Checkpoint 3 - Sistema de Autenticação em Python

Informações importantes:

- **Data de entrega:** Até 21/05/2025 às 23:59:59 **(impreterivelmente).**
  - Caso a entrega seja feita após o prazo acima e até 23/05/2025 às 23:59:59, haverá uma penalidade de 40% da nota.
  - A entrega não será aceita após 23/05/2025 23:59:59.
- **Forma de entrega:** coloque todos os arquivos produzidos em um arquivo zip e envie por e-mail para profelvis.fernandes@fiap.com.br.
- Alguns alunos poderão ser convidados a responder algumas perguntas sobre a entrega, como parte da avaliação.
- Caso haja suspeita de cópia entre alunos, a atividade será considerada anulada para todos os envolvidos (nota zero).

Objetivo

Desenvolver um sistema de autenticação de usuários em Python que aplique os seguintes conceitos:

- **Estruturas de dados** (dicionários, listas, tuplas)
- **Controle de fluxo** (condicionais, loops)
- **Funções**
- **Manipulação de arquivos** (leitura/escrita)
- **Tratamento de datas/horas**

Requisitos do Sistema

1. Autenticação Inicial

- O programa inicia solicitando login e senha.
- Login padrão de administrador: `admin` / senha `admin`.
- Se o usuário errar a senha 3 vezes consecutivas, o programa é encerrado.
- O programa não pode continuar enquanto o administrador não trocar sua senha inicial.

2. Armazenamento de Dados

- Os dados dos usuários devem ser salvos em um arquivo (ex: `usuarios.txt` ou `usuarios.json`).
- Cada usuário é representado por um **dicionário** com os seguintes campos:

```
{
    "nome": "Fulano da Silva",
    "login": "fulano123",
    "senha": "senhaSegura123",
    "perfil": "admin" ou "user",
    "ultimo_login": "2025-05-11 15:30:00",  # datetime formatado como string
    "tentativas_falhas": 0,
    "ativo": True
}
```

3. Funcionalidades por Perfil

Para administradores (`admin`):

- **Listar usuários:** Exibir todos os usuários (exceto senhas).
- **Alterar senha de qualquer usuário.**
- **Bloquear/desbloquear usuários.**
- **Ver histórico de último login de cada usuário.**
- **Cadastrar novos usuários**

Para usuários comuns (`user`):

- **Alterar apenas a própria senha.**

4. Regras de Negócio

- Ao fazer login com sucesso:
  - Atualizar o campo `ultimo_login` com a data/hora atual.
  - Zerar `tentativas_falhas`.
- Após 3 tentativas falhas de login:
  - Bloquear a conta (`ativo = False`).

    - Salvar a alteração no arquivo.

- Usuários bloqueados não podem logar, mesmo com credenciais corretas.

Tarefas a Serem Implementadas

1. Carregar Dados do Arquivo

- Ao iniciar o programa, carregar os usuários do arquivo para um dicionário em memória.
- Se o arquivo não existir, criar um novo com o usuário `admin` padrão.

2. Menu Interativo

- Após o login, exibir um menu de opções de acordo com o perfil:

```
[Admin]
1. Listar usuários
2. Alterar senha de usuário
3. Bloquear/desbloquear usuário
4. Sair

[User]
1. Alterar minha senha
2. Sair
```

3. Persistência de Dados

- Salvar alterações no arquivo sempre que houver modificações nos dados.

4. Tratamento de Datas

- Usar a biblioteca `datetime` para registrar o horário exato do último login.

Exemplo de Funcionamento

```
=== Sistema de Autenticação ===
Login: admin
Senha: admin
Bem-vindo, admin!
Último login: 2025-05-10 14:20:00
```

```
[Menu Admin]
1. Listar usuários
2. Alterar senha
3. Bloquear/desbloquear
4. Sair
Opção: 1

Usuários ativos:
- Maria (maria@email.com) [user]
- João (joao123) [user]
```

Critérios de Avaliação

1. **Funcionalidade (40%)**:
    - Login, validação de senha e bloqueio funcionam conforme as regras.
    - Dados são persistidos corretamente no arquivo.
2. **Organização do Código (30%)**:
    - Uso de funções para modularizar o código (ex: `carregar_usuarios()`, `salvar_usuarios()`, `autenticar_usuario()`).
    - Tratamento de erros (ex: arquivo não encontrado, entradas inválidas).
3. **Documentação (20%)**:
    - Comentários explicando partes complexas do código.
    - Relatório descrevendo a lógica implementada.
4. **Criatividade (10%)**:
    - Implementação de funcionalidades extras (ex: criptografia de senhas, registro de logs detalhados).

Entrega

- Código fonte em Python.
- Arquivo de dados de exemplo (ex: `usuarios.json`).

- Relatório em PDF explicando:
    - Estrutura do projeto.
    - Dificuldades encontradas.
    - Funcionalidades extras (se houver).