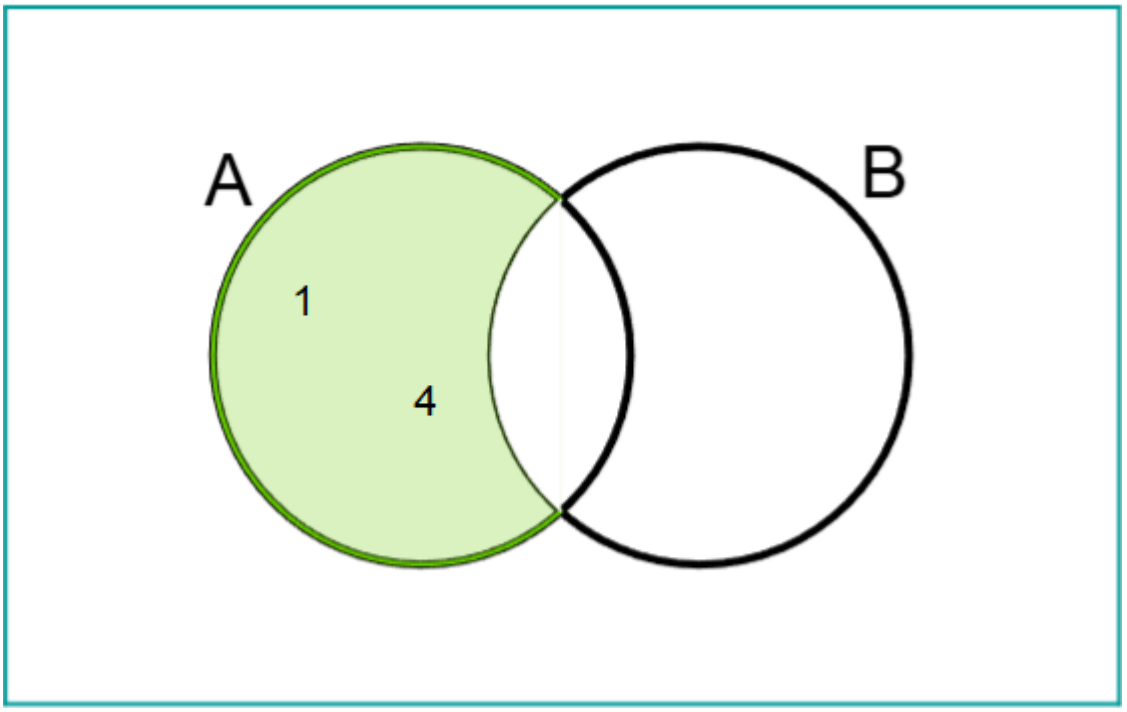
- **Unión: A ∪ B**



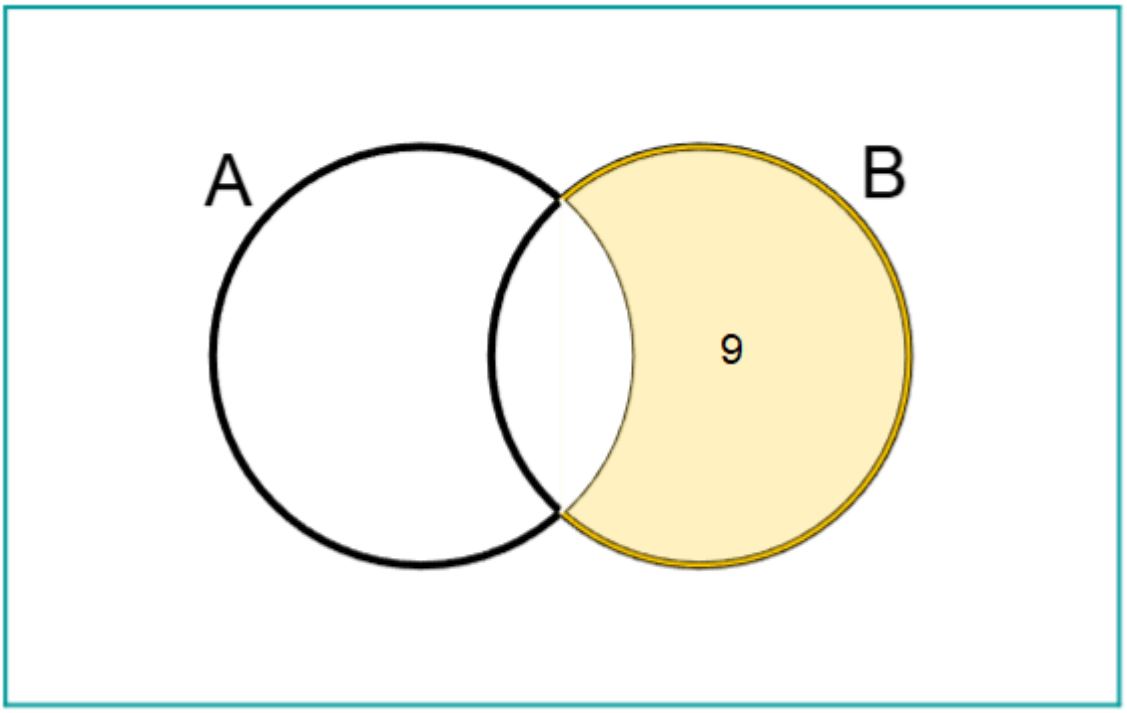
- **Intersección: A ∩ B**



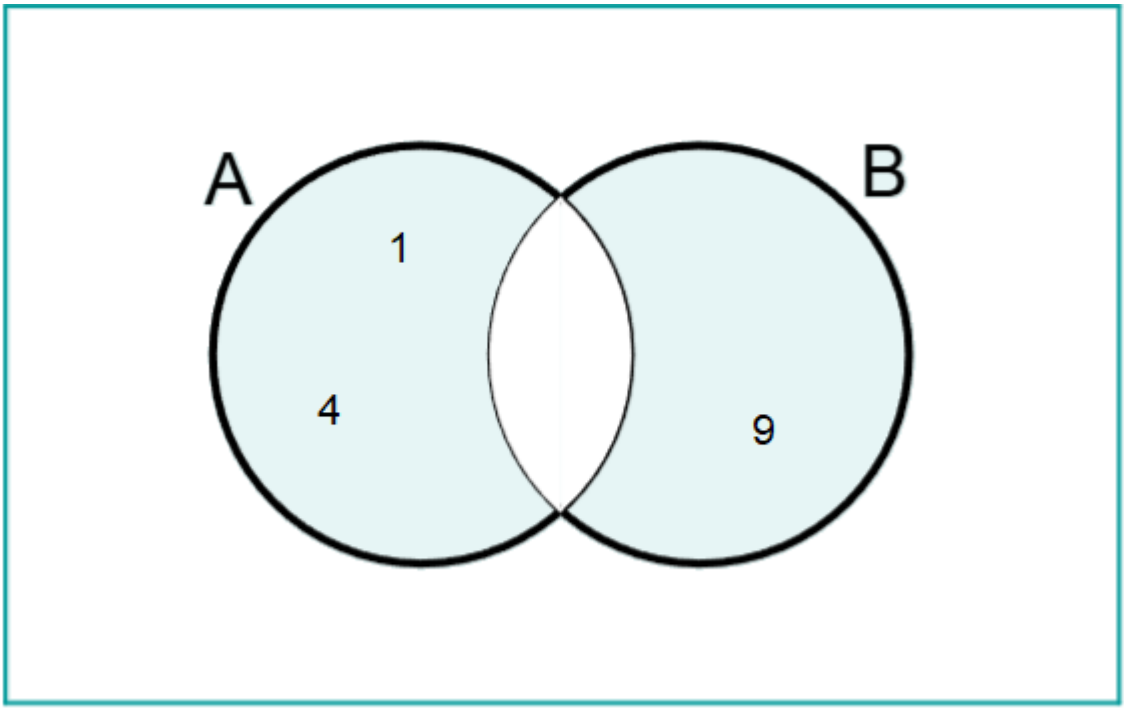
- **Diferencia: A - B**



Diferencia: B - A



- **Diferencia simétrica: A Δ B**



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 N°1:**
  “Hay al menos un dígito en el conjunto A que se repite en el conjunto B”
  Resultado: Verdadero
  R = { 2, 3, 5, 6, 8 }

Esta expresión se relaciona con la unión de los conjuntos A y B, ya que los elementos que pertenecen a ambos conjuntos se pueden obtener como resultado de esa operación. Indicamos como conjunto R ese resultado y el enunciado es verdadero en este caso, hay cinco elementos que se repiten.

- **Expresión N°2:**
  "El mayor dígito del conjunto B no supera al mayor dígito del conjunto A"
  Resultado = Falso
  R = { 9 }

Para esta expresión primero hay que identificar el elemento cuyo valor sea el mayor de los elementos que pertenezcan a B; y lo mismo para el conjunto A. Una vez encontrados se comparan los valores y se define si el enunciado es verdadero o falso. Para nuestro caso, el mayor elemento del conjunto B es el 9 y el mayor elemento del conjunto A es el 8. Entonces, el enunciado es falso porque 9 si supera a 8. Identificamos como elemento del conjunto R el mayor dígito de B.

