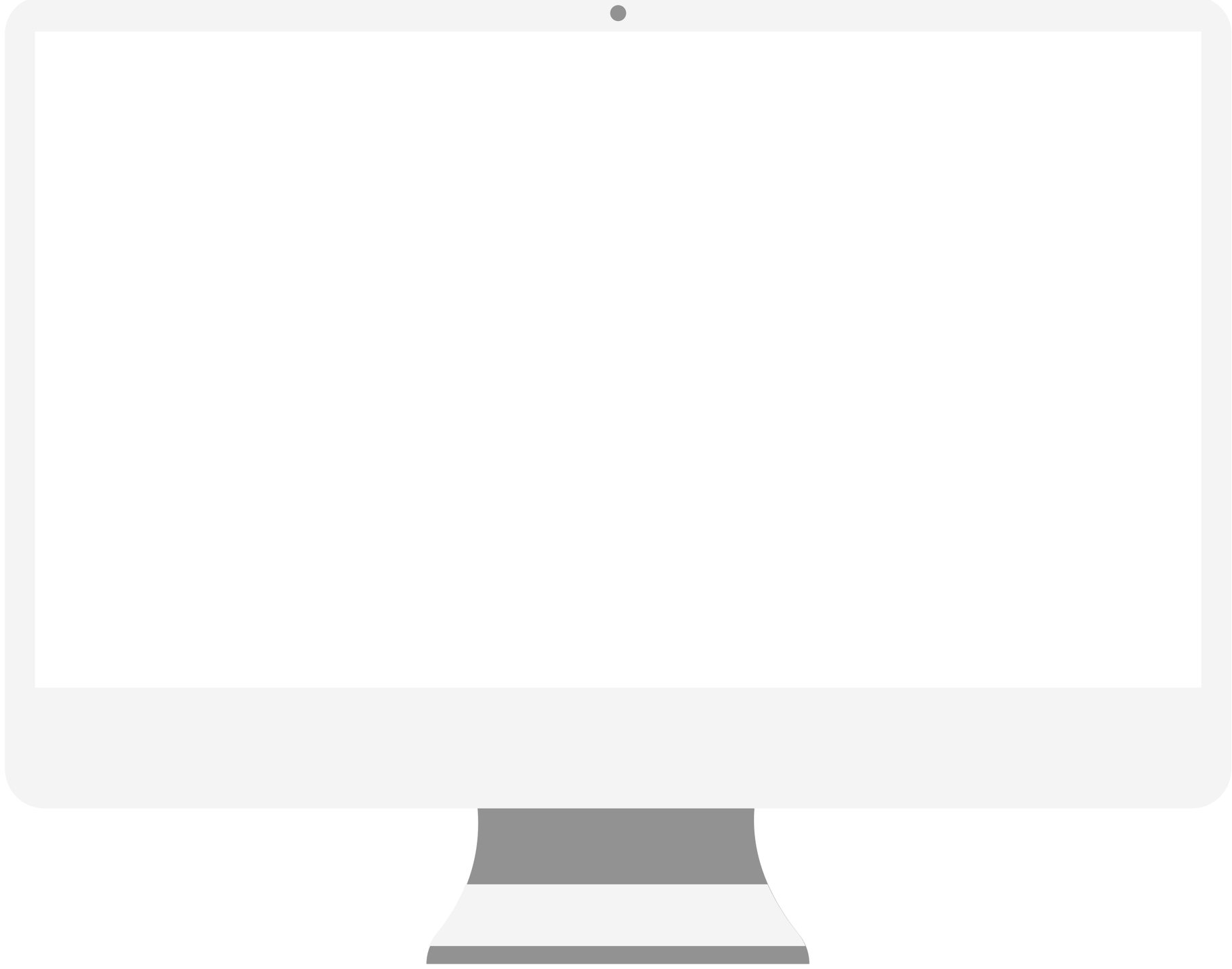
TC1028
Pensamiento computacional para ingeniería
Proyecto Integrador



Darío Cuauhtémoc Peña Mariano
Andres Salcedo Ortiz
Melanie

Programa 1
Funcionamiento:

El programa funciona con ayuda del paquete “random”, en el código creamos 14 funciones, las cuales tiene un objetivo en especial para cumplir con cada requerimiento del reto, las funciones son las siguientes:

crearMatriz: esta primera función recibe a y b, los cuales lo usaremos para definir el tamaño de nuestra matriz, esta función crea la matriz con cada elemento siendo un número aleatorio del 1-10, cuando termine de crear la matriz la retornaremos al final de la función.

