Tecnicatura Universitaria en Programación

Trabajo Integrador 2: Matemática y Programación

Alumnos

Copertino Nicolás Marcelo, Dni:39728814

Denegri Luis Enrique, Dni:42587203

Jiménez Daniel Alberto, Dni:95899862

Denunciato Ignacio, Dni:46952118

Comisión 12

Profesores:

Monica Leguiza

Mosqueda Luis

Teoría de conjuntos

La teoría de conjuntos es la rama de las matemáticas que estudia las colecciones de objetos llamadas conjuntos, analizando sus propiedades y relaciones. Es fundamental en matemáticas, lógica y computación, formando la base para muchas otras áreas matemáticas y con aplicaciones en áreas como la inteligencia artificial, las bases de datos y la programación.

Definición de conjuntos:

Un conjunto es una colección bien definida de objetos, que pueden tener cualquier naturaleza (números, personas, ideas, etc.).

Operaciones de conjuntos

Para ejemplificar las operaciones entre conjuntos vamos a usar como ejemplo de conjuntos formados DNIs de los integrantes del grupo.

DNI Luis: 42587203 → Conjunto L = {2,3,4,5,7,8}

DNI Nicolás: 39728814 → Conjunto N = {1,2,3,4,7,8,9}

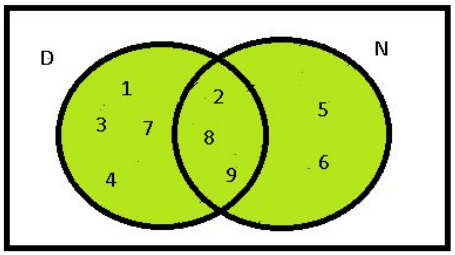
DNI Daniel: 9589962 → Conjunto D = {2,5,6,8,9}

DNI Ignacio: 46952118 → Conjunto I=={1,2,4,5,6,8,9}

Unión de Conjuntos:

La unión de dos conjuntos A y B es el conjunto que contiene todos los elementos que pertenecen a A, o a B, o a ambos (sin repetición). Se representa simbólicamente como A∪B.

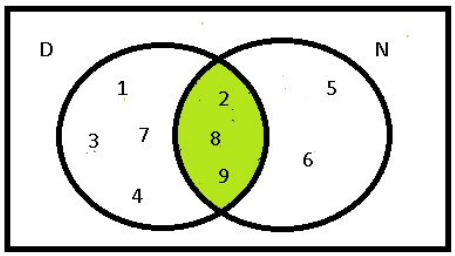
D∪N = {1,2,3,4,5,6,7,8,9}



Intersección de conjuntos:

La intersección de dos conjuntos A y B es el conjunto formado por elementos que son comunes a ambos. Se representa simbólicamente con A∩B.

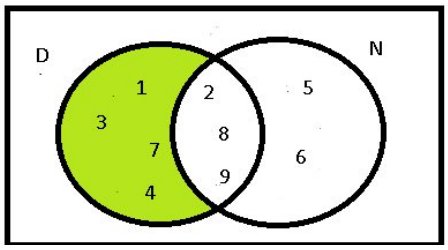
D∩N = {2,8,9}



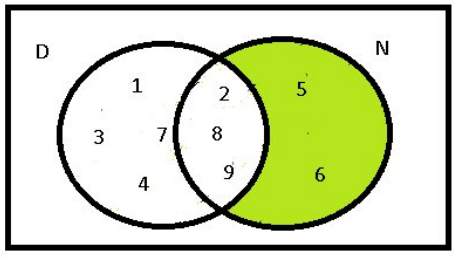
Diferencia de conjuntos

La diferencia de dos conjuntos A y B es un nuevo conjunto que contiene todos los elementos de A que no están en B. Se representa con A-B.

D-N = {1,3,4,7}



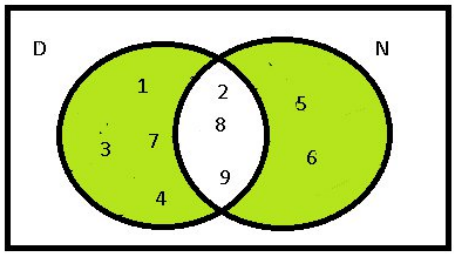
N-D = {5,6}



Diferencia simétrica

La diferencia simétrica de dos conjuntos A y B es un nuevo conjunto que contiene todos los elementos que pertenecen a uno de los dos conjuntos iniciales, pero no a ambos. Se representan como AΔB.