Parte 2 – Desarrollo del Programa en Python

A. Operaciones con DNIs

- Ingreso de los DNIs (reales o ficticios).

```
dni_1 = input("Ingrese el primer DNI: ")
dni_2 = input("Ingrese el segundo DNI: ")

print(f"DNI: {dni_1}")
print(f"DNI: {dni_2}\n")
```

```
Ingrese el primer DNI: 38416852
Ingrese el segundo DNI: 35829366
DNI: 38416852
DNI: 35829366
```

- Generación automática de los conjuntos de dígitos únicos.

```
conjuntoDni1 = set(dni_1)
conjuntoDni2 = set(dni_2)

print(f"Conjunto de digitos únicos del DNI N°1: {conjuntoDni1}")
print(f"Conjunto de digitos únicos del DNI N°2: {conjuntoDni2}\n")
```

```
Conjunto de digitos únicos del DNI Nº1: {'8', '1', '5', '3', '6', '4', '2'}
Conjunto de digitos únicos del DNI Nº2: {'8', '5', '3', '6', '9', '2'}
```

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

```python
print(f"Siendo los conjuntos A = {conjuntoDni1} y B = {conjuntoDni2}")

# UNIÓN
union = conjuntoDni1.union(conjuntoDni2)
print(f"A U B = {union}")

# INTERSECCIÓN
inter = conjuntoDni1 & conjuntoDni2
print(f"A ∩ B = {inter}")

# DIFERENCIAS
difA_B = conjuntoDni1.difference(conjuntoDni2)
difB_A = conjuntoDni2.difference(conjuntoDni1)
print(f"A - B = {difA_B}")
print(f"B - A = {difB_A}")

# DIFERENCIA SIMETRICA
difSimetrica = conjuntoDni1 ^ conjuntoDni2
print(f"A Δ B = {difSimetrica}\n")
```

```
Siendo los conjuntos A = {'8', '1', '5', '3', '6', '4', '2'} y B = {'8', '5', '3', '6', '9', '2'}
A ∪ B = {'8', '1', '5', '3', '6', '4', '9', '2'}
A ∩ B = {'8', '5', '3', '6', '2'}
A - B = {'4', '1'}
B - A = {'9'}
A Δ B = {'4', '9', '1'}
```

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

```python
def frecuencia(dni):
    listaContadores = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    for digito in str(dni):
        listaContadores[int(digito)] += 1
    return (f"Cero: {listaContadores[0]}\nUno: {listaContadores[1]}\nDos: {listaContadores[2]}\nTres: {listaContadores[3]}\nCuatro: {listaCont

print(f"En el primer DNI la frecuencia de cada dígito es:\n{frecuencia(dni_1)}")
print(f"En el segundo DNI la frecuencia de cada dígito es:\n{frecuencia(dni_2)}\n")
```

```
En el primer DNI la frecuencia de cada dígito es:
Cero: 0
Uno: 1
Dos: 1
Tres: 1
Cuatro: 1
Cinco: 1
Seis: 1
Siete: 0
Ocho: 2
Nueve: 0
```

```
En el segundo DNI la frecuencia de cada dígito es:
Cero: 0
Uno: 0
Dos: 1
Tres: 2
Cuatro: 0
Cinco: 1
Seis: 2
Siete: 0
Ocho: 1
Nueve: 1
```

- Suma total de los dígitos de cada DNI.

```python
sumaDni_1 = 0
sumaDni_2 = 0

for i in dni_1:
    sumaDni_1  += int(i)
print(f"La suma de los dígitos de {dni_1} es: {sumaDni_1}")

for i in dni_2:
    sumaDni_2  += int(i)
print(f"La suma de los dígitos de {dni_2} es: {sumaDni_2}\n")
```

```
La suma de los dígitos de 38416852 es: 37
La suma de los dígitos de 35829366 es: 42
```

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

```python
#   Expresión N°1:
print("Hay al menos un dígito en el conjunto A que se repite en el conjunto B")

if conjuntoDni1 & conjuntoDni2:
    print(f"Verdadero. Hay al menos un número que se repite: {conjuntoDni1 & conjuntoDni2}\n")
else:
    print(f"Falso. No se repite ningún número\n")

#   Expresión N°2:
print("El mayor dígito del conjunto B no supera al mayor dígito del conjunto A")

maxDni1 = max(conjuntoDni1)
maxDni2 = max(conjuntoDni2)
if maxDni2 > maxDni1:
    print(f"Falso. El mayor número del conjunto B es {maxDni2} y del conjunto A es {maxDni1}. Por lo tanto {maxDni2} > {maxDni1}")
else:
    print(f"Verdadero. El mayor número del conjunto B es {maxDni2} y del conjunto A es {maxDni1}. Por lo tanto {maxDni2} < {maxDni1}")
```

```
Hay al menos un dígito en el conjunto A que se repite en el conjunto B
Verdadero. Hay al menos un número que se repite: {'8', '5', '3', '6', '2'}

El mayor dígito del conjunto B no supera al mayor dígito del conjunto A
Falso. El mayor número del conjunto B es 9 y del conjunto A es 8. Por lo tanto 9 > 8
```

