Etapa 2

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

Segunda etapa:

Convertir las funciones vistas en la clase anterior en objetos y clases.

Vamos a adaptar el código desarrollado antes al paradigma de **objetos** en Python. Para ello, crearemos una **clase** llamada **Producto** que encapsulará los datos y las operaciones relacionadas con un producto. Luego, utilizaremos esta clase en otras clases para implementar la funcionalidad de *agregar*, *consultar*, *modificar* y *listar* productos, así como el manejo del carrito de compras.

Clase Producto:

La clase Producto es una representación de un producto en el sistema. Contiene las **propiedades** y **métodos** necesarios para manipular y acceder a los datos de un producto específico. Veamos una breve descripción de las tareas que realiza la clase Producto:

- **init(self, codigo, descripcion, cantidad, precio):** Este es el *constructor* de la clase Producto que se ejecuta al crear una *instancia* de la clase. Recibe los parámetros codigo, descripcion, cantidad y precio para inicializar las propiedades correspondientes del producto.
- **modificar(self, nueva_descripcion, nueva_cantidad, nuevo_precio):** Este método permite modificar los datos de un producto. Recibe los nuevos valores para la descripción, cantidad y precio del producto y actualiza las propiedades correspondientes.

Dentro de la clase Producto, las propiedades del producto (como codigo, descripcion, cantidad y precio) son **atributos de instancia**, lo que significa que son variables específicas de cada instancia de la clase. Cada vez que creas un objeto Producto, puedes acceder y modificar estas propiedades utilizando la notación de punto.

```
class Producto:
    # Definimos el constructor e inicializamos los atributos de instancia
    def __init__(self, codigo, descripcion, cantidad, precio):
        self.codigo = codigo           # Código del producto
        self.descripcion = descripcion # Descripción del producto
        self.cantidad = cantidad       # Cantidad disponible del producto
        self.precio = precio           # Precio del producto

    # Este método permite modificar un producto.
    def modificar(self, nueva_descripcion, nueva_cantidad, nuevo_precio):
        self.descripcion = nueva_descripcion  # Modifica la descripción del producto
        self.cantidad = nueva_cantidad        # Modifica la cantidad del producto
        self.precio = nuevo_precio            # Modifica el precio del producto
```

Asi podemos probar el funcionamiento de la clase:

```
print("\033[H\033[J")     # Limpiamos la pantalla

producto = Producto(1, 'Teclado USB 101 teclas', 10, 4500)
print(producto.codigo)       # Acceder al código del producto
print(producto.descripcion)  # Acceder a la descripción del producto
print(producto.cantidad)     # Acceder a la cantidad del producto
```

```
print(producto.precio)       # Acceder al precio del producto
print()
```

Modificar productos:
La modificación de los datos de un producto se realiza utilizando el método **modificar** de la clase **Producto**. Aquí un ejemplo de cómo utilizarlo:

```
producto.modificar(1,"Teclado Mecánico USB", 20, 4500)  # Modificar los datos del producto

print(producto.descripcion)  # Acceder a la nueva descripción del producto
print(producto.cantidad)     # Acceder a la nueva cantidad del producto
print(producto.precio)       # Acceder al nuevo precio del producto
```

En el ejemplo, agregamos un producto al inventario con el código 1 y luego lo consultamos utilizando el método **consultar_producto** de la clase **Inventario**. Si se encuentra el producto, podemos acceder a sus **propiedades** (como *descripcion*, *cantidad* y *precio*) y modificarlas directamente utilizando el método **modificar** de la clase **Producto**. Luego, podemos listar los productos actualizados utilizando el método l**istar_productos** de la clase **Inventario**.

Recordemos que en el código adaptado, los productos se manejan como objetos de la clase Producto, y las operaciones se realizan a través de los métodos de las clases correspondientes. Esto permite encapsular la lógica y los datos relacionados con los productos en objetos específicos.

Clase Inventario:

La clase Inventario es una representación de un inventario de productos en el sistema. Cuando creamos una instancia de esta clase, se inicializa una lista vacía que almacenará los productos.

Para agregar un producto al inventario, utilizamos el método **agregar_producto**. Este método recibe el código, descripción, cantidad y precio del producto que queremos agregar. Dentro del método, creamos un **objeto Producto** con estos datos y lo agregamos a la lista de productos del inventario.

Si queremos consultar un producto en el inventario, utilizamos el método **consultar_producto**, al cual le pasamos el código del producto que queremos buscar. El método busca en la lista de productos y devuelve el **objeto Producto** correspondiente si se encuentra, o **False** si no se encuentra ningún producto con el código dado.

