Ejercicios de Python

Semana 2

23.06.2025 - 29.06.2025

Caso General

Gestión de Inventario de una Tienda Online

Cualquier negocio que venda productos, desde una pequeña tienda en Instagram hasta un gigante como Amazon, necesita un sistema para saber qué productos tiene, cuántos quedan y cuánto cuestan. Automatizar esto con código es fundamental para evitar vender productos agotados y para mantener las finanzas organizadas.

- Yamil Ayma

Nivel Fácil: Calcular el Valor Total del Inventario

Caso

El dueño de la tienda necesita saber rápidamente el valor monetario total de todos los productos que tiene actualmente en su almacén. Esto es útil para informes financieros y para tener una idea del valor del negocio.

Objetivo

Escribe una función en Python llamada calcular_valor_total que reciba un inventario como una lista de diccionarios. Cada diccionario representa un producto y contiene las claves 'nombre', 'precio' y 'stock' (la cantidad de unidades disponibles). La función debe devolver un único número flotante que represente el valor total del inventario (la suma del precio * stock de cada producto).

Ejemplo real

Un sistema contable que, al final del día, genera un reporte con el valor total de los activos en el almacén.

Pruebas de validación

```
def probar_calcular_valor_total():
    inventario1 = [
        {'nombre': 'Laptop', 'precio': 1200.00, 'stock': 5},
        {'nombre': 'Mouse', 'precio': 25.50, 'stock': 20}
    ]
    # (1200 * 5) + (25.50 * 20) = 6000 + 510 = 6510.0
    resultado1 = calcular_valor_total(inventario1)
    print(f"Prueba 1: {resultado1 == 6510.0}")

    inventario2 = []
    resultado2 = calcular_valor_total(inventario2)
    print(f"Prueba 2 (inventario vacío): {resultado2 == 0.0}")

    inventario3 = [
        {'nombre': 'Teclado', 'precio': 75.00, 'stock': 0}
    ]
    resultado3 = calcular_valor_total(inventario3)
    print(f"Prueba 3 (sin stock): {resultado3 == 0.0}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_calcular_valor_total()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👨‍👩‍👧

Nivel Medio: Procesar una Venta y Actualizar el Inventario

Caso

Un cliente realiza una compra. El sistema debe verificar si hay suficiente stock para completar la venta y, si es posible, actualizar las cantidades en el inventario. Si no hay suficiente stock de algún producto, la venta debe ser rechazada por completo para no generar inconsistencias.

Objetivo

Crea una función llamada procesar_venta que reciba dos argumentos: el inventario (un diccionario donde las claves son los nombres de los productos y los valores son otros diccionarios con 'precio' y 'stock') y un carrito_compra (una lista de diccionarios con 'nombre' y 'cantidad').

- La función primero debe verificar si hay stock suficiente para todos los productos del carrito.
- Si hay stock para todo, debe restar la cantidad vendida del stock de cada producto en el inventario y devolver True.
- Si falta stock de al menos un producto, no debe modificar el inventario y debe devolver False.

Continua en la siguiente página

Ejemplo real

El proceso de "checkout" en cualquier tienda online. Cuando haces clic en "Comprar", el sistema realiza esta validación antes de procesar tu pago.

Pruebas de validación

```
def probar_procesar_venta():
    inventario_inicial = {
        'Laptop': {'precio': 1200.00, 'stock': 5},
        'Mouse': {'precio': 25.50, 'stock': 20},
        'Teclado': {'precio': 75.00, 'stock': 10}
    }

    # Prueba 1: Venta exitosa
    carrito1 = [{'nombre': 'Mouse', 'cantidad': 2}, {'nombre': 'Teclado', 'cantidad': 1}]
    inventario_p1 = inventario_inicial.copy() # Hacemos una copia para no afectar otras pruebas
    resultado1 = procesar_venta(inventario_p1, carrito1)
    print(f"Prueba 1 (Venta exitosa): {resultado1 is True and inventario_p1['Mouse']['stock'] == 18 and
inventario_p1['Teclado']['stock'] == 9}")

    # Prueba 2: Venta fallida por falta de stock
    carrito2 = [{'nombre': 'Laptop', 'cantidad': 10}] # Solo hay 5 laptops
    inventario_p2 = inventario_inicial.copy()
    resultado2 = procesar_venta(inventario_p2, carrito2)
    print(f"Prueba 2 (Falta de stock): {resultado2 is False and inventario_p2['Laptop']['stock'] == 5}") # El stock no
debe cambiar

    # Prueba 3: Venta fallida con múltiples productos
    carrito3 = [{'nombre': 'Mouse', 'cantidad': 5}, {'nombre': 'Laptop', 'cantidad': 6}] # Falta stock de Laptop
    inventario_p3 = inventario_inicial.copy()
    resultado3 = procesar_venta(inventario_p3, carrito3)
    print(f"Prueba 3 (Falta de stock en un item): {resultado3 is False and inventario_p3['Mouse']['stock'] == 20}") # El
stock de Mouse tampoco debe cambiar

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_procesar_venta()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👷‍♂️👷‍♀️

Nivel Avanzado: Generar un Reporte de Productos a Reabastecer

Caso

Para una gestión eficiente, el encargado del almacén necesita saber qué productos están por agotarse. El sistema debe ser capaz de generar un reporte que identifique estos productos basándose en un umbral configurable.