impresionMatriz: esta función recibe de argumento una matriz, y las mostraremos ordenada gráficamente por consola mediante prints.

primerFila: Esta función recibe como argumento una matriz, la función calculará la primer fila, mostrará por terminal los numeros de la fila y después el resultado la suma de la fila.

imprimirdiagonal: Esta función recibe como argumento una matriz, la función calculará la diagonal principal, mostrará por terminal los numeros que la componen, después la suma de éstos y por último retorna el valor de la suma

secundariaDiagonal: Esta función recibe como argumento una matriz, la función calculará la diagonal secundaria, mostrará por terminal los numeros que la componen, después la suma de éstos y por último retorna el valor de la suma.

ultimaColumna: Esta función recibe como argumento una matriz, la función calculará la última columna, mostrará por terminal los numeros que la componen, después la suma de éstos y por último retorna el valor de la suma.

columnasImpares: Esta función recibe como argumento una matriz, la función calculará las columnas impares, mostrará por terminal los numeros que las componen, y por últomo la suma de éstas.

filasPares: Esta función recibe como argumento una matriz, la función calculará las filas pares, mostrará por terminal los numeros de cada fila y después la suma de las filas pares.

columnasImpares: Esta función recibe como argumento una matriz, la función calculará las columnas impares, mostrará por terminal los numeros de cada columna impar y después el resultado de la suma de éstas.

todasFilas: Esta función recibe como argumento una matriz, la función calculará todas las filas, mostrará por terminal los numeros de cada fila y después la suma de todas las filas.
filasMenosPyU: Esta función recibe como argumento una matriz, la función calculará la primer y última fila, mostrará por terminal los numeros de éstas y después la suma de las filas, en caso de que la matriz sea mayor de 2x2, en caso contrario no sería una matriz y mostraría un mensaje de error al calcular.
elementosNoRango1_5: Esta función recibe como argumento una matriz, la función calculará los elementos de la martriz que no estén entre el 1 y el 5, los multiplos de 3 y los que no son multiplos de 3, por último mostrará los elementos de cada caso y el resultado de la suma.
columnasExceAnte: Esta función recibe como argumento una matriz, la función calculará la antepenultima columna, mostrará por terminal todas las columnas excepto la antepenultima, también mostrar el reultado de la suma de éstas columnas, en caso que no tenga antepenultima mostraría un mensaje de error al calcular.
sumaTrianguloSuperior: Esta función recibe como argumento una matriz, la función calculará la
la diagonal principal con la función "**imprimirdiagonal**" y la trinagular superior en inferior de la matriz, mostrará la suma de estos tres calculos anteriores, la suma de la diagonal principal con la triangular superior y la suma de la diagonal principal con la triangular inferior.
Por último creamos la funcion **funcionesMatrices**, esta función será la función principal que se ejecute, la cual no recibe argumento, pedimos al usuario un numero que sea n y otro que sea m, con estos números crearemos una matriz con ayuda de **crearMatriz**, la matriz resultante la pasaremos como argumento a nuestras 12 funciones restantes que harán todos nuestros cálculos, mostrarán resultados por pantalla y con esto terminaríamos nuestro código