Además, la **clase Inventario** nos proporciona el método **listar_productos**, el cual muestra en pantalla un listado de todos los productos existentes en el inventario. Recorre la lista de productos y muestra las propiedades de cada producto, como el código, descripción, cantidad y precio.

Estas funcionalidades nos permiten gestionar el inventario de productos de manera eficiente y realizar operaciones como agregar nuevos productos, consultar su información y obtener una lista actualizada de los productos disponibles.

```
class Inventario:
    # Definimos el constructor e inicializamos los atributos de instancia
    def __init__(self):
        self.productos = []  # Lista de productos en el inventario (variable de clase)


    # Este método permite crear objetos de la clase "Producto" y
    # agregarlos al inventario.
    def agregar_producto(self, codigo, descripcion, cantidad, precio):
        nuevo_producto = Producto(codigo, descripcion, cantidad, precio)
        self.productos.append(nuevo_producto)  # Agrega un nuevo producto a la lista


    # Este método permite consultar datos de productos que están en el inventario
    # Devuelve el producto correspondiente al código proporcionado o False si no existe.
    def consultar_producto(self, codigo):
        for producto in self.productos:
            if producto.codigo == codigo:
                return producto
        return False
```

```
    # Este método permite modificar datos de productos que están en el inventario
    # Utiliza el método consultar_producto del inventario y modificar del producto.
    def modificar_producto(self, codigo, nueva_descripcion, nueva_cantidad, nuevo_precio):
        producto = self.consultar_producto(codigo)
        if producto:
            producto.modificar(nueva_descripcion, nueva_cantidad, nuevo_precio)


    # Este método imprime en la terminal una lista con los datos de los
    # productos que figuran en el inventario.
    def listar_productos(self):
        print("-"*30)
        for producto in self.productos:
            print(f"Código: {producto.codigo}")
            print(f"Descripción: {producto.descripcion}")
            print(f"Cantidad: {producto.cantidad}")
            print(f"Precio: {producto.precio}")
            print("-"*30)


    # Este método elimina el producto indicado por codigo de la lista
    # mantenida en el inventario
    def eliminar_producto(self, codigo):
        for producto in self.productos:
            if producto.codigo == codigo:
                self.productos.remove(producto)
                print("Producto eliminado.")
                break
        else:
            print("Producto no encontrado.")
```

Veamos como es el funcionamiento de la clase:

- **Constructor (init(self)):** El constructor de la clase Inventario se ejecuta al crear una instancia de la clase. No requiere parámetros y se encarga de inicializar la lista de productos vacía.
- **agregar_producto(self, codigo, descripcion, cantidad, precio):** Este método permite agregar un nuevo producto al inventario. Recibe los parámetros codigo, descripcion, cantidad y precio que representan los datos del producto a agregar. Dentro del método, se crea un objeto Producto con los valores proporcionados y se agrega a la lista de productos del inventario.
- **consultar_producto(self, codigo):** Este método permite consultar un producto en el inventario a partir de su código. Recibe el parámetro codigo que representa el código del producto a consultar. El método busca en la lista de productos del inventario y devuelve el objeto Producto correspondiente si se encuentra, o None si no se encuentra ningún producto con el código dado.
- **modificar_producto(self, codigo, nueva_descripcion, nueva_cantidad, nuevo_precio):** Este método permite modificar datos de productos que están en el inventario. Utiliza el método consultar_producto del inventario y modificar del producto.
- **eliminar_producto(self, codigo):** Este método recibe como parámetro el código del producto que se desea eliminar. Dentro del método, se realiza un bucle que recorre la lista de productos del inventario. Se compara el código del producto actual con el código proporcionado. Si se encuentra un producto con el código coincidente, se utiliza el método remove para eliminar ese producto de la lista self.productos. Luego se muestra un mensaje indicando que el producto ha sido eliminado. Si el bucle termina sin encontrar un producto con el código dado, se muestra un mensaje indicando que el producto no fue encontrado.
- **listar_productos(self):** Este método muestra en pantalla un listado de todos los productos existentes en el inventario. Recorre la lista de productos y muestra las propiedades de cada producto (como código, descripción, cantidad y precio) en un formato legible.

