**Relatório do Projeto Final**

**Feito Por:** Victor Silva e Matheus Mallmman

**06/03/2025**

**12º GPSI**

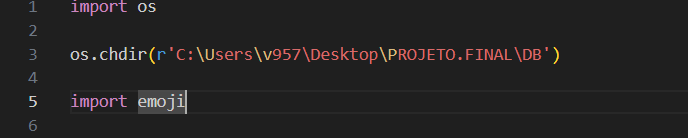
**Tema do Projeto:** biblioteca em Pythons

**Introdução:** Este projeto tem como objetivo criar uma base de dados para o armazenamento das informações pessoais dos mecânicos da oficina, permitindo as operações de leitura, eliminação, criação e atualização dos dados.

1. **APP.PY**

Este código implementa uma aplicação em Python que permite ao utilizador interagir com um menu para realizar operações em um banco de dados da oficina de mecânicos. As operações disponíveis são criar, ler, atualizar e apagar informações de mecânicos. Abaixo, explico detalhadamente o que cada parte do código faz.

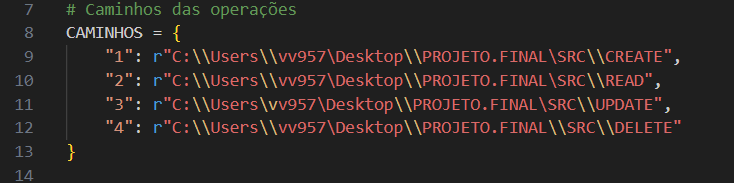
**2.Importação de Módulos**

****

**IMPORT OS:** Este módulo fornece uma maneira de interagir com o sistema operativo. Ele é utilizado aqui para mudar o diretório de trabalho, limpar o terminal, verificar se os arquivos existem e executar scripts Python.

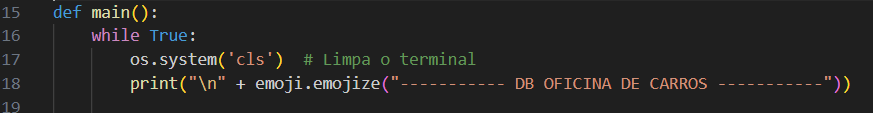
**emoji**: Esse módulo é usado para adicionar emojis nas mensagens impressas no terminal. Ele facilita a inclusão de símbolos gráficos que tornam a interface mais amigável e Dinâmico.

**3.Definição dos Caminhos das Operações**

****

**CAMINHOS:** Um dicionário que armazena os caminhos completos para os scripts de cada operação (criar, ler, atualizar, apagar). O código usa essas informações para localizar e executar os scripts correspondentes à opção escolhida pelo utilizador.

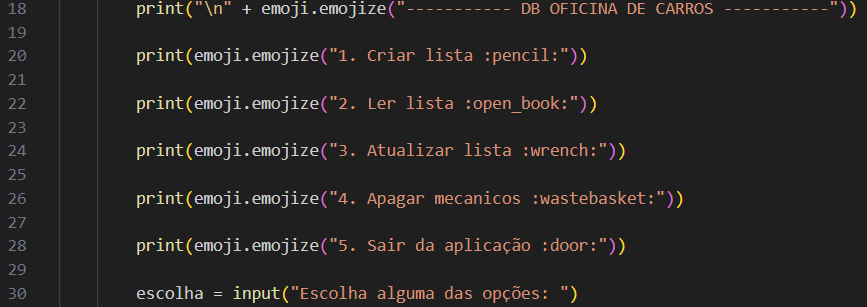
**4.Função Principal (main)**



**os.system('cls')**: Limpa a tela do terminal. Isso é útil para manter a interface limpa e mais fácil de entender.

**emoji.emojize(...)**: Exibe um título com um emoji, tornando a interface mais visual e atraente.

**5.Menu de Opções**

****

**Aqui, são exibidas as opções que o utilizador pode escolher, com o uso de emojis para tornar a interface mais interativa. O utilizador pode interagir:**

* **Criar uma lista de mecânicos**
* **Ler a lista existente**
* **Atualizar a lista**
* **Apagar registros de mecânicos**
* **Sair da aplicação**

