RETO 4

PROBLEMA DEL TRANSPORTE

Crear una nueva carpeta llamada reto4, con un archivo transporte.py, con las siguientes funciones:

EJERCICIO 1

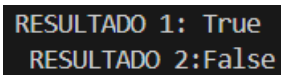
Crear una función *calcularTotal*, que recibe una lista con números. Estos números corresponden a la cantidad de productos que habrá en el origen o los que necesita en los destinos. La función se barre todos los ítems de la lista y los va sumando en una variable. Al final retorna el valor de la suma de todos los ítems.

EJERCICIO 2

Crear una función *validarListas*, recibe como parámetros dos listas: listaOrigen y listaDestino. Invoca a *calcularTotal*, para conocer el total de productos de listaOrigen y guarda el resultado en una variable. Invoca a *calcularTotal*, para conocer el total de productos de listaDestino y guarda el resultado en una variable

Si ambas listas tienen el mismo total de productos retorna True, caso contrario retorna False.

Ejecutar el archivo *TestValidar.py*. Resultado esperado:



EJERCICIO 3

Crear una función llamada *resolverProblema* que reciba dos listas llamadas ***listaOrigen*** y ***listaDestino***.

La ***listaOrigen*** tendrá los valores de lo que se produce en los orígenes del problema.

La ***listaDestino*** tendrá los valores de la cantidad de productos que requiere cada destino del problema.

Así por ejemplo, si el problema es este:

listaOrigen llegaría a la función con los valores [300,200,100]

listaDestino llegaría a la función con los valores [100,300,200,300]

Para resolver el problema, vamos a ejecutar el siguiente algoritmo:

1. Creamos una lista sin elementos llamada movimientos, en la cual se van a registrar como Strings los movimientos necesarios para resolver el problema.
2. Vamos a iterar los *destinos* y por cada *destino*, vamos a ir accediendo a cada uno de los orígenes. Es decir tenemos un for anidado, donde el for externo es de los destinos y por cada destino tenemos un for interno de los orígenes. Como parte del algoritmo vamos a modificar los valores de ambas listas conforme avanza el programa, por tal motivo debemos usar la opción que utiliza los índices para iterar las listas.
3. Para hacer una primera prueba donde se note que se está combinando cada destino, con cada origen, simplemente vamos a imprimir los valores que se van obteniendo de la listaDestino y de la listaOrigen, con el formato:

   “destino: <valorDestino> - origen: <valorOrigen>”

Ejecutar el archivo *TestCombinar.py* y deberíamos tener los resultados:

```
****PROBLEMA 1*****
   ORIGENES: [200, 300, 200]
   DESTINOS: [100, 200, 400]
destino:100 - origen:200
destino:100 - origen:300
destino:100 - origen:200
destino:200 - origen:200
destino:200 - origen:300
destino:200 - origen:200
destino:400 - origen:200
destino:400 - origen:300
destino:400 - origen:200
```

4. Ahora además de imprimir el valor de origen y valor de destino, vamos a compararlos, para determinar el valor a restar, que sería el menor de los valores entre valorOrigen y valorDestino. Guardamos este valor en una variable llamada cantidadMovimiento.

5. Con el índice del origen y el índice del destino que estamos analizando en la iteración actual, creamos un movimiento, que será expresado como String, en este formato “<indiceOrigen> - <indiceDestino> - <candidadMovimiento>”, por ejemplo “0 - 0 - 100” que significaría desde el origen 0, hacia el destino 0, se mueven 100 unidades de producto. Agregamos este movimiento a la lista de movimientos.

6. Restamos la cantidad obtenida en el paso 5, de la lista de orígenes y de lista de destinos, en las posiciones respectivas que estamos iterando. Como restamos los valores de las listas, para volver a hacer las comparaciones en el for interno, es importante volver a obtener el valor de la lista de destinos.