Asi podemos probar el funcionamiento de la clase:

```
# Crear una instancia de la clase Inventario
mi_inventario = Inventario()

# Crear una instancia de la clase Carrito
mi_carrito  = Carrito()
```

```
# Agregar productos al inventario
mi_inventario.agregar_producto(1, 'Teclado USB 101 teclas', 10, 4500)
mi_inventario.agregar_producto(2, 'Mouse USB 3 botones', 5, 2500)
mi_inventario.agregar_producto(3, 'Monitor LCD 22 pulgadas', 15, 52500)
mi_inventario.agregar_producto(4, 'Monitor LCD 27 pulgadas', 25, 78500)
mi_inventario.agregar_producto(5, 'Pad mouse', 5, 500)

# Consultar un producto en el inventario
producto = mi_inventario.consultar_producto(2)
if producto:
    print("Producto encontrado:")
    print(f"Código: {producto.codigo}")
    print(f"Descripción: {producto.descripcion}")
    print(f"Cantidad: {producto.cantidad}")
    print(f"Precio: {producto.precio}")
else:
    print("Producto no encontrado.")

# Modificar un producto en el inventario
mi_inventario.modificar_producto(3, "Nuevo Producto 3", 20, 34.99)

# Listar todos los productos en el inventario
print("Lista de productos en el inventario:")
inventario.listar_productos()
```

En este ejemplo, creamos una instancia de la **clase Inventario** llamada inventario. Luego, utilizamos el método **agregar_producto** para agregar tres productos al inventario, cada uno con un código, descripción, cantidad y precio específicos.

Después, utilizamos el método **consultar_producto** para buscar un producto en el inventario utilizando su código. Si el producto se encuentra, se muestra su información. En caso contrario, se muestra un mensaje indicando que el producto no fue encontrado.

A continuación, utilizamos el método **modificar_producto** para actualizar los datos de un producto en el inventario. En este caso, modificamos el producto con código 3 cambiando su *descripción*, *cantidad* y *precio*.

El método **eliminar_producto** recibe como parámetro el código del producto que se desea eliminar. Se realiza un bucle que recorre la lista de productos del inventario y se compara el código del producto actual con el código proporcionado. Si se encuentra un producto con el código coincidente, se elimina ese producto de la lista y se muestra un mensaje indicando que el producto ha sido eliminado. Si el bucle termina sin encontrar un producto con el código dado, se muestra un mensaje indicando que el producto no fue hallado.

Por último, utilizamos el método **listar_productos** para mostrar en pantalla la lista completa de productos en el inventario, incluyendo los cambios realizados.

NOTA: Debes asegurarte de que el código del producto sea único y no se repita en el inventario, ya que la eliminación se basa en el código del producto y al dar las altas no estamos validando esta situación. Más adelante el inventario se implementa con una tabla de la base de datos, y podemos hacer que el código o id sea una unico.

Clase Carrito:

La clase Carrito representa un carrito de compras donde se pueden agregar y quitar productos. Permite realizar operaciones como agregar un producto al carrito, quitar un producto del carrito y mostrar el contenido del carrito.

La clase Carrito tiene los siguientes métodos:

- **agregar_producto(codigo, cantidad):** Este método permite agregar un producto al carrito de compras. Recibe como parámetros el código numérico del producto y la cantidad que se desea agregar. El método realiza las siguientes verificaciones:

1. Verifica si el producto existe en el inventario utilizando el método consultar_producto de la clase Inventario. Si el producto no existe, muestra un mensaje de error.

2. Verifica si la cantidad en stock del producto es suficiente para agregar al carrito. Si la cantidad es insuficiente, muestra un mensaje de error.

Si el producto ya está en el carrito, actualiza la cantidad del producto en el carrito sumándole la cantidad deseada.

Si el producto no está en el carrito, lo agrega como un nuevo ítem al carrito.

- **quitar_producto(codigo, cantidad):** Este método permite quitar un producto del carrito de compras. Recibe como parámetros el código numérico del producto y la cantidad que se desea quitar. El método realiza las siguientes verificaciones:

1. Verifica si el producto está en el carrito. Si no se encuentra en el carrito, muestra un mensaje de error.

2. Verifica si la cantidad a quitar es mayor a la cantidad del producto en el carrito. Si es mayor, muestra un mensaje de error. Si la cantidad a quitar es menor o igual a la cantidad del producto en el carrito, disminuye la cantidad del producto en el carrito en la cantidad deseada. Si la cantidad resultante es cero, elimina el producto del carrito.

- **mostrar_contenido():** Este método muestra en pantalla el contenido del carrito de compras. Recorre la lista de productos en el carrito y muestra la información de cada producto, como el código, la descripción, la cantidad y el precio.

La clase Carrito utiliza el método consultar_producto de la clase Inventario para verificar la existencia de los productos en el inventario y obtener su información.