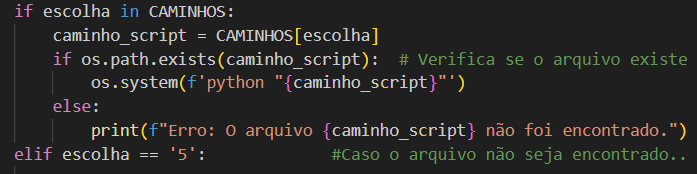
**Essas são algumas das opções que o programa que o utilizador poderá acessar.**

**6.Leitura da Opção do Utilizador**

****

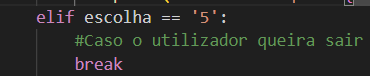
**input():** A função input captura a opção escolhida pelo utilizador. O programa então limpa a tela novamente para preparar a próxima fase**.**

**7.Execução da Operação Correspondente**

****

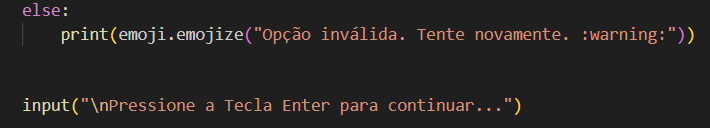
* **O código verifica se a opção escolhida pelo utilizador está presente no dicionário CAMINHOS.**
* **Se a opção estiver válida, ele verifica se o arquivo correspondente existe no diretório especificado.**
* **Se o arquivo existir, o script é executado usando o comando os.system(f'python "{caminho\_script}"').**
* **Caso contrário, exibe uma mensagem de erro dizendo que o arquivo não foi encontrado.**

**8.Opção de Sair**

****

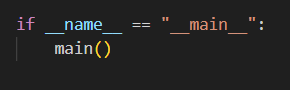
Se o utilizador escolher a opção '5', o programa termina o loop e sai da aplicação.

**9.Opção Inválida e Aguarda o Usuário Pressionar Enter**

****

* Se o utilizador digitar uma opção que não seja válida (ou seja, que não esteja entre as opções de 1 a 5), uma mensagem de erro é exibida informando que a escolha foi inválida.
* Após cada operação, o programa aguarda que o utilizador pressione **Enter** para continuar, evitando que o menu seja fechado imediatamente.

**10.Execução da Função Principal**

****

Essa linha garante que a função main() será executada quando o script for executado diretamente, e não quando for importado como um módulo em outro script.

**Resumo da parte 1 do APP.PY**

Este código oferece uma interface de linha de comando simples e interativa para realizar operações em uma base de dados de mecânicos. O utilizador pode escolher entre criar, ler, atualizar e excluir dados. Ele também é bem organizado, utilizando dicionários para gerenciar os caminhos dos scripts e métodos como os.system() para interagir com o sistema operativo. A utilização de emojis torna o terminal mais amigável e fácil de entender**.**

**Parte 2 Create.PY**

**Explicação do Código para o Relatório**

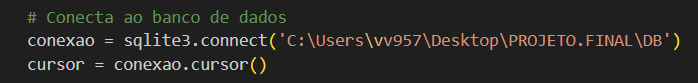
O código apresentado tem a função de criar uma tabela chamada MECANICOS em um banco de dados SQLite, caso ela ainda não exista. Aqui está uma explicação detalhada de cada parte:

1. **Importação do Módulo SQLite**

****

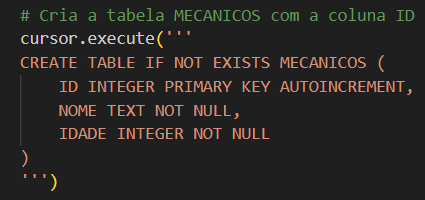
* O módulo sqlite3 permite a interação com um banco de dados SQLite, facilitando a criação de tabelas e a execução de consultas SQL.

1. **Conexão com o Banco de Dados**

****

* O comando sqlite3.connect() estabelece uma conexão com o banco de dados localizado no caminho especificado.
* Se o banco de dados não existir, ele será criado automaticamente.
* A variável cursor é usada para executar comandos SQL dentro do banco de dados.

