Módulo 3 – E-commerce CLI (Fundamentos de Python)

1) Propósito
Desarrollar una **aplicación de consola en Python** que modele el comportamiento básico de un **ecommerce**:

- Mostrar un catálogo de productos.
- Permitir agregar productos a un carrito.
- Calcular el total a pagar.

Usando únicamente contenidos del **Módulo 3**: tipos de datos, condicionales, ciclos, estructuras de datos y funciones.

2) Alcance del ejercicio (MVP)

2.1. Catálogo de productos

- Definir en el código un **catálogo inicial** con **mínimo 5 productos**.
- Cada producto debe tener:
  - id (entero y único),
  - nombre,
  - categoría (ejemplo: "ropa", "tecnología", "hogar"),
  - precio (número > 0).

Sugerencia: usar una **lista de diccionarios** o un **diccionario** donde la llave sea el id.

2.2. Menú principal
Al ejecutar el programa, se debe ver un menú similar:

Bienvenido/a a tu Ecommerce

1) Ver catálogo de productos
2) Buscar producto por nombre o categoría
3) Agregar producto al carrito
4) Ver carrito y total
5) Vaciar carrito
0) Salir

El menú debe repetirse hasta que el usuario elija la opción **0) Salir**.

2.3. Carrito de compras

El carrito será una estructura en memoria (por ejemplo, una lista) donde se almacenan los productos seleccionados.

Funcionalidad mínima:

- **Agregar producto al carrito (opción 3)**
  - Pedir el id del producto.
  - Pedir la **cantidad** (entero > 0).
  - Validar que el id exista. Si no existe, mostrar mensaje de error.
  - Guardar en el carrito el producto y la cantidad.
- **Ver carrito y total (opción 4)**
  - Listar los ítems del carrito:
    - id, nombre, cantidad, precio unitario, subtotal.
  - Mostrar el **total a pagar**: suma de todos los subtotales.
- **Vaciar carrito (opción 5)**
  - Dejar el carrito vacío y mostrar un mensaje de confirmación.

2.4. Búsqueda de productos (opción 2)

- Permitir buscar productos por **nombre** o por **categoría**.
- Mostrar todos los productos que coincidan con el texto ingresado (ej.: si escribe “ropa”, mostrar todas las prendas).

3) Requisitos técnicos

El código debe:

- Estar escrito en **Python 3** y ejecutarse desde consola:
  python ecommerce_m3.py
- Usar **tipos de datos** básicos:
  - int, float, str, bool.
- Usar **estructuras de datos**:
  - Al menos **una** estructura compuesta para el catálogo (lista o diccionario).
  - Al menos **una** estructura compuesta para el carrito (lista o diccionario).
- Usar **condicionales** (if, elif, else) para:
  - Validar opciones del menú.
  - Validar que la cantidad sea > 0.
  - Mostrar mensajes distintos según si el carrito está vacío o no.
- Usar **ciclos**:
  - Un ciclo while para el menú principal.
  - Ciclos for o while para recorrer catálogo y carrito al mostrarlos.
- Usar **funciones**:
  - Mínimo **3 funciones** claramente separadas, por ejemplo:
    - mostrar_menu()
    - listar_productos(catalogo)
    - buscar_productos(catalogo)
    - agregar_al_carrito(catalogo, carrito)
    - mostrar_carrito_y_total(carrito)
  - Al menos **una función** debe recibir parámetros y **retornar un valor**.
- Usar nombres en **snake_case**, identación correcta y algunos **comentarios breves**.

4) Entregables

- Archivo Python: ecommerce_m3.py.
- (Opcional si se pide en el curso) un README.txt o README.md con:
  - Breve descripción del programa.
  - Cómo ejecutarlo.

Rúbrica de evaluación

Criterio	Excelente (3 pts)	Adecuado (2 pts)	Básico (1 pt)	Insuficiente (0 pts)
Modelado de datos	Catálogo y carrito bien estructurados; totales siempre correctos.	Estructuras correctas con pequeños detalles; totales casi siempre correctos.	Datos se manejan, pero hay errores o modelos poco adecuados.	No se logra manejar bien catálogo o carrito; totales incorrectos.
Menú y flujo (ciclos)	Menú claro; ciclo se repite hasta salir; opciones funcionan sin fallas.	Menú entendible; algún problema menor en el flujo.	Menú confuso o incompleto; el ciclo se corta o se vuelve engorroso.	Sin ciclo estable; el programa termina o se bloquea fácilmente.
Condicionales y validaciones	if/elif/else bien usados; valida opciones, id y cantidad; mensajes claros.	Lógica correcta en general; faltan algunas validaciones o mensajes.	Condicionales limitadas; varios casos sin validar o confusos.	No usa condicionales relevantes o la lógica es incorrecta.
Funciones y organización	≥3 funciones claras; buen uso de parámetros y retorno; código modular.	Usa funciones, pero con tareas mezcladas o poca modularidad.	Pocas funciones; la mayor parte del código en el bloque principal.	No hay funciones útiles; código en un solo bloque.
Legibilidad y mensajes	Nombres en snake_case; identación correcta; mensajes al usuario claros.	Código legible con algunos nombres/mensajes mejorables.	Código algo desordenado; mensajes poco claros.	Código muy difícil de leer; mensajes confusos o inexistentes.