Rumbo al Proyecto final Full Stack Python con objetos y API REST

Primera etapa:

Implementar un CRUD de productos y un carrito de compras utilizando arreglos y funciones.

Funciones para gestionar un arreglo con datos de productos

Escribimos una serie de funciones para crear una pequeña app que maneje un arreglo que contenga diccionarios con los datos de los productos.
Los diccionarios tienen 4 claves:

- **codigo** : numérico entero. Lo utilizamos como id.
- **descripcion** : string, descripcion del producto.
- **cantidad** : numérico entero, cantidad de items de este producto en stock,
- **precio** : numérico de punto flotante, precio unitario del producto

Las funciones que escribiremos son las siguientes:

- agregar_producto(codigo, descripcion, cantidad, precio)

Consulta un producto a partir de su código y devuelve sus datos.

Parámetros:

- codigo: int, código numérico del producto.
- descripcion: str, descripción alfabética del producto.
- cantidad: int, cantidad en stock del producto.
- precio: float, precio de venta del producto.

Retorna:

- True

```
# ----------------------------------------------------------------
# Función para agregar un producto al arreglo
# ----------------------------------------------------------------
def agregar_producto(codigo, descripcion, cantidad, precio):
    # Creamos un diccionario con los datos del producto...
    nuevo_producto = {
        'codigo': codigo,
        'descripcion': descripcion,
        'cantidad': cantidad,
        'precio': precio
    }
    # Y lo agregamos a nuestro arreglo.
    productos.append(nuevo_producto)
    return True
```

- consultar_producto(codigo)

Consultar un producto a partir de su código

Parámetros:

- codigo: int, código numérico del producto.

Retorna:

- dict: datos del producto en forma de diccionario, o False si no se encontró el producto.

```
# ----------------------------------------------------------------------
# Función para consultar un producto a partir de su código
# ----------------------------------------------------------------------
def consultar_producto(codigo):
    # Recorremos la lista de productos...
    for producto in productos:
        # Y si el código es el correcto,
        if producto['codigo'] == codigo:
            # Refresamos el diccionario correspondiente.
            return producto
    # Si el bucle finaliza sin encontrar el producto,
    # regresamos "falso."
    return False
```

- modificar_producto(codigo, nueva_descripcion, nueva_cantidad, nuevo_precio)

Modifica los datos de un producto a partir de su código.

Parámetros:

- codigo: int, código numérico del producto.
- nueva_descripcion: str, nueva descripción alfabética del producto.
- nueva_cantidad: int, nueva cantidad en stock del producto.
- nuevo_precio: float, nuevo precio de venta del producto.

Retorna:

- True si se pudo hacer la modificación, o False si no se encontró el producto.

```
# ----------------------------------------------------------------------
# Función para modificar los datos de un producto a partir de su código
# ----------------------------------------------------------------------
def modificar_producto(codigo, nueva_descripcion, nueva_cantidad, nuevo_precio):
    # Recorremos la lista de productos...
    for producto in productos:
        # Y si el código es el correcto,
        if producto['codigo'] == codigo:
            # ...actualizamos los valores de cada clave del diccionario
            producto['descripcion'] = nueva_descripcion
            producto['cantidad'] = nueva_cantidad
            producto['precio'] = nuevo_precio

            # Como no hay otro producto con ese código, salimos del bucle.
            return True
    # Si llegamos aqui, el producto no existe.
    return False
```

- listar_productos()

Muestra en pantalla un listado de los productos existentes.

Parámetros:

Retorna:

- True

```
# ----------------------------------------------------------------------
# Función para obtener un listado de los productos en pantalla
# ----------------------------------------------------------------------
def listar_productos():
    # Recorremos la lista de productos...
    print("-"*30)
    for producto in productos:
        # Y mostramos los datos de cada uno de ellos.
        print(f"Código: {producto['codigo']}")
        print(f"Descripción: {producto['descripcion']}")
        print(f"Cantidad: {producto['cantidad']}")
        print(f"Precio: {producto['precio']}")
        print("-"*30)
```

- eliminar_producto(codigo)

Elimina un producto a partir de su código.

Parámetros:

- codigo: int, código numérico del producto.

Retorna:

- True si se pudo eliminar, o False si no se encontró el producto.

```
# ----------------------------------------------------------------
# Función para modificar los eliminar un producto a partir de su código
# ----------------------------------------------------------------
def eliminar_producto(codigo):
    # Recorremos la lista de productos...
    for producto in productos:
        # Y si el código es el correcto,
        if producto['codigo'] == codigo:
            # ...lo quitamos de la lista.
            productos.remove(producto)
            # Como no hay otro producto con ese código, salimos del bucle.
            return True
    # Si llegamos aqui, el producto no existe.
    return False
```

Funciones para el manejo del "carrito de compras"

Con estas funciones, podrás agregar y quitar productos al carrito de compras y mostrar el contenido del carrito en pantalla.
Se realizan las verificaciones necesarias para asegurarse de que el producto exista en el arreglo de productos y que la cantidad en stock sea suficiente.
Los diccionarios que representan los elementos en el carrito tienen las mismas claves que los diccionarios que representan a los productos.

- **codigo** : numérico entero. Lo utilizamos como id.
- **descripcion** : string, descripcion del producto.
- **cantidad** : numérico entero, cantidad de items de este producto en stock,
- **precio** : numérico de punto flotante, precio unitario del producto

Las funciones que escribiremos son las siguientes:

- agregar_al_carrito(codigo, cantidad)

Agrega un producto al carrito de compras.
Se verifica que el item exista en el arreglo de productos y que la cantidad sea suficiente para realizar la operación. Si el producto ya existe en el carrito, en lugar de agregar un item nuevo en el arreglo, unicamente se actualiza la cantidad.

