Ing. Laura Coto Sarmiento. MSc.

Consideraciones importantes…

- Hasta no comprender el problema y poder expresarlo con sus propias palabras, no inicie el proceso de solución. Comente siempre la estrategia con su equipo de trabajo para determinar lo mejor.

- Analice pensando en las entradas, salidas y restricciones del caso y posteriormente el proceso.

- Cree las funciones respectivas para:
  - Datos de entrada y salida.
  - Validación de entradas, y responsable construir las respuestas formales y comprensibles al usuario.
  - Función de procesamiento, lo más limpia posible.
  - ~~Programe usando 2 archivos: el principal y el de procesos (su librería).~~
  - Programe el manejo de excepciones.
  - No olvidé documentar cada detalle como lo enseñado en clase.
  - Cada def debe tener al menos un return y la respectiva documentación.
  - ~~Coloque todo dentro de un menú.~~

Retos numéricos iterativos

Reto 1 – Compare si dos números son iguales (Debe llamar a cuentarDigitos y usar ciclos)

Entrada		***Salida***
123	34456	La cantidad dígitos debe ser igual.
4356	4356	Los números son iguales.
6987	5987	Los números no son iguales
0	0	Los números son iguales.

Reto 2 – Verificar la paridad de un dígito en una posición dada.

Codificar una función para determinar si el dígito que está en la posición "x" de un número "n" es par.

Algunas consideraciones son las siguientes:

- x no puede ser mayor a la cantidad de dígitos de un número.
- x tampoco puede ser un valor negativo, debe ser entero.
- N debe ser un número entero positivo mayor o igual a 1.

Entrada	***Salida***
esParDigitoEnPosicion(125.22,3)	Número y posición deben ser números enteros
esParDigitoEnPosicion(1242,0)	La posición debe ser mayor a 1
esParDigitoEnPosicion(-45,1)	El número debe ser mayor o igual a 1
esParDigitoEnPosicion(1242,8)	La posición indicada es inválida
esParDigitoEnPosicion(12521,3)	False
esParDigitoEnPosicion(12481,4)	True

Reto 3 – Dados 2 valores numéricos, eliminar en el primero los valores que se repiten en el segundo, los valores deben ser diferentes:

Entrada 1	***Entrada 2***	***Salida***
24375	345	27
12	21	Todos los valores fueron eliminados
234	987	234

Reto 4 – Desarrolle la función repetirDigito. Recibe tres valores enteros:
- Un dígito a buscar (entre 0 y 9)
- Un número analizado (>=0)
- Una cantidad de repeticiones (>=1)

Cada vez que aparezca el dígito a buscar en el número analizado, ese dígito debe repetirse en este número la cantidad de veces indicada por el tercer parámetro. Forme y retorne un número que cumpla con este requerimiento. El orden de los dígitos en el resultado es el mismo orden de la entrada según muestran los ejemplos del funcionamiento:

Entrada	***Salida***
repetirDigito(4, 124564583, 2)	1244564458 3
repetirDigito(1, 18117, 3)	1118111117
repetirDigito(9, 18117, 5)	18117

Reto 5 –Rotemos a la Izquierda

Construya una función en Python llamada rotarIzq (num) que reciba un número entero y mueva cada dígito una posición hacia adelante, el dígito menos significativo lo pone como dígito más significativo. Ejemplo:

Entrada	***Salida***
rotarIzq(“432”)	Debe ingresar un número mayor a 3 dígitos
rotarIzq(432)	Debe ingresar un número mayor a 3 dígitos
rotarIzq(4321)	1432
rotarIzq(10000)	1000
rotarIzq(100001)	110000

Reto 6 –Número perfecto

Un número perfecto es un número natural que es igual a la suma de sus divisores propios positivos. Así,

6 es un número perfecto porque sus divisores propios son 1, 2 y 3;

Y

6 = 1 + 2 +3.

Debe codificar una función que determine si un número natural es o no un número perfecto. Cree la función **esPerfecto**(n).

Entrada	***Salida***
esPerfecto(25.33)	Debe ingresar un número entero
esPerfecto(-11)	El número debe ser mayor a 0
esPerfecto(6)	True
esPerfecto(18)	False
esPerfecto(28)	True
esPerfecto(496)	True