Pseudocódigo:

```
Paquetes: “random”
Variables:
Inicio

Importar random

Función crearMatriz(a, b):
    matriz = Lista vacía
    Para cada elemento x en el rango a:
        listaLit = Lista vacía
        Para cada elemento y en el rango b:
            dato = Entero aleatorio entre 1 y 10
            Añadir dato a listaLit
        Añadir listaLit a matriz
    Imprimir una línea en blanco
    Devolver matriz

Función impresionMatriz(matriz):
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Imprimir matriz[x][y] seguido de dos espacios
        Imprimir nueva línea

Función primerFila(matriz):
    Imprimir "primera fila:" seguido de los elementos de matriz[0]
    Imprimir "suma de la primera fila:" seguido de la suma de matriz[0]
    Devolver matriz[0]

Función imprimirdiagonal(matriz):
    diagonalPrincipal = Lista vacía
    n = 0
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Si y es igual a n:
                Añadir matriz[x][n] a diagonalPrincipal
        Incrementar n
    Imprimir "diagonal principal:" seguido de los elementos de diagonalPrincipal
    Imprimir "Suma diagonal principal:" seguido de la suma de diagonalPrincipal
    Devolver diagonalPrincipal

Función secundariaDiagonal(matriz):
    diagonalSecundaria = Lista vacía
    b = Longitud de matriz[0]
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Si y es igual a b-1:
                Añadir matriz[x][b-1] a diagonalSecundaria
        Decrementar b
    Imprimir "diagonal secundaria:" seguido de los elementos de diagonalSecundaria
    Imprimir "suma de la diagonal secundaria:" seguido de la suma de diagonalSecundaria
    Devolver diagonalSecundaria

Función ultimaColumna(matriz):
    b = Longitud de matriz[0]
    ultimaColumn = Lista vacía
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Si y es igual a b-1:
                Añadir matriz[x][y] a ultimaColumn
    Imprimir "ultima columna:" seguido de los elementos de ultimaColumn
    Imprimir "la suma de la ultima columna es" seguido de la suma de ultimaColumn
    Devolver ultimaColumn

Función filasPares(matriz):
    Imprimir "Las filas pares son:"
    Imprimir una línea en blanco
    resultado = 0
    Para cada elemento x en el rango de la longitud de matriz:
        Si (x+1) % 2 == 0:
            Imprimir "filas pares" seguido de los elementos de matriz[x]
            resultado = resultado + suma de matriz[x]
    Imprimir "la suma de las filas pares es:" seguido de resultado

Función columnasImpares(matriz):
    resultado = 0
    Imprimir "columnas impares:"
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Si (y+1) % 2 != 0:
                Imprimir matriz[x][y] seguido de dos espacios
                resultado = resultado + matriz[x][y]
        Imprimir una línea en blanco
    Imprimir "El resultado de la suma de las columnas impares es:" seguido de resultado

Función todasFilas(matriz):
    resultado = 0
    Imprimir "todas las filas:"
    Para cada elemento x en el rango de la longitud de matriz:
        Imprimir "fila numero" x+1 ":" seguido de los elementos de matriz[x]
        resultado = resultado + suma de matriz[x]
    Imprimir "la suma de todas las filas es:" seguido de resultado

Función filasMenosPyU(matriz):
    suma = 0
    Si la longitud de matriz es mayor que 2:
        Imprimir "todas las filas excepto primera y ultima:"
        Para cada elemento x en el rango de la longitud de matriz:
            Si x es igual a 0 o x es igual a la longitud de matriz - 1:
                Imprimir "fila omitida xxxxxxxx"
            Sino:
                Imprimir los elementos de matriz[x]
                suma = suma + suma de matriz[x]
        Imprimir "La suma de estas filas es:" seguido de suma
    Sino:
        Imprimir "Error, la matriz debe ser de al menos 2x2 para omitir la primera y última fila"

Función elementosNoRango1_5(matriz):
    elementosNo1_5 = Lista vacía
    multiplos3 = Lista vacía
    noMultiplos3 = Lista vacía
    Para cada elemento x en el rango de la longitud de matriz:
        Para cada elemento y en el rango de la longitud de matriz[0]:
            Si no(matriz[x][y] está en el rango de 1 a 5):
                Añadir matriz[x][y] a elementosNo1_5
            Si matriz[x][y] % 3 == 0:
                Añadir matriz[x][y] a multiplos3
            Sino:
                Añadir matriz[x][y] a noMultiplos3
    Imprimir "Los elementos que no están en el rango 1-5 de la matriz son:" seguido de elementosNo1_5
    Imprimir "Suma de los elementos que no están en el rango 1-5 de la matriz es:" seguido de la suma de elementosNo1_5
    Imprimir "Los múltiplos de 3 de la matriz son:" seguido de multiplos3
    Imprimir "Suma de los múltiplos de 3 de la matriz es:" seguido de la suma de multiplos3
    Imprimir "Los NO múltiplos de 3 de la matriz son:" seguido de noMultiplos3
    Imprimir "Suma de los NO múltiplos de 3 de la matriz es:" seguido de la suma de noMultiplos3

Función columnasExceAnte(matriz):
    suma = 0
    Si la longitud de matriz[0] es mayor que 3:
        Imprimir "columnas excepto antepenúltima:"
        Para cada elemento x en el rango de la longitud de matriz:
            b = Longitud de matriz
            Para cada elemento y en el r
Fin
```