1. **Criação da Tabela MECANICOS**

****

* O comando CREATE TABLE IF NOT EXISTS verifica se a tabela MECANICOS já existe. Se não existir, ela será criada.
* **ID INTEGER PRIMARY KEY AUTOINCREMENT**: Define a coluna ID como chave primária, garantindo que cada mecânico tenha um identificador único e numérico, que aumenta automaticamente a cada novo registro.
* **NOME TEXT NOT NULL**: Define a coluna NOME como um campo de texto obrigatório (NOT NULL), onde será armazenado o nome do mecânico.
* **IDADE INTEGER NOT NULL**: Define a coluna IDADE como um número inteiro obrigatório, onde será armazenada a idade do mecânico.

1. **Fechando a Conexão com o Banco de Dados**

****

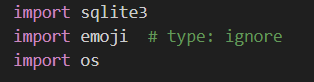
* Após a execução da consulta SQL, a conexão com o banco de dados é fechada para evitar o consumo desnecessário de recursos.

**Resumo**

* O código cria e gerencia um banco de dados SQLite.
* A tabela MECANICOS armazena informações sobre mecânicos, incluindo ID, Nome e Idade.
* A coluna ID é uma chave primária autoincrementada para garantir a unicidade dos registros.
* A conexão com o banco de dados é fechada após a execução dos comandos.

**Parte 3 Delete.PY**

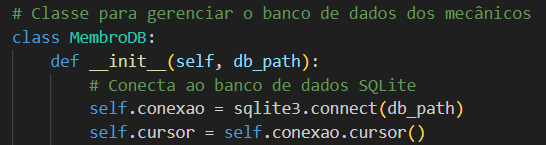
1. **Importação de Bibliotecas**

****

* sqlite3: Biblioteca para interagir com um banco de dados SQLite.
* emoji: Biblioteca para adicionar emojis ao terminal e melhorar a experiência visual.
* os: Biblioteca usada para limpar a tela do terminal e melhorar a interface do usuário.

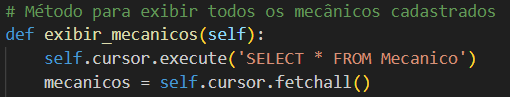
1. **Classe MembroDB**

A classe MembroDB gerencia todas as operações no banco de dados.

****

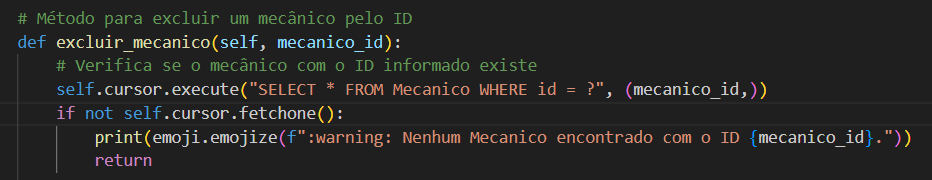
\_\_init\_\_: Inicializa a conexão com o banco de dados usando o caminho do banco (db\_path)

1. **Método exibir\_mecanicos()**

****

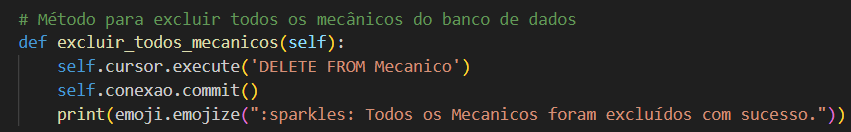
* Obtém e exibe todos os mecânicos cadastrados no banco de dados.
* Se houver mecânicos cadastrados, ele os imprime com os respectivos IDs, nomes e idades.

1. **Método excluir\_mecanico(mecanico\_id)**

****

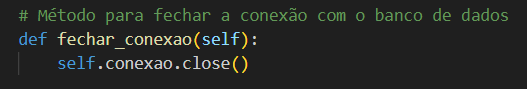
* Verifica se um mecânico com o ID fornecido existe no banco.
* Se não existir, exibe uma mensagem de erro e retorna.
* Se existir, exclui o mecânico e confirma a operação no banco de dados.

