Desafio: Sistema de Gestão de Contatos

Objetivo:
Desenvolver um sistema completo de gerenciamento de contatos que opera via linha de comando. O programa deverá permitir ao usuário adicionar, listar, buscar e remover contatos. O diferencial deste desafio é que a lista de contatos deve ser persistente, ou seja, ela será salva em um arquivo contatos.csv e carregada toda vez que o programa iniciar, garantindo que os dados nunca sejam perdidos.
Este projeto consolida **todos os conhecimentos** adquiridos até agora:

- **Aula 1:** Lógica de programação, variáveis, entrada/saída de dados e estruturas condicionais para o menu.
- **Aula 2:** Funções para modularizar o código, listas para armazenar os contatos em memória e dicionários para representar cada contato individualmente.
- **Aula 3:** Leitura e escrita de arquivos com a biblioteca **Pandas** para carregar e salvar a agenda de contatos, garantindo a persistência dos dados.

Contexto:
Sua agenda de contatos no papel já não é mais suficiente. Você decidiu usar suas novas habilidades em Python para criar uma solução digital, um programa que não só organiza seus contatos mas também salva tudo de forma segura em seu computador.

Etapa 1: Estrutura do Projeto e dos Dados

1. **Arquivo Principal:** Crie seu script Python (ex: agenda.py).
2. **Arquivo de Dados:** O programa irá criar e gerenciar um arquivo chamado contatos.csv. Este arquivo terá as colunas: nome, telefone, email.
3. **Estrutura em Memória:** Dentro do seu programa, os contatos serão armazenados em uma **lista de dicionários**. Exemplo:
   ```
   lista_contatos = [
      {'nome': 'Ana Silva', 'telefone': '91988887777', 'email': 'ana.silva@email.com'},
      {'nome': 'Bruno Costa', 'telefone': '91999995555', 'email': 'bruno.costa@email.com'}
   ]
   ```

Etapa 2: Funções para Manipulação de Arquivos (Pandas)

Você precisará de duas funções centrais para lidar com o arquivo contatos.csv.

1. **carregar_contatos()**:
   - Esta função será chamada **no início** do programa.
   - Ela deve verificar se o arquivo contatos.csv existe.
   - Se existir, deve usar pd.read_csv() para ler os dados e retorná-los como uma lista de dicionários.
   - Se **não existir**, deve retornar uma lista vazia [].

- **Dica:** Use um bloco try-except para lidar com o caso de o arquivo não ser encontrado (FileNotFoundError).

2. **salvar_contatos(lista_contatos)**:
   - Esta função receberá a lista de contatos atual como **parâmetro**.
   - Ela será chamada **toda vez que uma alteração for feita** (adicionar ou remover um contato).
   - Deve converter a lista de dicionários em um DataFrame do Pandas e, em seguida, usar df.to_csv('contatos.csv', index=False) para salvar os dados no arquivo, sobrescrevendo a versão antiga.

Etapa 3: Funções do Núcleo da Aplicação

Crie funções para cada funcionalidade da agenda.

1. **adicionar_contato(lista_contatos)**:
   - Pede ao usuário o nome, telefone e email do novo contato.
   - Cria um dicionário para o novo contato e o adiciona à lista_contatos (usando .append()).
   - Exibe uma mensagem de sucesso.
2. **listar_contatos(lista_contatos)**:
   - Usa um laço for para percorrer a lista.
   - Imprime os detalhes de cada contato de forma organizada.
   - Se a lista estiver vazia, informa ao usuário.
3. **remover_contato(lista_contatos)**:
   - Pede ao usuário o email do contato que deseja remover (o email é um bom identificador único).
   - Percorre a lista para encontrar o contato correspondente.
   - Se encontrar, remove o dicionário da lista e informa o sucesso.
   - Se não encontrar, informa ao usuário.

Etapa 4: O Menu Principal Interativo

Esta é a parte que une tudo.

1. No início do seu script, chame a função carregar_contatos() para popular sua lista_contatos.
2. Crie um laço while True para o menu.
3. Exiba as opções: "1. Adicionar", "2. Listar", "3. Remover", "4. Sair".
4. Use if/elif/else para chamar as funções correspondentes.
5. **Importante:** Dentro das opções "Adicionar" e "Remover", logo após modificar a lista_contatos, chame a função salvar_contatos() para que a alteração seja imediatamente persistida no arquivo contatos.csv.
6. Na opção "Sair", use break para encerrar o programa.

⭐ Desafio Bônus (Opcional) ⭐

1. **Busca de Contato:** Implemente uma quarta funcionalidade: "Buscar Contato". O usuário

digita parte de um nome e o programa exibe todos os contatos que contêm aquele texto no nome.

2. **Backup Automático:** Modifique a função salvar_contatos para que, antes de salvar, ela crie uma cópia de segurança do arquivo antigo. Use o módulo os para renomear contatos.csv para contatos.bak.