Funcionamiento del programa:

reto1_prog1.py U, M | reto1_prog2.py U, M | reto1_prog3.py U, M

python > retos > reto1_compu1 > reto1_prog1.py > ...

```python
import random

#Creacion de la matriz
def crearMatriz(a,b): ···

#impresion matriz
def impresionMatriz(matriz): ···
#imprimir primera fila
def primerFila(matriz): ···

#primer diagonal
def imprimirdiagonal(matriz): ···
#SECUNDARI diagonal
def secundariaDiagonal(matriz): ···
#ultima columna
def ultimaColumna(matriz): ···

#filas pares
def filasPares(matriz): ···
#columnas impares
def columnasImpares(matriz): ···
#todas las filas
def todasFilas(matriz): ···
##todas las filas excepto primero y ultima
def filasMenosPyU(matriz): ···
##todos los elementos que no esten en el rango 1-5
def elementosNoRango1_5(matriz): ···
##columnas excepto antepenultima
def columnasExceAnte(matriz): ···
def sumaTrianguloSuperior(matriz): ···

def funcionesMatrices(): ···


funcionesMatrices()
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SQL CONSOLE | COMMENTS

```
PS D:\tec\trabajosProg> & C:/Users/dari-/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/tec/trabajosProg/python/retos/reto1_compu1/reto1_prog1.py
Hola, a continuación in gresa tu n que quieras para tu matriz:5
Hola, a continuación in gresa tu m que quieras para tu matriz:5
------------------------------------------------------------

2  5  2  10  6
7  7  1  7  5
2  10  2  6  4
10  1  1  5  6
4  1  10  5  5
diagonal principal: 2 7 2 5 5
Suma diagonal principal: 21
primera fila: 2 5 2 10 6
suma de la primera fila: 25
ultima columna: 6 5 4 6 5
la suma de la ultima columna es 26
Las filas pares son:

filas pares 7 7 1 7 5
filas pares 10 1 1 5 6
la suma de las filas pares es: 50
columnas impares:
2  2  6
7  1  5
2  2  4
10  1  6
4  10  5
El resultado de la suma de las columnas impares es: 67
diagonal principal: 2 7 2 5 5
Suma diagonal principal: 21
suma triangular superior de la matriz: 52
suma triangular inferior de la matriz: 51
suma diagonal principal de la matriz y triangular superior: 73
suma diagonal principal de la matriz y triangular inferior: 72
todas las filas:
fila numero 1 : 2 5 2 10 6
fila numero 2 : 7 7 1 7 5
fila numero 3 : 2 10 2 6 4
fila numero 4 : 10 1 1 5 6
fila numero 5 : 4 1 10 5 5
la suma de todas las filas es: 124
todas las filas excepto primera y ultima:
fila omitida xxxxxxxx
7 7 1 7 5
2 10 2 6 4
10 1 1 5 6
fila omitida xxxxxxxx
La suma de estas filas es: 74
columnas excepto antepenultima:
2  5  X  10  6
7  7  X  7  5
2  10  X  6  4
10  1  X  5  6
4  1  X  5  5

