Link github: <https://github.com/MauricioCifuentes2000/arq_proy.git>

Codigo .py:

#binario a hexa (enteros y fracciones)

#complemento restringido de binario

#decimal a hexadecimal (fracciones y periodicidad)

import sys

def binary\_to\_hexadecimal(binary\_str):

    """

    Convierte un número binario (con parte entera y fraccionaria) a hexadecimal.

    """

    if '.' in binary\_str:

        integer\_part, fractional\_part = binary\_str.split('.')

    else:

        integer\_part, fractional\_part = binary\_str, ''

    # Conversión de la parte entera

    hex\_integer = hex(int(integer\_part, 2))[2:].upper() if integer\_part else '0'

    # Conversión de la parte fraccionaria

    hex\_fractional = ''

    if fractional\_part:

        # Convertir la fracción binaria a decimal

        fractional\_decimal = 0

        for i, digit in enumerate(fractional\_part, 1):

            fractional\_decimal += int(digit) \* (2 \*\* -i)

        # Convertir la fracción decimal a hexadecimal

        fractional\_decimal \*= 16

        for \_ in range(10):  # Limitar a 10 dígitos hexadecimales para evitar infinitas repeticiones

            digit = int(fractional\_decimal)

            hex\_fractional += hex(digit)[2:].upper()

            fractional\_decimal = (fractional\_decimal - digit) \* 16

            if fractional\_decimal == 0:

                break

    if hex\_fractional:

        return f"{hex\_integer}.{hex\_fractional}"

    else:

        return hex\_integer

def restricted\_complement(binary\_str):

    """

    Calcula el complemento restringido de un número binario.

    Suposición: El primer bit es el bit de signo y no se invierte.

    """

    if not all(c in '01' for c in binary\_str):

        return "Error: La entrada no es un número binario válido."

    if len(binary\_str) == 0:

        return "Error: Entrada vacía."

    # El primer bit (bit de signo) permanece igual

    sign\_bit = binary\_str[0]

    # Invertir los demás bits

    complemented\_bits = ''.join('1' if bit == '0' else '0' for bit in binary\_str[1:])

    return sign\_bit + complemented\_bits

def decimal\_to\_hexadecimal(decimal\_str):

    """

    Convierte un número decimal (con parte entera y fraccionaria) a hexadecimal,

    mostrando periodicidad si existe.

    """

    if '.' in decimal\_str:

        integer\_part, fractional\_part = decimal\_str.split('.')

    else:

        integer\_part, fractional\_part = decimal\_str, ''

    # Conversión de la parte entera

    hex\_integer = hex(int(integer\_part))[2:].upper() if integer\_part else '0'

    # Conversión de la parte fraccionaria

    if fractional\_part:

        fractional\_decimal = float("0." + fractional\_part)

        seen = {}

        hex\_fractional = ''

        repeating\_start = -1

        for i in range(20):  # Limitar a 20 iteraciones para detectar repeticiones

            fractional\_decimal \*= 16

            digit = int(fractional\_decimal)

            hex\_digit = hex(digit)[2:].upper()

            hex\_fractional += hex\_digit

            fractional\_decimal -= digit

            # Detectar periodicidad

            if fractional\_decimal in seen:

                repeating\_start = seen[fractional\_decimal]

                break

            seen[fractional\_decimal] = i + 1

            if fractional\_decimal == 0:

                break

        if repeating\_start != -1:

            non\_repeating = hex\_fractional[:repeating\_start]

            repeating = hex\_fractional[repeating\_start:]

            hex\_fractional = f"{non\_repeating}({repeating})"

        return f"{hex\_integer}.{hex\_fractional}"

    else:

        return hex\_integer

def display\_menu():

    """

    Muestra el menú de opciones al usuario.

    """

    print("\n===== Menú de Operaciones =====")

    print("1. Conversión de Binario a Hexadecimal")

    print("2. Obtener el Complemento Restringido de un Número Binario")

    print("3. Conversión de Decimal a Hexadecimal")

    print("4. Salir")

    print("================================")

def main():

    while True:

        display\_menu()

        choice = input("Selecciona una opción (1-4): ").strip()

        if choice == '1':

            binary\_input = input("Introduce el número binario (ejemplo: 1010.101): ").strip()

            result = binary\_to\_hexadecimal(binary\_input)

            print(f"Hexadecimal: {result}")

        elif choice == '2':

            binary\_input = input("Introduce el número binario para obtener su complemento restringido: ").strip()

            result = restricted\_complement(binary\_input)

            print(f"Complemento Restringido: {result}")

        elif choice == '3':

            decimal\_input = input("Introduce el número decimal (ejemplo: 10.625): ").strip()

            try:

                # Validar entrada decimal

                float(decimal\_input)

                result = decimal\_to\_hexadecimal(decimal\_input)

                print(f"Hexadecimal: {result}")

            except ValueError:

                print("Error: Entrada no es un número decimal válido.")

        elif choice == '4':

            print("¡Gracias por usar el programa! Hasta luego.")

            sys.exit()

        else:

            print("Opción inválida. Por favor, elige una opción entre 1 y 4.")

            continue

        # Preguntar si el usuario desea realizar otra operación

        while True:

            again = input("¿Deseas realizar otra operación? (s/n): ").strip().lower()

            if again == 's':

                break

            elif again == 'n':

                print("¡Gracias por usar el programa! Hasta luego.")

                sys.exit()

            else:

                print("Entrada inválida. Por favor, responde con 's' o 'n'.")

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

    main()