### **Ejercicio de practica n° 2 ET**

Una tienda llamada **LudoStore** se especializa en la venta de **juegos de mesa**. La información de cada juego está registrada en un diccionario llamado juegos, donde las llaves son los códigos de los juegos y los valores son listas que contienen información asociada a cada juego.

juegos = {

'C001': ['Catan', 'Devir', 'Estrategia', 3, 4],

'C002': ['Dixit', 'Libellud', 'Creatividad', 3, 6],

'C003': ['Carcassonne', 'Z-Man Games', 'Estrategia', 2, 5],

'C004': ['Uno', 'Mattel', 'Familiar', 2, 10],

'C005': ['Terraforming Mars', 'Stronghold Games', 'Estrategia', 1, 5],

'C006': ['Jungle Speed', 'Asmodee', 'Reflejos', 2, 8],

...

}

código: [nombre, editorial, tipo, min\_jugadores, max\_jugadores]

También se cuenta con un diccionario inventario, donde las llaves son los mismos códigos de los juegos y los valores incluyen el precio y el stock disponible:

inventario = {

'C001': [29990, 8],

'C002': [18990, 5],

'C003': [25990, 0],

'C004': [6990, 20],

'C005': [39990, 3],

'C006': [15990, 10],

...

}

### 

### 

### 

### **Funcionalidades requeridas**

Debes crear un programa en Python, el programa debe tener el siguiente **menú principal**:

\*\*\* MENÚ PRINCIPAL \*\*\*

1. Stock por editorial

2. Búsqueda por precio

3. Actualizar precio

4. Salir

### **1. Stock por editorial**

Debe entregar el **stock total** de juegos de una **editorial** ingresada por el usuario, sin importar si la escribe en mayúsculas o minúsculas.  
Función a implementar: stock\_editorial(editorial)

### **2. Búsqueda por precio**

Debe entregar una lista con los juegos que estén **dentro de un rango de precios** ingresado por el usuario, siempre que tengan **stock mayor a 0**.  
Formato de salida: ['Nombre -- Código'] ordenados alfabéticamente.  
Validar que el rango ingresado sea de tipo entero; si no lo es, mostrar:  
"Debe ingresar valores enteros!!"  
Si no hay juegos en ese rango, mostrar:  
"No hay juegos en ese rango de precios."  
Función a implementar: busqueda\_precio(p\_min, p\_max)

### **3. Actualizar precio**

Permite actualizar el precio de un juego dado su código.  
 Función a implementar: actualizar\_precio(codigo, nuevo\_precio)

* Si el código no existe → retornar False
* Si se actualiza correctamente → retornar True  
   En el programa principal:
* Si True → mostrar "Precio actualizado!!"
* Si False → mostrar "El código no existe!!"  
   Luego preguntar si desea actualizar otro juego (s/n)

### **4. Salir**

Finaliza el programa mostrando:  
 "Programa finalizado."