Desafio Prático: Sistema de Gestão de Estoque

Objetivo:
Desenvolver um sistema de linha de comando para gerenciar o estoque de uma pequena loja. O programa deverá permitir ao usuário adicionar novos produtos, remover produtos existentes, listar todos os itens do estoque e calcular o valor total de todos os produtos. Este desafio consolida os conceitos de funções, estruturas de dados (listas e dicionários) e estruturas de repetição (while e for).
Contexto:
A "Mercearia do Bairro" precisa de um sistema simples para controlar os produtos em suas prateleiras. Atualmente, eles fazem tudo no papel e estão perdendo o controle do que entra e sai. Sua tarefa é criar a primeira versão desse sistema.

Etapa 1: Estruturando os Dados

O coração do seu sistema será a estrutura de dados que armazena o estoque.

1. Crie uma lista chamada estoque.
2. Cada item nessa lista será um **dicionário**, e cada dicionário representará um produto.
3. Cada produto (dicionário) deve ter as seguintes chaves:
   - codigo: Um número inteiro único para identificar o produto.
   - nome: O nome do produto (uma string).
   - preco: O preço unitário do produto (um float).
   - quantidade: A quantidade desse produto em estoque (um inteiro).

Exemplo de como a lista estoque ficaria:

```
estoque = [
   {'codigo': 1, 'nome': 'Arroz 5kg', 'preco': 25.50, 'quantidade': 50},
   {'codigo': 2, 'nome': 'Feijão 1kg', 'preco': 8.90, 'quantidade': 70}
]
```

Etapa 2: Modularizando com Funções

Para manter seu código organizado, você deve criar uma função para cada uma das principais operações do sistema.

1. **adicionar_produto(codigo, nome, preco, quantidade)**:
   - Esta função deve receber os quatro dados de um produto como **parâmetros**.
   - Dentro dela, crie um novo dicionário com esses dados e adicione-o à lista estoque.
   - Deve retornar uma mensagem de sucesso (ex: "Produto adicionado!").
2. **listar_produtos()**:

- Não precisa de parâmetros.
- Deve usar um laço **for** para percorrer a lista estoque.
- Para cada produto no estoque, deve imprimir suas informações de forma formatada (ex: "Cód: 1 | Produto: Arroz 5kg | Preço: R$ 25.50 | Qtd: 50").
- Se o estoque estiver vazio, deve informar ao usuário.

3. **remover_produto(codigo)**:
   - Recebe o codigo do produto a ser removido como parâmetro.
   - Deve percorrer a lista estoque para encontrar o produto com o código correspondente.
   - Se encontrar, deve remover o dicionário daquele produto da lista. Use o método .remove() das listas.
   - Se não encontrar, deve informar que o produto não existe.

Etapa 3: O Menu Interativo (Laço Principal)

O programa deve rodar continuamente, exibindo um menu de opções para o usuário até que ele decida sair.

1. Use um laço **while True** para criar o loop principal do menu.
2. Dentro do laço, exiba as seguintes opções:
   --- Mercearia do Bairro ---
   1. Adicionar Produto
   2. Listar Produtos
   3. Remover Produto
   4. Sair

3. Peça ao usuário para digitar a opção desejada.
4. Use uma estrutura if/elif/else para verificar a escolha do usuário e chamar a função apropriada.
   - Se for '1', peça os dados do novo produto (código, nome, etc.) e chame a função adicionar_produto.
   - Se for '2', chame a função listar_produtos.
   - Se for '3', peça o código do produto a ser removido e chame remover_produto.
   - Se for '4', use a palavra-chave **break** para encerrar o laço while e finalizar o programa com uma mensagem de despedida.

⭐ Desafio Bônus (Opcional) ⭐

Se você terminar o desafio principal, tente implementar estas melhorias:

1. **Função para Calcular Valor Total:** Crie uma função calcular_valor_total() que percorra o estoque e retorne a soma do valor de todos os produtos (preço *

quantidade de cada um). Adicione uma opção "5. Ver valor total do estoque" no menu.

2. **Listagem Ordenada com Lambda:** Modifique a função listar_produtos() para que ela possa listar os produtos em ordem. Peça ao usuário se ele quer ordenar por nome ou preco. Use a função sorted() junto com uma **função lambda** como chave de ordenação para realizar essa tarefa.

Exemplo para ordenar por preço:

```
# Dentro da função listar_produtos
estoque_ordenado = sorted(estoque, key=lambda produto: produto['preco'])
# Agora, itere sobre estoque_ordenado em vez do estoque original
```

Este desafio irá testar sua habilidade de combinar diferentes ferramentas do Python para criar uma aplicação funcional e bem estruturada. Boa sorte!