Parámetros:

- codigo: int, código numérico del producto a agregar.
- cantidad: int, cantidad del producto a agregar.

Retorna:

- True si se pudo agregar el producto al carrito, False si no se encontró el producto o la cantidad en stock es insuficiente.

```
def agregar_al_carrito(codigo, cantidad):
    # Vemos si existe el producto...
    producto = consultar_producto(codigo)

    # Si se devolvio un falso, el producto no existe.
    if producto is False:
        print("El producto no existe.")
        return False
```

```
    # Vemos si hay cantidad suficiente de ese producto...
    if producto['cantidad'] < cantidad:
        print("Cantidad en stock insuficiente.")
        return False

    # Verificar si el producto ya está en el carrito
    for item in carrito:
        if item['codigo'] == codigo:
            # Si existe, sumo a la cantidad del carrito...
            item['cantidad'] += cantidad
            # ...y descuento del stock de ese producto.
            producto['cantidad'] -= cantidad
            return True

    # Si el producto no está en el carrito, se agrega como un nuevo item
    nuevo_item = {
        'codigo': codigo,
        'descripcion': producto['descripcion'],
        'cantidad': cantidad,
        'precio': producto['precio']
    }
    carrito.append(nuevo_item)

    # ...y descuento del stock de ese producto.
    producto['cantidad'] += cantidad
    return True
```

- quitar_del_carrito(codigo, cantidad)

Quita un producto del carrito de compras.
Se verifica que el item exista en el arreglo de productos y que la cantidad sea suficiente para realizar la operación. Si se puede realizar la operación, tambien se restituye la cantidad que se quita del carrito al stock. Si la cantidad queda en cero, se elimina el item del carrito.

Parámetros:

- codigo: int, código numérico del producto a quitar.
- cantidad: int, cantidad del producto a quitar.

Retorna:

- True si se pudo quitar el producto del carrito, False si no se encontró el producto en el carrito o la cantidad a quitar es mayor a la cantidad en el carrito.

```
# ----------------------------------------------------------------
# Función para quitar un producto del carrito de compras
# ----------------------------------------------------------------
def quitar_del_carrito(codigo, cantidad):
    # Recorremos la lista de productos en el carrito...
    for item in carrito:
        # Si se ecuentra ese producto...
        if item['codigo'] == codigo:
            # Si no hay cantidad suficiente en el carrito...
            if cantidad > item['cantidad']:
                print("Cantidad a quitar mayor a la cantidad en el carrito.")
                return False

            # Si llegue aqui, es que puedo descontarlos del carrito...
            item['cantidad'] -= cantidad

            # ...y reponerlos en el stock de productos!
            producto = consultar_producto(codigo)
            modificar_producto(codigo, producto["descripcion"], producto["cantidad"]-cantidad, producto["precio"] )

            # Compruebo si la cantidad de ese producto en el carrito es cero:
            if item['cantidad'] == 0:
                # Si quedo en cero, lo elimino del carrito.
                carrito.remove(item)
            return True

    # Si el bucle finaliza sin novedad, es que ese producto NO ESTA en el carrito.
    print("El producto no se encuentra en el carrito.")
    return False
```

- mostrar_carrito()

Muestra en pantalla el contenido del carrito de compras.

Parámetros:

- codigo: int, código numérico del producto a agregar.
- cantidad: int, cantidad del producto a agregar.

Retorna:

- True si se pudo agregar el producto al carrito, False si no se encontró el producto o la cantidad en stock es insuficiente.

```
def mostrar_carrito():
    """
    Muestra en pantalla el contenido del carrito de compras.
    """
    # Inicializamos una variable para sumar los importes de cada item
    suma = 0
    print("-"*30)

    # Recorremos la lista de productos en el carrito
    # y mostramos los datos de cada producto.
    for item in carrito:
        print(f'Cod: {item["codigo"]} - {item["descripcion"]}')
        print(f'Precio: {item["precio"]}  Cantidad: {item["cantidad"]}')

        # Calculamos el importe a pagar por el producto...
        importe = item["precio"] * item["cantidad"]

        # ...y acumulamos el importe en la variable suma.
        suma += importe
        print(f'Importe: {importe}')
        print("-"*30)
    print(f'Importe TOTAL: {suma}')
    return True
```

Ejemplo del uso de las funciones implementadas

```
# Agregamos productos a la lista:
agregar_producto(1, 'Teclado USB 101 teclas', 10, 4500)
agregar_producto(2, 'Mouse USB 3 botones', 5, 2500)
agregar_producto(3, 'Monitor LCD 22 pulgadas', 15, 52500)
agregar_producto(4, 'Monitor LCD 27 pulgadas', 25, 78500)
agregar_producto(5, 'Pad mouse', 5, 500)
# eliminar_producto(5) # Eliminamos un producto del stock.

# Consultar un producto por su código
# producto = consultar_producto(1)
# if producto:
#     print(f"Producto encontrado: {producto['descripcion']}")
# else:
#     print("Producto no encontrado.")

# Modificar un producto por su código
# modificar_producto(1, 'Teclado mecánico 62 teclas', 20, 34000)

# Listamos todos los productos en pantalla
listar_productos()

# Agregamos productos al carrito
agregar_al_carrito(1, 2)
agregar_al_carrito(2, 1)

# agregar_al_carrito(30, 1) # intentamos agregar un producto inexistente

# Intentar agregar un producto con cantidad insuficiente
# agregar_al_carrito(1, 150)

# Quitar un producto del carrito
quitar_del_carrito(1, 1)

# Mostrar el contenido del carrito
mostrar_carrito()
```