1. **Método excluir\_todos\_mecanicos()**

****

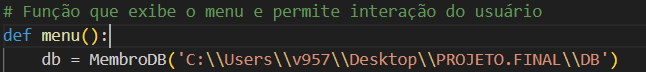
* Remove todos os registros da tabela Mecanico.
* Confirma a exclusão no banco de dados.

1. **Método fechar\_conexao()**

****

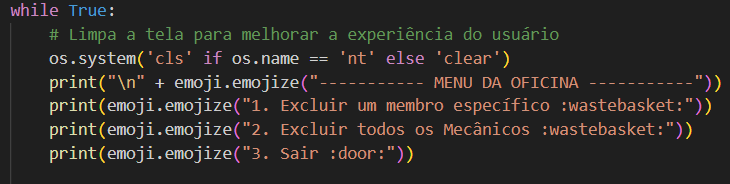
* Fecha a conexão com o banco de dados para evitar desperdício de recursos.

1. **Função menu()**

****

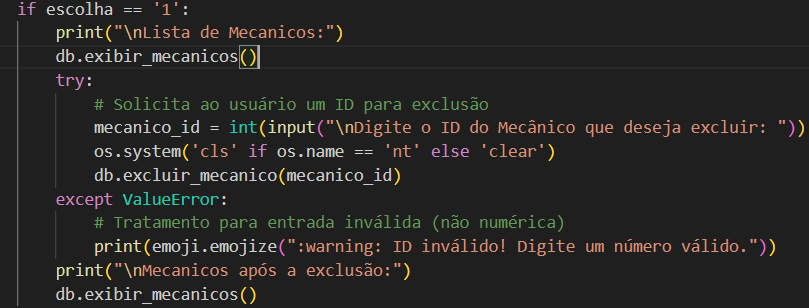
* Cria uma instância da classe MembroDB e conecta ao banco de dados.

1. **Exibição do Menu**

****

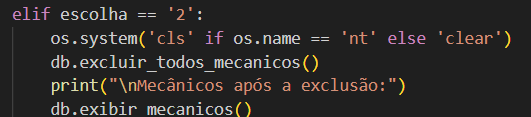
* Exibe o menu com três opções:

1. Excluir um mecânico específico
2. Excluir todos os mecânicos
3. Sair do programa
4. **Opção 1 - Excluir um Mecânico Específico**

****

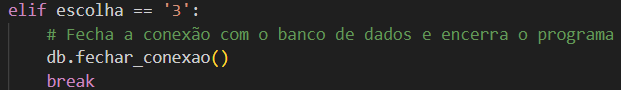
* Exibe a lista de mecânicos cadastrados.
* Solicita um ID válido ao usuário para exclusão.
* Se a entrada for inválida (não numérica), exibe um aviso.

1. **Opção 2 - Excluir Todos os Mecânicos**

****

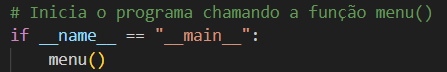
* Remove todos os mecânicos da tabela Mecanico

1. **Opção 3 - Sair do Programa**

****

* Fecha a conexão com o banco de dados e encerra o programa.

1. **Execução do Programa**

****

* Garante que o menu seja executado apenas se o arquivo for executado diretamente.

**Resumo**

* O código gerencia um banco de dados SQLite que armazena informações sobre mecânicos.
* Ele permite listar, excluir um mecânico específico ou excluir todos os mecânicos.
* Utiliza a biblioteca emoji para melhorar a experiência visual.
* A função menu() fornece uma interface interativa para o usuário.

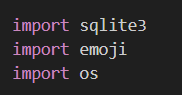
**Parte 4 Read.PY**

* **Objetivo do Código**

Este código tem como objetivo gerenciar e exibir informações sobre mecânicos cadastrados em um banco de dados SQLite. Ele oferece um menu interativo que permite ao usuário:

