La parte de la consigna que trabaje me pedía :

- Ingreso de los DNIs (reales o ficticios).
- Generación automática de los conjuntos de dígitos únicos.
- Cálculo y visualización de:diferencias y diferencia simétrica.
- Suma total de los dígitos de cada DNI.

Siguiendo esta consigna elabore código y funciones para cumplimentar la consigna:
La primera función es **ingresarDni()** que solicita al usuario los números de DNI separados por comas y los convierte en una lista para almacenarlos en la lista dnis.

```
def ingresarDni():
    dnis = input("Ingrese los DNI, separados por ',':")
    dnis = dnis.split(',')
    return dnis
```

Después con **generarConjuntosUnicos(dnis)** se toma cada DNI de la lista dnis y se los convierte en un conjunto de dígitos únicos, eliminando duplicados.

```
def generarConjuntosUnicos(dnis):
    conjuntos = []
    for dni in dnis:
         conjunto_unico = set(dni) # set lo que hace es crear un conjunto de dígitos únicos
y desordenados -> {3,2,4,1}
        conjuntos.append(conjunto_unico)
    return conjuntos
```

Luego se comienza con las funciones relacionadas a matemática:
diferencia_listas(lista1, lista2) calcula la diferencia entre dos listas, retornando los elementos que están en la primera lista pero no en la segunda mediante el bucle for que itera sobre cada elemento en la lista y si no lo encuentra en la lista 2 ni en el resultado mediante el método .append lo agrega al final de la lista llamada resultado

```
def diferencia_listas(lista1, lista2):
    resultado = []
    for elemento in lista1:
        if elemento not in lista2 and elemento not in resultado:
            resultado.append(elemento)
    return resultado
```

En **diferencia_simetrica_listas(lista1, lista2)** calcula la diferencia simétrica entre dos listas, retornando los elementos que están en una lista o en la otra, pero no en ambas. Tiene la misma funcionalidad que diferencia_listas mediante bucles for solamente que esta vez compara el elemento con ambas listas y con el resultado.

```
def diferencia_simetrica_listas(lista1, lista2):
    resultado = []
    for elemento in lista1:
        if elemento not in lista2 and elemento not in resultado:
            resultado.append(elemento)
    for elemento in lista2:
        if elemento not in lista1 and elemento not in resultado:
            resultado.append(elemento)
    return resultado
```

diferencia(conjuntos) aplica la función de diferencia de manera secuencial entre todos los conjuntos generados; cuando se

pasan más de dos dni y se tienen más de dos conjuntos esta función compara el primero con el segundo y a ese resultado lo compara con el tercero, por esta razón usualmente devuelve una lista vacía, debido a que si se pasan más de 2 dni usualmente los números terminan repitiendose

```
def diferencia(conjuntos):
    diferencia = conjuntos[0]
    for conj in conjuntos[1:]:
        diferencia = diferencia_listas(diferencia, conj)
    return diferencia
```

diferenciaSimetrica(conjuntos) aplica la función de diferencia simétrica de manera secuencial entre todos los conjuntos generados, cuando se pasan más de dos conjuntos se realiza el primero con el segundo y a ese resultado con el tercero, por eso a veces también se presta a confusión ya que para realizar la comparación siempre se toman dos conjuntos parámetros.

```
def diferenciaSimetrica(conjuntos):
    diferencia_simetrica = conjuntos[0]
    for conj in conjuntos[1:]:
        diferencia_simetrica = diferencia_simetrica_listas(diferencia_simetrica, conj)
    return diferencia_simetrica
```

Antes de continuar se deben agregar dos opciones que fueron contempladas y descartadas pensando en mostrar la mayor cantidad de variables disponibles.

1. OPCIÓN CON FUNCIONES INTEGRADAS DE PYTHON: Esta opción fue investigada e implementada en la parte de mi compañera cuando realiza las opciones de unión e intersección por ese motivo yo solo procedo a listarla. Se comprende que esta forma es más concisa y eficiente, sin embargo la implementación larga resulta más simple de leer y entender las operaciones de diferencia y diferencia simétrica aun sabiendo que se adicionan muchas más líneas de código.

```
def diferenciaSimetrica(conjuntos):
    return set.symmetric_difference(*conjuntos)

def diferencia(conjuntos):
    return set.difference(*conjuntos)
```

2. OPCIÓN REFACTORIZADA: Esta vez se observó una repetición en el código, en los bucles FOR utilizados en las funciones de diferencia y diferencia simétrica. Se logró refactorizar pero de nuevo se observó un detrimento en la lectura del código, por lo que se decidió continuar con la primera forma, entendiendo que la refactorización contempla una mejor práctica.

```
def agregar_diferencia(lista1, lista2, resultado):
    for elemento in lista1:
        if elemento not in lista2 and elemento not in resultado:
            resultado.append(elemento)

def diferencia_listas(lista1, lista2):
    resultado = []
    agregar_diferencia(lista1, lista2, resultado)
    return resultado

def diferencia_simetrica_listas(lista1, lista2):
    resultado = []
    agregar_diferencia(lista1, lista2, resultado)
    agregar_diferencia(lista2, lista1, resultado)
    return resultado
```

Continuando con el trabajo la parte de matemática se solicita la suma de todos los dígitos de los dni. Con la función **sumaTotal(dnis)** se calcula la suma total de todos los dígitos numéricos presentes en los DNIs ingresados. Se ingresa al bucle for y por cada dni se ejecuta la variable suma_dni en donde se itera sobre los números del dni y si efectivamente son digitos numericos, retorna true , se convierte en un entero y se suma. Agregando a la variable suma_total. Al final se retorna el valor de la misma

```
def sumaTotal(dnis):
    suma_total = 0
    for dni in dnis:
        suma_dni = sum(int(d) for d in dni if d.isdigit())
        suma_total += suma_dni
    return suma_total
```

Por último se ejecuta esta parte del código, donde pasan en una función todos los print que necesitamos, coordinando la ejecución de las funciones anteriores y mostrando los resultados al usuario.

```
def ejecutarOperacionesAdicionales():
    print('___Operaciones Adicionales con DNIs___')
    dnis = ingresarDni()
    conjuntos= generarConjuntosUnicos(dnis)

    print('1. Conjuntos unicos:', conjuntos)
    print('2. Diferencia entre todos los conjuntos:', diferencia(conjuntos))
    print('3. Diferencia simetrica entre conjuntos:', diferenciaSimetrica(conjuntos))
    print('4. Suma total', sumaTotal(dnis))

ejecutarOperacionesAdicionales()
```

En este ejercicio, se abordaron operaciones fundamentales de la teoría de conjuntos, como la diferencia y la diferencia simétrica, y se implementaron mediante estructuras básicas como los bucles, las condicionales y las funciones.

Estas operaciones matemáticas se tradujeron al código utilizando bucles y condicionales. Los bucles for permitieron iterar sobre los elementos de las listas, mientras que las estructuras if se utilizaron para evaluar condiciones específicas, como la pertenencia de un elemento a otra lista. Además, el uso de funciones facilitó la organización del código en bloques reutilizables, mejorando su legibilidad y mantenimiento.

Esta integración de conceptos matemáticos y estructuras de programación me permitió una comprensión más profunda de ambas disciplinas, demostrando cómo los principios abstractos de la teoría de conjuntos pueden aplicarse de manera práctica en el desarrollo de algoritmos y soluciones computacionales.