suma de las columnas excepto la antepenultima: 108
diagonal secundaria: 6 7 2 1 4
suma de la diagonal secundaria: 20
Los elementos que no estan en el rango 1-5 de la matiz son: 10 6 7 7 7 10 6 10 6 10
Suma de los elementos que no estan en el rango 1-5 de la matiz es: 79
Los multiplos de 3 de la matriz son: 6 6 6
suma de los multiplos de 3 de la matriz es: 18
Los NO multiplos de 3 de la matriz son: 2 5 2 10 7 7 1 7 5 2 10 2 4 10 1 1 5 4 1 10 5 5
Suma de los NO multiplos de 3 de la matriz es: 106
------------------------------------------------------------
PS D:\tec\trabajosProg>
```

Programa 2
Funcionamiento:

El programa le pide al usuario que rellene una matriz de 3x3 el cual será un cuadro mágico, con esta matriz el programa calculará la suma de la diagonal principal, la diagonal secundaria, la primer, segunda y tercera fila y columna, estos valores se guardarán en literales al terminar los calculos,
después se hará una comprobación para saber si el cuadro es mágico, esto sucede cuando todas las sumas anteriores dan el mismo número, en caso de ser correcto se mostrará por consola “Felicidades tu cuadro SI es un cuadro mágico”, en caso contrario se mostrará por consola “Lo sentimos, tu cuadro NO es un cuadro mágico” y permitirá al usuario escribir “SI” o “NO”, en caso de ser SI, el código permitirá al usuario intentar de nuevo ingresar su cuadro para saber si es mágico o no, en caso de que la respuesta sea NO, se mostrará por pantalla “Gracias, nos vemos pronto” y se terminará el programa.

Pseudocódigo:

Paquetes: “random”
Variables:
Inicio

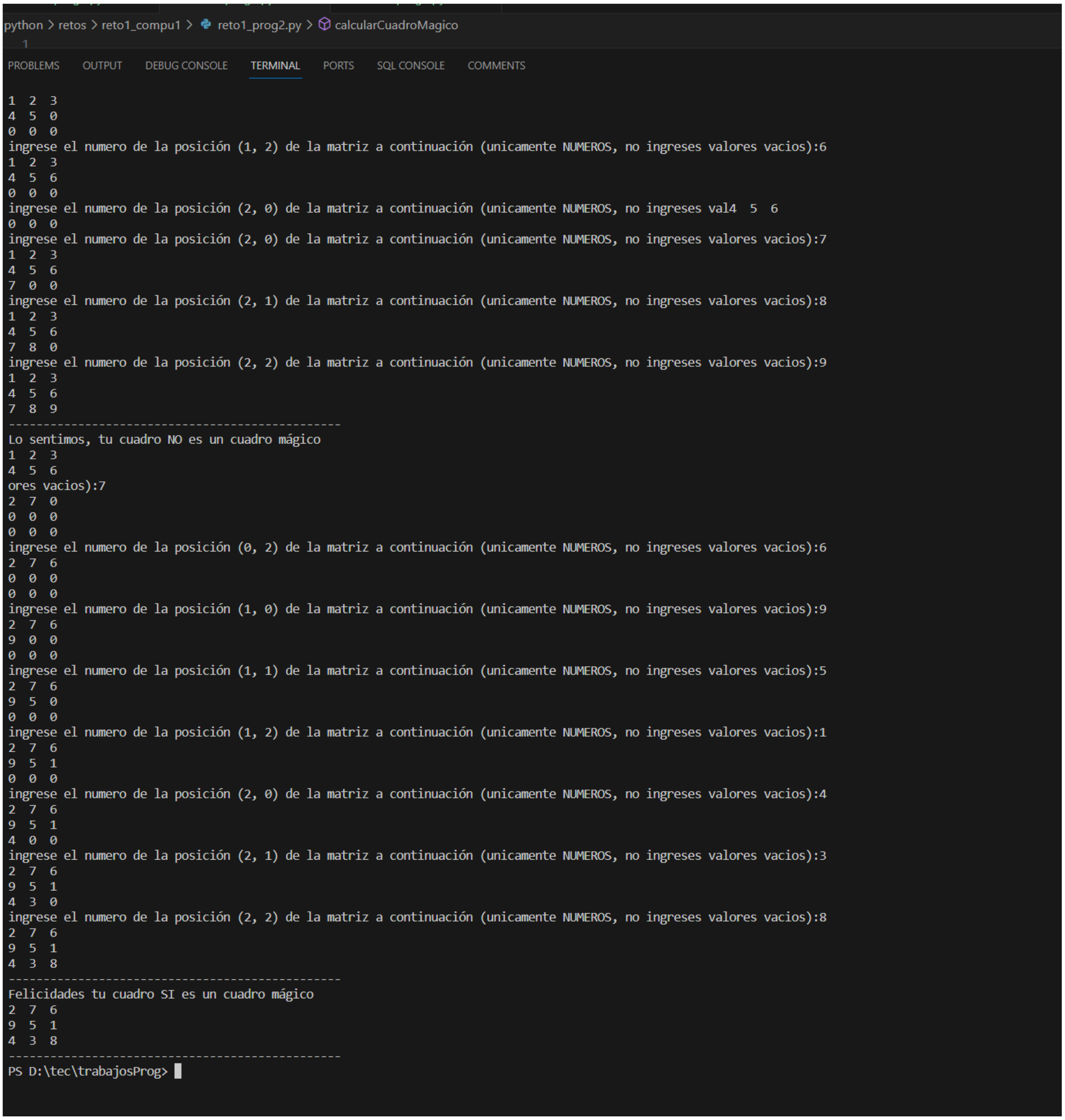
```
# Función para imprimir una matriz
def imprimirMatriz(matriz):
    # Itera sobre las filas de la matriz
    for x in range(len(matriz)):
        # Itera sobre las columnas de la matriz
        for y in range(len(matriz[0])):
            # Imprime el elemento en la posición (x, y)
            imprimir(matriz[x][y], end="  ")
        # Imprime un salto de línea al final de cada fila