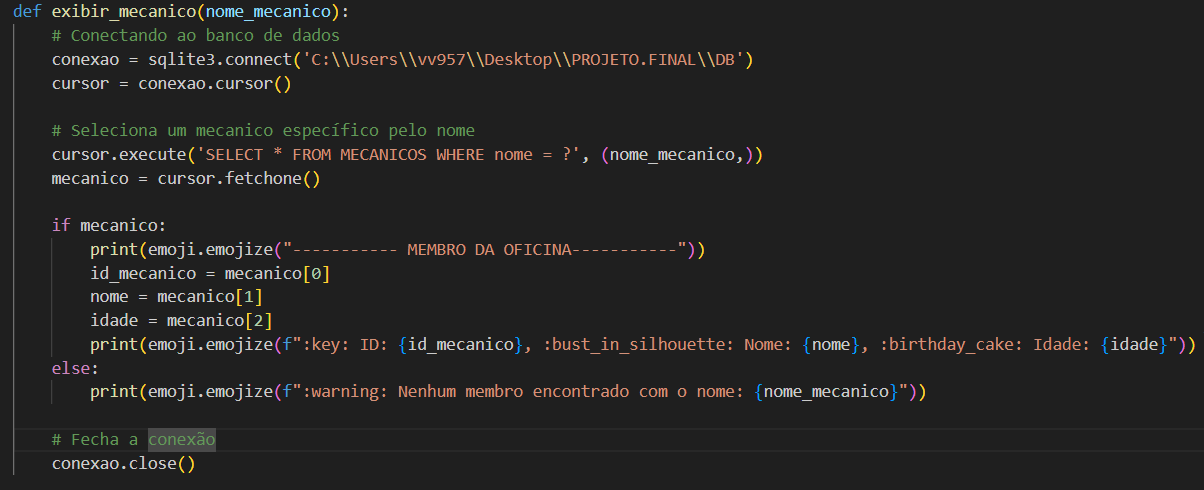
* Listar todos os mecânicos cadastrados.
* Pesquisar um mecânico pelo nome.
* Sair do programa.

1. **Importação de Bibliotecas**

****

* sqlite3: Biblioteca que permite interagir com o banco de dados SQLite.
* emoji: Biblioteca usada para adicionar emojis ao terminal, tornando a interface mais amigável.
* os: Biblioteca que permite limpar a tela do terminal para uma melhor experiência do usuário.

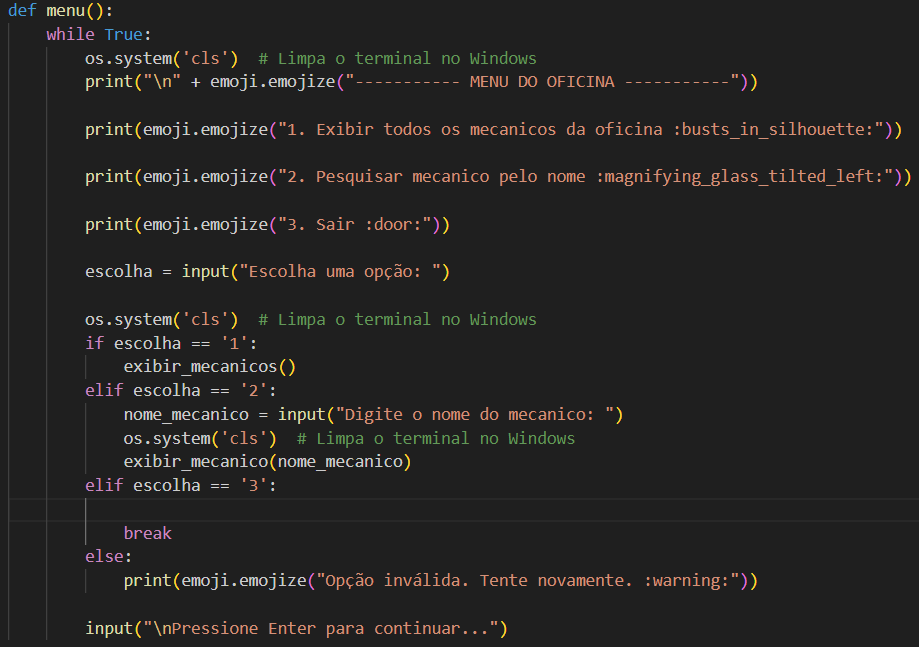
1. **Função exibir\_mecanicos()**

****

**Explicação**

* Abre uma conexão com o banco de dados.
* Consulta um mecânico pelo nome, utilizando um SELECT com filtro WHERE nome = ?.
* Se o mecânico for encontrado, exibe suas informações.
* Se não for encontrado, exibe uma mensagem de aviso.
* Fecha a conexão com o banco de dados.