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).

```python
anio_nacimiento_1 = int(input("Ingrese el primer año de nacimiento: "))
anio_nacimiento_2 = int(input("Ingrese el segundo año de nacimiento: "))

print(f"\n1er Año de nacimiento: {anio_nacimiento_1}")
print(f"\n2do Año de nacimiento: {anio_nacimiento_2}")
```

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

```
lista_anios_nacimiento = [anio_nacimiento_1, anio_nacimiento_2]

contador_pares = 0
contador_impares = 0

for anio in lista_anios_nacimiento:
    if anio % 2 == 0:
        contador_pares += 1
    else:
        contador_impares += 1

print(f"\n-- Analisis de años --")
print(f"Cantidad de años pares: {contador_pares}")
print(f"Cantidad de años impares: {contador_impares}")
```

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

```
if anio_nacimiento_1 > 2000 and anio_nacimiento_2 > 2000:
    print("Pertenecen - Grupo Z")
```

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

```
if es_bisiesto(anio_nacimiento_1) or es_bisiesto(anio_nacimiento_2):
    print("Nota: Tenemos un año especial!")
```

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

```
print("\n-- Comprobador de año Bisiesto --")

anio_a_verificar = int(input("Ingrese un año para determinar si es bisiesto: "))

if es_bisiesto(anio_a_verificar):
    print(f"El año {anio_a_verificar} es Bisiesto!")
else:
    print(f"El año {anio_a_verificar} No es Bisiesto.")
```

```python
def es_bisiesto(anio: int) -> bool: # le pido que me devuelva un booleano (V o F)
    """
    Esta funcion determina si un año especifico es bisiesto

    Es bisiesto cuando es divisible por 4, excepto si es divisible por 100
    a menos que también sea divisible por 400

    Args:
        anio (int): año a verificar

    Returns:
        bool: True si el año es bisiesto, false si no lo es.
    """
    return (anio % 4 == 0 and anio % 100 != 0) or (anio % 400 == 0)
```

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

```python
print("\n--Calculo de Producto Cartesiano--")

anio_actual = datetime.datetime.now().year
edad_1 = anio_actual - anio_nacimiento_1
edad_2 = anio_actual - anio_nacimiento_2

print(f"Edad calculada para el 1er integrante: {edad_1} años.")
print(f"Edad calculada para el 2do integrante: {edad_2} años.")

# convierto el año y edad en texto, y se pasa a una lista en digitos individuales
digitos_anio_1 = list(str(anio_nacimiento_1))
digitos_edad_1 = list(str(edad_1))

digitos_anio_2 = list(str(anio_nacimiento_2))
digitos_edad_2 = list(str(edad_2))

# Calculamos y mostramos los productos cartesianos.
producto_1 = calcular_producto_cartesiano(digitos_anio_1, digitos_edad_1)
print(f"\nEl producto cartesiano de los digitos de {anio_nacimiento_1} y {edad_1} es:")
print(producto_1)

producto_2 = calcular_producto_cartesiano(digitos_anio_2, digitos_edad_2)
print(f"\nEl producto cartesiano de los digitos de {anio_nacimiento_2} y {edad_2} es:")
print(producto_2)
```

```python
def calcular_producto_cartesiano(conjunto_a: list, conjunto_b: list) -> list:
    """
    Calcula el producto cartesiano entre dos conjuntos a traves de listas

    Args:
        conjunto_a - lista 1
        conjunto_b - lista 2

    Returns:
        list: devuelve una lista de tuplas que representa el producto cartesiano
    """
    producto = []

    for elemento_a in conjunto_a:
        for elemento_b in conjunto_b:
            producto.append((elemento_a, elemento_b))
    return producto
```

Parte 3 – Video de Presentación

Link del video explicativo:

Video Explicativo Trabajo Integrado II - Matemática

https://www.youtube.com/watch?v=ihR_yQKPxNk

Repositorio Github con nuestro Proyecto Trabajo Integrador II - Matemática

https://github.com/enkai12/UTN-INTEGRADORES/tree/main/Matematica