Para utilizar la clase Carrito, se debe crear una instancia de la misma y luego se pueden llamar a sus métodos para realizar las operaciones de agregar, quitar y mostrar el contenido del carrito.

```
class Carrito:
    # Definimos el constructor e inicializamos los atributos de instancia
    def __init__(self):
        self.items = []  # Lista de items en el carrito (variable de clase)


    # Este método permite agregar productos del inventario al carrito.
    def agregar(self, codigo, cantidad, inventario):
        # Nos aseguramos que el producto esté en el inventario
        producto = inventario.consultar_producto(codigo)
        if producto is False:
            print("El producto no existe.")
            return False

        # Verificamos que la cantidad en stock sea suficiente
        if producto.cantidad < cantidad:
            print("Cantidad en stock insuficiente.")
            return False

        # Si existe y hay stock, vemos si ya existe en el carrito.
        for item in self.items:
            if item.codigo == codigo:
                item.cantidad += cantidad
                # Actualizamos la cantidad en el inventario
                producto = inventario.consultar_producto(codigo)
                producto.modificar(producto.descripcion, producto.cantidad - cantidad, producto.precio)
                return True

        # Si no existe en el carrito, lo agregamos como un nuevo item.
        nuevo_item = Producto(codigo, producto.descripcion, cantidad, producto.precio)
        self.items.append(nuevo_item)
        # Actualizamos la cantidad en el inventario
        producto = inventario.consultar_producto(codigo)
        producto.modificar(producto.descripcion, producto.cantidad - cantidad, producto.precio)
        return True


    # Este método quita unidades de un elemento del carrito, o lo elimina.
    def quitar(self, codigo, cantidad, inventario):
        for item in self.items:
            if item.codigo == codigo:
                if cantidad > item.cantidad:
                    print("Cantidad a quitar mayor a la cantidad en el carrito.")
                    return False
                item.cantidad -= cantidad
                if item.cantidad == 0:
                    self.items.remove(item)
                # Actualizamos la cantidad en el inventario
                producto = inventario.consultar_producto(codigo)
                producto.modificar(producto.descripcion, producto.cantidad + cantidad, producto.precio)
                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

    def mostrar(self):
        print("-"*30)
        for item in self.items:
            print(f"Código: {item.codigo}")
            print(f"Descripción: {item.descripcion}")
            print(f"Cantidad: {item.cantidad}")
            print(f"Precio: {item.precio}")
            print("-"*30)
```

Asi podemos probar el funcionamiento de la clase y todo lo implementado hasta aqui:

```
# ----------------------------------------------------------------
# Ejemplo de uso de las clases y objetos definidos antes:
# ----------------------------------------------------------------
print("\033[H\033[J")     # Limpiamos la pantalla

# Crear una instancia de la clase Inventario
mi_inventario = Inventario()

# Crear una instancia de la clase Carrito
mi_carrito  = Carrito()

# Agregar productos al inventario
mi_inventario.agregar_producto(1, 'Teclado USB 101 teclas', 10, 4500)
mi_inventario.agregar_producto(2, 'Mouse USB 3 botones', 5, 2500)
mi_inventario.agregar_producto(3, 'Monitor LCD 22 pulgadas', 15, 52500)
mi_inventario.agregar_producto(4, 'Monitor LCD 27 pulgadas', 25, 78500)
mi_inventario.agregar_producto(5, 'Pad mouse', 5, 500)
# mi_inventario.modificar_producto(1,"Teclado Mecánico USB", 20, 4500)

# Consultar un producto en el inventario
# producto = mi_inventario.consultar_producto(2)
# if producto:
#     print("Producto encontrado:")
#     print(f"Código: {producto.codigo}")
#     print(f"Descripción: {producto.descripcion}")
#     print(f"Cantidad: {producto.cantidad}")
#     print(f"Precio: {producto.precio}")
# else:
#     print("Producto no encontrado.")


mi_inventario.listar_productos() # Mostramos el inventario

# Agregar productos al carrito
mi_carrito.agregar(1, 2, mi_inventario )  # Agregar 2 unidades del producto con código 1 al carrito
mi_carrito.agregar(3, 1, mi_inventario )  # Agregar 1 unidad del producto con código 3 al carrito
mi_carrito.quitar (1, 1, mi_inventario )  # Quitar 1 unidad del producto con código 1 al carrito

mi_carrito.mostrar()             # Mostramos el contenido del carrito
mi_inventario.listar_productos() # Mostramos el inventario
```