DΔN = {0,2,4,5,7,9}



Parte 1 – Desarrollo Matemático (Conjuntos y Lógica)

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 cual sería el resultado con los conjuntos que tienen.

El algoritmo pide al usuario ingresar por teclado 3 números de DNIs, distingue que se ingresen números de 8 dígitos, si no se cumple con esta condición el algoritmo muestra un mensaje de error y le pide al usuario que ingrese el DNI nuevamente. Con los DNIs forma listas de dígitos únicos y resuelve las siguientes condiciones lógicas:

a) Si hay una lista que tiene una mayor cantidad de dígitos únicos, muestra cuál es ese DNI y que cantidad de dígitos únicos tiene. Si no, si se produce un empate entre la cantidad de dígitos únicos muestra dicho empate y especifica cuáles son los DNIs empatados.

b) El algoritmo recorre las listas con dígitos únicos, si encuentra al digito 0 devuelve por pantalla cual es el DNI que lo contiene. Si no, devuelve un mensaje que dice que ningún DNI contiene al digito 0.

Parte1Desarrollo Matemático.py

C: > Users > ST-P > Desktop > Parte1Desarrollo Matemático.py > ...

```python
#Pedir los DNIs al usuario
def pedir_dni(numero):
    dni = input(f"Ingrese el DNI #{numero} (8 dígitos): ")
    while len(dni) != 8 or not dni.isdigit():
        print("Entrada inválida. El DNI debe tener exactamente 8 números.")
        dni = input(f"Ingrese el DNI #{numero} nuevamente: ")
    return dni
#Se evalua cada digito de la lista y se quitan los numeros repetidos
def quitar_repetidos(lista):
    nueva_lista = []
    for digito in lista:
        if digito not in nueva_lista:
            nueva_lista.append(digito)
    return nueva_lista
#Lista con mayor cantidad de digitos
def mostrar_resultados(largos):
    mayor = max(largos)
    ganadores = [i+1 for i, largo in enumerate(largos) if largo == mayor]

    if len(ganadores) == 1:
        print(f"El DNI #{ganadores[0]} tiene más dígitos únicos: {mayor}")
    else:
        print(f"Hay un empate entre los DNIs con más dígitos únicos: {mayor}")
        print("DNIs empatados:", ", ".join(f"#{n}" for n in ganadores))
#Identifica los ceros en las listas
def verificar_ceros(listas):
    encontrados = []
    for i, lista in enumerate(listas, start=1):
        if '0' in lista:
            encontrados.append(i)
    if encontrados:
        for n in encontrados:
            print(f"El DNI #{n} contiene el dígito 0.")
    else:
        print("Ningún DNI contiene el dígito 0.")

def main():
    dnis = [pedir_dni(i) for i in range(1, 4)]
    listas = [list(dni) for dni in dnis]
    listas_sin_repe = [quitar_repetidos(lista) for lista in listas]

    print("\nDNI sin dígitos repetidos:")
    for i, sin_repe in enumerate(listas_sin_repe, start=1):
        print(f"DNI #{i}:", sin_repe)

    largos = [len(sin_repe) for sin_repe in listas_sin_repe]
    print("\nResultado:")
    mostrar_resultados(largos)

    print("\nVerificación de ceros:")
    verificar_ceros(listas)

if __name__ == "__main__":
    main()
```

Parte 2 – Desarrollo del Programa en Python

A. Operaciones con DNIs

El algoritmo le pide al usuario que ingrese por pantalla la cantidad de DNIs con los que desea operar. Y distingue que se ingrese un numero de 8 dígitos, si no se cumple con esta condición el algoritmo muestra un mensaje de error y le pide al usuario que ingrese el DNI nuevamente.

Con los DNIs forma conjuntos de dígitos únicos ordenados de menor a mayor. Con estos conjuntos realiza las siguientes operaciones entre conjuntos: Unión, Intersección y Diferencia simétrica.

Evalúa si en los conjuntos hay dígitos compartidos, si un conjunto tiene más de 6 dígitos muestra el mensaje “Diversidad numérica alta”.

Además, especifica la cantidad de veces que se repiten dígitos por DNI, finalmente suma todos los dígitos de cada DNI y muestra el resultado de dicha suma.

B. Operaciones con años de nacimiento

El algoritmo pide que se ingrese por pantalla la cantidad de integrantes del grupo, luego sus años de nacimiento y distingue que se ingrese un numero de cuatro dígitos, si esto no sucede muestra un mensaje de error.