# Función para calcular y retornar la suma de la diagonal principal
def calcularSumaDiagonalPrin(matriz):
    # Inicializa una lista para almacenar los elementos de la diagonal principal
    diagonalPrincipal = []
    # Inicializa un contador para rastrear la posición en la diagonal
    n = 0
    # Itera sobre las filas de la matriz
    for x in range(len(matriz)):
        # Itera sobre las columnas de la matriz
        for y in range(len(matriz[0])):
            # Si y es igual a n, el elemento está en la diagonal principal
            if y == n:
                # Agrega el elemento a la lista de la diagonal principal
                diagonalPrincipal.append(matriz[x][n],)
        # Incrementa el contador de posición en la diagonal
        n += 1
    # Retorna la suma de los elementos en la diagonal principal

# Función para calcular y retornar la suma de la diagonal secundaria
def calcularSumaDiagonalSec(matriz):
    # Inicializa una lista para almacenar los elementos de la diagonal secundaria
    diagonalSecundaria = []
    # Obtiene el número de columnas en la matriz
    b = len(matriz[0])
    # Itera sobre las filas de la matriz
    for x in range(len(matriz)):
        # Itera sobre las columnas de la matriz
        for y in range(len(matriz[0])):
            # Si y es igual a b - 1, el elemento está en la diagonal secundaria
            if y == b - 1:
                # Agrega el elemento a la lista de la diagonal secundaria
                diagonalSecundaria.append(matriz[x][b - 1],)
        # Decrementa el contador de posición en la diagonal secundaria
        b -= 1
    # Retorna la suma de los elementos en la diagonal secundaria

# Función para calcular y retornar las sumas de las columnas
def calcularSumasColumnas(matriz):
    # Obtiene el número de columnas en la matriz
    b = len(matriz[0])
    # Inicializa listas para almacenar los elementos de cada columna
    ultimaColumna = []
    segundaColumna = []
    primeraColumna = []
    # Itera sobre las filas de la matriz
    for x in range(len(matriz)):
        # Itera sobre las columnas de la matriz
        for y in range(len(matriz[0])):
            # Si y es igual a b - 1, el elemento está en la última columna
            if y == b - 1:
                ultimaColumna.append(matriz[x][y])
            # Si y es igual a b - 2, el elemento está en la segunda columna
            elif y == b - 2:
                segundaColumna.append(matriz[x][y])
            # Si y es igual a b - 3, el elemento está en la primera columna
            elif y == b - 3:
                primeraColumna.append(matriz[x][y])
    # Retorna una lista con la suma de los elementos de cada columna

# Función para calcular y retornar las sumas de las filas
def calcularSumasFilas(matriz):
    # Inicializa listas para almacenar los elementos de cada fila
    ultimaFila = []
    segundaFila = []
    primeraFila = []
    # Itera sobre las filas de la matriz
    for x in range(len(matriz)):
        # Si x es igual a 0, los elementos están en la primera fila
        if x == 0:
            primeraFila = matriz[x]
        # Si x es igual a 1, los elementos están en la segunda fila
        elif x == 1:
            segundaFila = matriz[x]
        # Si x es igual a 2, los elementos están en la última fila
        elif x == 2:
            ultimaFila = matriz[x]
    # Retorna una lista con la suma de los elementos de cada fila

# Función para calcular si la matriz es un cuadro mágico y mostrar el resultado
def calcularCuadroMagico():
    # Pedir matriz al usuario
    matriz = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
    for x in range(len(matriz)):
        for y in range(len(matriz[0])):
            mensaje = x, y
            # Pedir al usuario el número en la posición (x, y) de la matriz
            numeroUsuario = int(input("Ingrese el número de la posición " + str(mensaje) + " de la matriz a continuación (solo números, no valores vacíos): "))
            # Asignar el número a la posición correspondiente en la matriz
            matriz[x][y] = numeroUsuario
            # Imprimir la matriz actualizada
            imprimirMatriz(matriz)