7. Realizar un debug e ir validando que el algoritmo se ejecute tal como se hizo en los videos es decir que se van disminuyendo los valores de las listas.

8. Tomar en cuenta que si ya se llega a cero en el destino, ya no es necesario seguir iterando los demás orígenes, podemos abandonar la iteración usando un break.

9. Eliminar los prints de su programa y al final retornar la lista de movimientos.

10. Ejecutar *TestResolver.py*, se espera este resultado:

```
****PROBLEMA 1*****
   ORIGENES: [200, 300, 200, 400]
   DESTINOS: [100, 300, 200, 300, 200]
SOLUCION: ['0-0-100', '0-1-100', '1-1-200', '1-2-100', '2-2-100', '2-3-100', '3-3-200', '3-4-200']
****PROBLEMA 2*****
   ORIGENES: [500, 300, 200, 400, 300]
   DESTINOS: [400, 500, 600, 200]
SOLUCION: ['0-0-400', '0-1-100', '1-1-300', '2-1-100', '2-2-100', '3-2-400', '4-2-100', '4-3-200']
```

EJERCICIO 4

Se tiene una lista de Strings, con esta estructura:

notas=[“1-Juan-Perez-10”, “2-Rosario-Tijeras-10”, “3-Alam Brito-9”]

Donde cada String, tiene información separada por guión, cada elemento significa:

“1-Juan-Perez-10”

```
****PROBLEMA 1*****
   ORIGENES: [200, 300, 200, 400]
   DESTINOS: [100, 300, 200, 300, 200]
SOLUCION: ['0-0-100', '0-1-100', '1-1-200', '1-2-100', '2-2-100', '2-3-100', '3-3-200', '3-4-200']
****PROBLEMA 2*****
   ORIGENES: [500, 300, 200, 400, 300]
   DESTINOS: [400, 500, 600, 200]
SOLUCION: ['0-0-400', '0-1-100', '1-1-300', '2-1-100', '2-2-100', '3-2-400', '4-2-100', '4-3-200']
```

Posición 0: Número de curso del estudiante, en este caso, pertenece al curso 1

“1-Juan-Perez-10”

Posición 1: Nombre del estudiante, en este caso Juan

“1-Juan-Perez-10”

Posición 2: Apellido del estudiante, en este caso Perez

“1-Juan-Perez-10”

Posición 3: Nota del estudiante, en este caso 10

Crear una función llamada *recuperarCurso*, que recibe como parámetro infoEstudiante, que es una cadena con la información descrita antes. La función aplica un split y retorna el número del curso, convertida a entero.

Crear una función llamada *recuperarNombreCompleto*, que recibe como parámetro infoEstudiante, que es una cadena con la información descrita antes. La función aplica un split y retorna el nombre del Estudiante, seguido del apellido, por ejemplo retorna “Juan Perez”.

Crear una función llamada *recuperarNota*, que recibe como parámetro infoEstudiante, que es una cadena con la información descrita antes. La función aplica un split y retorna la nota del estudiante, convertida a entero.

Crear una función llamada *buscarEstudiante*, que recibe como parámetros: listaEstudiantes, numeroCurso y nota. En la listaEstudiantes, tiene la información guardada como se indicó al inicio del ejercicio. La función itera toda la lista de estudiantes y busca el primer estudiante que coincidan: el número de curso y la nota que se está buscando. Cuando encuentre el estudiante, retorna su nombre completo.

Para recuperar la nota, el curso y el nombre completo de cada elemento de la lista, debe usar las funciones recuperarCurso, recuperarNota y recuperarNombreCompleto.

Ejecutar el archivo *TestBusqueda.py*, se espera los resultados:

```
Estudiantes: ['1-Juan-Perez-9', '2-Rosario-Tijeras-10', '1-Chito-Vera-10', '1-Ricardo
 Lopez-10', '2-John-Cena-10']
 EL PRIMER ESTUDIANTE DEL CURSO 1, CON NOTA 10, es: Chito Vera
```