# Sistema de Gestión de Inventario para un Taller de Automóviles

## Fecha de Entrega:

21 dic 2023 -14:00

## Descripción y Objetivos:

Desarrollar un programa en Python que simule un sistema de gestión de inventario para un taller de reparación de automóviles. Este sistema permitirá al usuario añadir, modificar, eliminar y consultar piezas de automóviles, así como realizar seguimiento de las existencias.

### Requisitos Funcionales:

1. **Gestión de Piezas:**
    - Añadir nuevas piezas al inventario (nombre, tipo, cantidad, precio).
    - Modificar los datos de las piezas existentes.
    - Eliminar piezas del inventario.
    - Consultar las piezas disponibles (búsqueda por nombre, tipo, etc.).
2. **Control de Inventario:**
    - Visualizar el inventario actual (lista de todas las piezas con sus detalles).
    - Actualizar la cantidad de una pieza (por ejemplo, después de una venta o compra).
    - Alerta cuando la cantidad de alguna pieza esté por debajo de un umbral mínimo (2 piezas).
3. **Interfaz de Usuario:**
    - Crear una interfaz de consola sencilla para interactuar con el sistema.

    - Menú interactivo con opciones para realizar todas las funciones anteriores.

4. **Funciones y Estructuras de Datos:**
    - Utilizar funciones para organizar el código (por ejemplo, una función para añadir piezas, otra para la búsqueda, etc.).
    - Emplear listas y/o tuplas para almacenar la información de las piezas.
5. **Inventario de demostración:**
    - Debe existir un fichero main que contenga una demostración de todos los métodos principales.

## Requisitos No Funcionales:

- **Legibilidad del Código:** El código debe estar bien organizado y comentado adecuadamente.
- **Manejo de Errores:** El programa debe ser capaz de manejar situaciones de error (por ejemplo, entrada de usuario no válida) de manera elegante.
- **Documentación:** Incluir un archivo README con instrucciones de uso y descripción de las funcionalidades.

## Estructura de una Pieza:

Cada pieza en el inventario puede representarse como un diccionario o una instancia de una clase.

Si optas por un diccionario:

```
pieza = {
    'nombre': 'Filtro de aire',
    'tipo': 'Filtro',
    'cantidad': 10,
    'precio': 15.0
}
```

Si optas por usar una clase:

```
class Pieza:
    def __init__(self, nombre, tipo, cantidad, precio):
        self.nombre = nombre
        self.tipo = tipo
        self.cantidad = cantidad
```

```
        self.precio = precio

    def __str__(self):
        return f"Nombre: {self.nombre}, Tipo: {self.tipo},
Cantidad: {self.cantidad}, Precio: {self.precio}"
```

## Métodos Principales:

1. **anyadir(Pieza):**
    - Método para añadir una nueva pieza al inventario.
    - Deberá verificar si la pieza ya existe; si es así, actualizar la cantidad.
2. **modificar(Pieza):**
    - Método para modificar los detalles de una pieza existente.
    - Permitir cambiar el nombre, tipo, cantidad y precio.
3. **eliminar(Pieza):**
    - Método para eliminar una pieza del inventario.
4. **buscar(string):**
    - Método para buscar piezas por nombre o tipo.
    - Devuelve una lista de piezas que coincidan con los criterios de búsqueda.
5. **mostrar_inventario:**
    - Método para mostrar todas las piezas en el inventario.

## Ejemplos de Uso

## 1. Añadir Piezas al Inventario

Para añadir una nueva pieza al inventario, se creará un diccionario con los detalles de la pieza y se pasará a la función `anyadir`. Si la pieza ya existe, se actualizará su cantidad.
Ejemplo:

```
pieza = {'nombre': 'Filtro de aire', 'tipo': 'Filtro', 'cantidad':
5, 'precio': 15.0}
anyadir(inventario, pieza)
```

## 2. Modificar Piezas Existentes

Para modificar una pieza, se creará un diccionario con los nuevos detalles y se pasará, junto con el nombre de la pieza a modificar, a la función `modificar`.
Ejemplo:

```
nueva_pieza = {'nombre': 'Filtro de aire', 'tipo': 'Filtro
mejorado', 'cantidad': 5, 'precio': 20.0}
modificar(inventario, 'Filtro de aire', nueva_pieza)
```

## 3. Eliminar Piezas del Inventario

Para eliminar una pieza, se pasará el nombre de la pieza a la función `eliminar`.
Ejemplo:

```
eliminar(inventario, 'Filtro de aire')
```

## 4. Buscar Piezas

Para buscar piezas por nombre o tipo, se pasará el criterio de búsqueda a la función `buscar`.
Ejemplo:

```
resultados = buscar(inventario, 'Filtro')
```

## 5. Mostrar el Inventario Completo

Para visualizar todas las piezas en el inventario, se llamará a la función `mostrar_inventario`.
Ejemplo:

```
inventario_actual = mostrar_inventario(inventario)
print(inventario_actual)
```

## 6. Actualizar la Cantidad de una Pieza

Para actualizar la cantidad de una pieza específica, se pasará el nombre de la pieza y la nueva cantidad a la función `actualizar_cantidad`.
Ejemplo:

```
actualizar_cantidad(inventario, 'Filtro de aire', 10)
```

## Entrega y Evaluación:

- **Código Fuente:** Entregar todos los archivos de código fuente (.py).
- **Demostración:** Preparar un breve video o presentación que muestre el funcionamiento del programa.
- **Informe:** Redactar un informe que incluya una explicación del código, los problemas encontrados y cómo se resolvieron.