    # Calcular sumas y verificar si es un cuadro mágico
    sumaDiagonalPrin = calcularSumaDiagonalPrin(matriz)
    sumaDiagonalSec = calcularSumaDiagonalSec(matriz)
    sumaColumnas = calcularSumasColumnas(matriz)
    sumaFilas = calcularSumasFilas(matriz)

    # Verificar si todas las sumas son iguales
    if (sumaDiagonalPrin == sumaDiagonalSec == sumaColumnas[0] == sumaColumnas[1] == sumaColumnas[2] == sumaFilas[0] == sumaFilas[1] == sumaFilas[2]):
        # Si todas las sumas son iguales, imprimir un mensaje de felicitaciones
        print("----------------------------------------------------")
        print("¡Felicidades! Tu cuadro SÍ es un cuadro mágico")
        imprimirMatriz(matriz)
        print("----------------------------------------------------")
    else:
        # Si no son iguales, imprimir un mensaje de que no es un cuadro mágico
        print("----------------------------------------------------")
        print("Lo sentimos, tu cuadro NO es un cuadro mágico")
        imprimirMatriz(matriz)
        respuesta = input("¿Quieres intentarlo de nuevo? (escribe SI/NO en mayúsculas únicamente) ")
        print("----------------------------------------------------")
        if respuesta == "SI":
            # Si el usuario quiere intentarlo de nuevo, llamar a la función nuevamente
            calcularCuadroMagico()
        else:
            # Si no, imprimir un mensaje de despedida
            print("Gracias, nos
regresar 0
```

Fin

Funcionamiento del programa:

```python
def diagonalPrin(matriz):...

def secundariaDiagonal(matriz):...

def sumaColumnas(matriz):...

def sumaFilas(matriz):...


def calcularCuadroMagico():
    #Pedir matriz a usuiario
    matriz=[
    [0,0,0],
    [0,0,0],
    [0,0,0]
    ]
    for x in range(len(matriz)):

        for y in range(len(matriz[0])):
          mensaje = x,y
          numeroUsuario= int(input("ingrese el numero de la posición "+ str(mensaje)+" de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):"))
          matriz[x][y] = numeroUsuario

          impresionMatriz(matriz)
    #Pedir matriz a usuiario
    sumadiagonalP= diagonalPrin(matriz)
    sumadiagonalS= secundariaDiagonal(matriz)
    sumaPcolumna= sumaColumnas(matriz)[0]
    sumaScolumna= sumaColumnas(matriz)[1]
    sumaTcolumna= sumaColumnas(matriz)[2]
    sumaPfila= sumaFilas(matriz)[0]
    sumaSfila= sumaFilas(matriz)[1]
    sumaTfila= sumaFilas(matriz)[2]

    if sumadiagonalP==sumadiagonalS==sumaPcolumna==sumaScolumna==sumaTcolumna==sumaPfila==sumaSfila==sumaTfila:
        print("-----------------------------------------------")
        print("Felicidades tu cuadro SI es un cuadro mágico")
        impresionMatriz(matriz)
        print("-----------------------------------------------")
    else:
        print("-----------------------------------------------")
        print("Lo sentimos, tu cuadro NO es un cuadro mágico")
        impresionMatriz(matriz)
        respuesta= input("Quieres intentarlo de nuevo? (escribe SI/NO en mayusculas unicamente) ")
        print("-----------------------------------------------")
        if respuesta=="SI":
            calcularCuadroMagico()
        else:
            print("Gracias, nos vemos pronto")
            return 0