Objetivo

Escribe una función llamada generar_reporte_reabastecimiento que reciba dos argumentos: el inventario (la misma estructura del nivel medio) y un umbral_stock (un número entero). La función debe devolver una lista de diccionarios, donde cada diccionario representa un producto cuyo stock es igual o inferior al umbral_stock. El reporte debe estar ordenado por la cantidad de stock, de menor a mayor, para priorizar los productos más urgentes. Cada diccionario en la lista de resultados debe contener 'nombre', 'stock' y un nuevo campo 'necesidad' ("Urgente" si el stock es 0, "Baja" en caso contrario).

Continua en la siguiente página

Ejemplo real

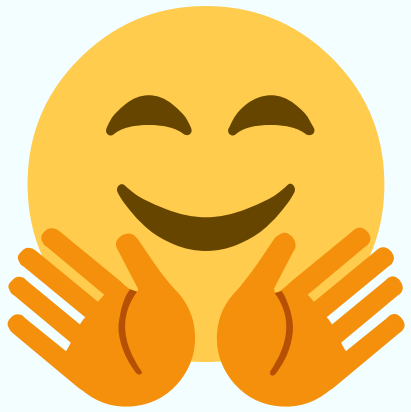
Un sistema de gestión de almacenes que envía automáticamente un correo electrónico al gerente de compras con la lista de productos que necesitan ser pedidos a los proveedores.

Pruebas de validación

```
def probar_generar_reporte_reabastecimiento():
    inventario = {
        'Laptop': {'precio': 1200.00, 'stock': 5},
        'Mouse': {'precio': 25.50, 'stock': 20},
        'Teclado': {'precio': 75.00, 'stock': 10},
        'Monitor': {'precio': 300.00, 'stock': 0},
        'Webcam': {'precio': 50.00, 'stock': 8}
    }

    # Prueba 1: Umbral de 10
    reporte1 = generar_reporte_reabastecimiento(inventario, 10)
    # Esperado: Monitor (0), Laptop (5), Webcam (8), Teclado (10)
    resultado_esperado1 = [
        {'nombre': 'Monitor', 'stock': 0, 'necesidad': 'Urgente'},
        {'nombre': 'Laptop', 'stock': 5, 'necesidad': 'Baja'},
        {'nombre': 'Webcam', 'stock': 8, 'necesidad': 'Baja'},
        {'nombre': 'Teclado', 'stock': 10, 'necesidad': 'Baja'}
    ]
    print(f"Prueba 1 (Umbral 10): {reporte1 == resultado_esperado1}")

    # Prueba 2: Umbral de 4 (solo debería encontrar Monitor)
    reporte2 = generar_reporte_reabastecimiento(inventario, 4)
    resultado_esperado2 = [
        {'nombre': 'Monitor', 'stock': 0, 'necesidad': 'Urgente'}
    ]
    print(f"Prueba 2 (Umbral 4): {reporte2 == resultado_esperado2}")

    # Prueba 3: Umbral de 30 (debería incluirlos todos, ordenados)
    reporte3 = generar_reporte_reabastecimiento(inventario, 30)
    resultado_esperado3 = [
        {'nombre': 'Monitor', 'stock': 0, 'necesidad': 'Urgente'},
        {'nombre': 'Laptop', 'stock': 5, 'necesidad': 'Baja'},
        {'nombre': 'Webcam', 'stock': 8, 'necesidad': 'Baja'},
        {'nombre': 'Teclado', 'stock': 10, 'necesidad': 'Baja'},
        {'nombre': 'Mouse', 'stock': 20, 'necesidad': 'Baja'}
    ]
    print(f"Prueba 3 (Umbral 30): {reporte3 == resultado_esperado3}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_generar_reporte_reabastecimiento()
```

Copie el código

Pegue estas lineas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 👨‍👧



Éxitos en los ejercicios

Si debuggear es el proceso de remover errores de código entonces programar debe ser el proceso de ponerlos.
- Edsger Dijkstra

Aprende mucho