Si todos los ingresantes nacieron después del año 2000, mostrara el mensaje “Grupo Z”. También identifica si los años de nacimiento son pares o impares y si alguno de estos años es bisiesto muestra el mensaje “Tenemos un año especial”.

Finalmente calcula el producto cartesiano entre los años de nacimiento y las edades.

ejercicio2.py

C: > Users > ST-P > Desktop > ejercicio2.py > ingresar_dnis

```python
def ingresar_dnis():
    dnis = []
    cantidad = int(input("¿Cuántos DNIs quiere ingresar? "))
    for i in range(cantidad):
        while True:
            dni = input(f"Ingrese el DNI número {i + 1} (8 dígitos): ")
            if dni.isdigit() and len(dni) == 8:
                dnis.append(dni)
                break
            else:
                print("Entrada inválida. El DNI debe tener exactamente 8 dígitos numéricos.")
    return dnis

def obtener_conjuntos_digitos(dnis):
    conjuntos = []
    for dni in dnis:
        conjuntos.append(set(dni))
    return conjuntos

def mostrar_operaciones_conjuntos(conjuntos):
    union_total = set.union(*conjuntos)
    interseccion_total = set.intersection(*conjuntos)
    diferencia_simetrica = set.symmetric_difference(*conjuntos)

    print("\nOperaciones entre todos los conjuntos:")
    print("Unión de conjuntos:", sorted(union_total))
    print("Intersección de conjuntos:", sorted(interseccion_total))
    print("Diferencia simétrica de conjuntos:", sorted(diferencia_simetrica))

    # Evaluación de condiciones lógicas:
    if interseccion_total:
        for digito in sorted(interseccion_total, key=int):
            print(f"Dígito compartido: {digito}")

    for idx, conjunto in enumerate(conjuntos):
        if len(conjunto) > 6:
            print(f"Diversidad numérica alta en DNI {idx + 1}")

def contar_frecuencias(dnis):
    print("\nFrecuencia de dígitos por DNI:")
    for idx, dni in enumerate(dnis):
        print(f"\nDNI {idx + 1}: {dni}")
        for digito in sorted(set(dni), key=int):
            print(f"El dígito {digito} aparece {dni.count(digito)} veces")

def sumar_digitos(dnis):
    print("\nSuma total de dígitos por DNI:")
    for idx, dni in enumerate(dnis):
        suma = sum(int(d) for d in dni)
        print(f"DNI {idx + 1}: {dni} -> Suma: {suma}")

def main():
    print("Operaciones de conjuntos")
    dnis = ingresar_dnis()
    conjuntos = obtener_conjuntos_digitos(dnis)

    print("\nConjuntos de dígitos únicos por DNI :")
    for i, conjunto in enumerate(conjuntos):
        print(f"DNI {i + 1}: {sorted(conjunto, key=int)}")
```

ejercicio2b.py ejercicio2.py

C: > Users > ST-P > Desktop > ejercicio2b.py > main

```python
from datetime import datetime
from itertools import product

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

# Ingreso de años de nacimiento
def ingresar_anios():
    anios = []
    cantidad = int(input("¿Cuántos integrantes hay en el grupo? "))
    for i in range(cantidad):
        while True:
            anio = input(f"Ingrese el año de nacimiento del integrante {i + 1}: ")
            if anio.isdigit() and len(anio) == 4:
                anios.append(int(anio))
                break
            else:
                print("Año inválido. Debe tener 4 dígitos numéricos.")
    return anios

# Contar años pares e impares
def contar_pares_impares(anios):
    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}")

# Evaluar condiciones del grupo
def evaluar_condiciones(anios):
    if all(anio > 2000 for anio in anios):
        print("Grupo Z")

    if any(es_bisiesto(anio) for anio in anios):
        print("Tenemos un año especial")

# Calcular edades actuales
def calcular_edades(anios):
    anio_actual = datetime.now().year
    return [anio_actual - anio for anio in anios]

# Producto cartesiano
def producto_cartesiano(anios, edades):
    conjunto_anios = set(anios)
    conjunto_edades = set(edades)
    resultado = list(product(conjunto_anios, conjunto_edades))

    print("\nProducto cartesiano entre años y edades:")
    for par in resultado:
        print(par)

# Función principal
def main():
    print("=== ANÁLISIS DE AÑOS DE NACIMIENTO ===")
    anios = ingresar_anios()
```