calcularCuadroMagico()
```

```
1  2  0
0  0  0
0  0  0
ingrese el numero de la posición (0, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):3
1  2  3
0  0  0
0  0  0
ingrese el numero de la posición (1, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):4
1  2  3
4  0  0
0  0  0
ingrese el numero de la posición (1, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):5
1  2  3
4  5  0
0  0  0
ingrese el numero de la posición (1, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):6
1  2  3
4  5  6
0  0  0
ingrese el numero de la posición (2, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses val4  5  6
0  0  0
ingrese el numero de la posición (2, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):7
1  2  3
4  5  6
7  0  0
ingrese el numero de la posición (2, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):8
1  2  3
4  5  6
7  8  0
ingrese el numero de la posición (2, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):9
1  2  3
4  5  6
7  8  9
-----------------------------------------------
Lo sentimos, tu cuadro NO es un cuadro mágico
1  2  3
4  5  6
ores vacios):7
2  7  0
0  0  0
0  0  0
ingrese el numero de la posición (0, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):6
2  7  6
0  0  0
0  0  0
ingrese el numero de la posición (1, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):9
2  7  6
9  0  0
0  0  0
ingrese el numero de la posición (1, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):5
```


python > retos > reto1_compu1 > reto1_prog2.py > calcularCuadroMagico
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS
1 2 3
4 5 0
0 0 0
ingrese el numero de la posición (1, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):6
1 2 3
4 5 6
0 0 0
ingrese el numero de la posición (2, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses val4 5 6
0 0 0
ingrese el numero de la posición (2, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):7
1 2 3
4 5 6
7 0 0
ingrese el numero de la posición (2, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):8
1 2 3
4 5 6
7 8 0
ingrese el numero de la posición (2, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):9
1 2 3
4 5 6
7 8 9
Lo sentimos, tu cuadro NO es un cuadro mágico
1 2 3
4 5 6
ores vacios):7
2 7 0
0 0 0
0 0 0
ingrese el numero de la posición (0, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):6
2 7 6
0 0 0
0 0 0
ingrese el numero de la posición (1, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):9
2 7 6
9 0 0
0 0 0
ingrese el numero de la posición (1, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):5
2 7 6
9 5 0
0 0 0
ingrese el numero de la posición (1, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):1
2 7 6
9 5 1
0 0 0
ingrese el numero de la posición (2, 0) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):4
2 7 6
9 5 1
4 0 0
ingrese el numero de la posición (2, 1) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):3
2 7 6
9 5 1
4 3 0
ingrese el numero de la posición (2, 2) de la matriz a continuación (unicamente NUMEROS, no ingreses valores vacios):8
2 7 6
9 5 1
4 3 8
Felicidades tu cuadro SI es un cuadro mágico
2 7 6
9 5 1
4 3 8
PS D:\tec\trabajosProg>

Programa 3
Funcionamiento:

-Este programa calcula la serie de fibonacci en la posición n, para esto hicimos una función que recibe un argumento , en caso de que el argumento sea 0 regresa una lista vacía, en caso de que sea 1 regresa [0] en caso de que sea 2 regresa[0,1], si no es ninguno de estos casos sea guarda en una lista el resultado de volver a llamar a la función y a sea lista se le agregan el resultado de sumar los elementos de esa misma lista en la ultima y penultima posición, por ultimo la función regresa esta lista. Al comenzar el programa pedimos al usuario un número, este será el numero de posición a la que se llegará la serie de fibonacci, por ultimo imprimiremos el resultado de la función que creamos con el argumento igual al número que ingresó el usuario.

Pseudocódigo:

Inicio
Función serieFibonacci2(cantidad):
    Si cantidad es igual a 0:
        Devolver una lista vacía []
    Sino, si cantidad es igual a 1:
        Devolver una lista con un elemento [0]
    Sino, si cantidad es igual a 2:
        Devolver una lista con dos elementos [0, 1]
    Sino:
        numeros = llamar a serieFibonacci2 con argumento (cantidad - 1)
        Añadir (numeros[-1] + numeros[-2]) a la lista numeros
        Devolver numeros

n = Leer un número entero desde la entrada de usuario
Imprimir "Este es el resultado:"
Imprimir cada elemento de la lista devuelta por serieFibonacci2(n) separados por espacio
regresar 0
Fin

Funcionamiento del programa:

reto1_prog1.py U, M | reto1_prog2.py U, M | reto1_prog3.py U, M

python > retos > reto1_compu1 > reto1_prog3.py > ...

```python

def serieFibonacci2(cantidad):

    if cantidad == 0:
        return []
    elif cantidad == 1:
        return [0]
    elif cantidad ==2:
        return [0,1]
    else:
        numeros= serieFibonacci2(cantidad-1)
        numeros.append(numeros[-1]+numeros[-2])
        return numeros

n = int(input("Ingresa cuantos elementos de la serie de fibonacci quieres que calcule el programa:"))
print("Este es el resultado:")
print(*serieFibonacci2(n))
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SQL CONSOLE | COMMENTS

```
PS D:\tec\trabajosProg> & C:/Users/dari-/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/tec/trabajosProg/python/retos/reto1_compu1/reto1_prog3.py
Ingresa cuantos elementos de la serie de fibonacci quieres que calcule el programa:20
Este es el resultado:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
PS D:\tec\trabajosProg>
```