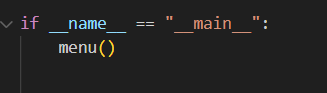
1. **Função menu()**

****

**Explicação**

* Exibe um menu interativo para o usuário.
* O usuário pode escolher entre:
* **Opção 1**: Listar todos os mecânicos.
* **Opção 2**: Pesquisar um mecânico pelo nome.
* **Opção 3**: Sair do programa.
* **Limpa a tela após cada ação** para uma melhor visualização.
* **Solicita a entrada do usuário** e executa a função correspondente.

1. **Execução do Programa**

****

* Esse trecho garante que o programa só execute o menu quando for rodado diretamente, e não se for importado como módulo em outro código.

**Conclusão**

Este código é um sistema simples para gerenciamento de mecânicos utilizando SQLite. Ele permite visualizar todos os registros, pesquisar um mecânico pelo nome e interagir via um menu no terminal.

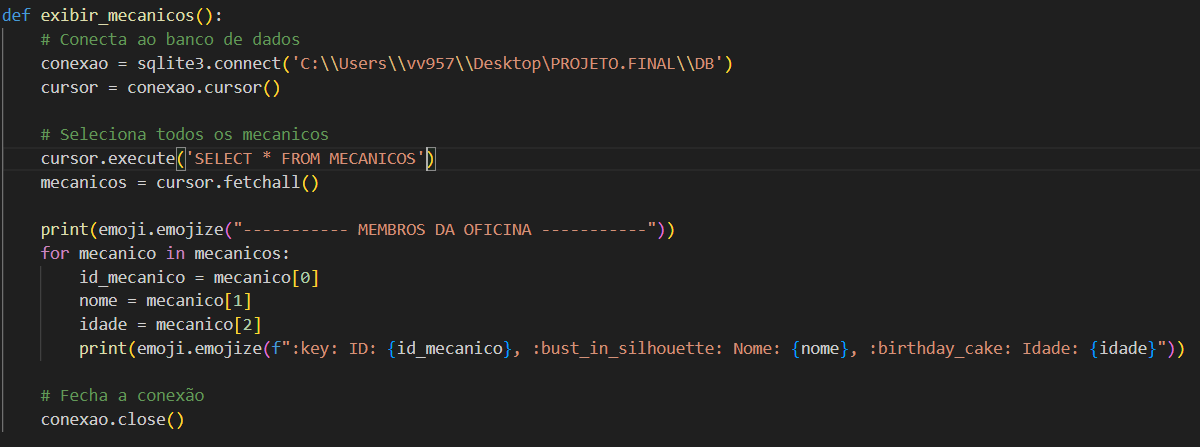
**Parte 5 (FINAL) Update.PY**

1. **Importação de Bibliotecas**

* sqlite3 → Permite a interação com o banco de dados SQLite.

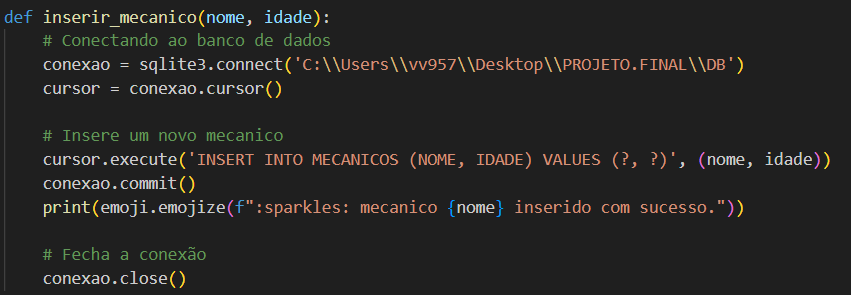
* emoji → Adiciona ícones no terminal para melhorar a experiência do usuário.
* os → Usado para limpar a tela do terminal (os.system('cls') no Windows).

1. **Função para Exibir Todos os Mecânicos**

****

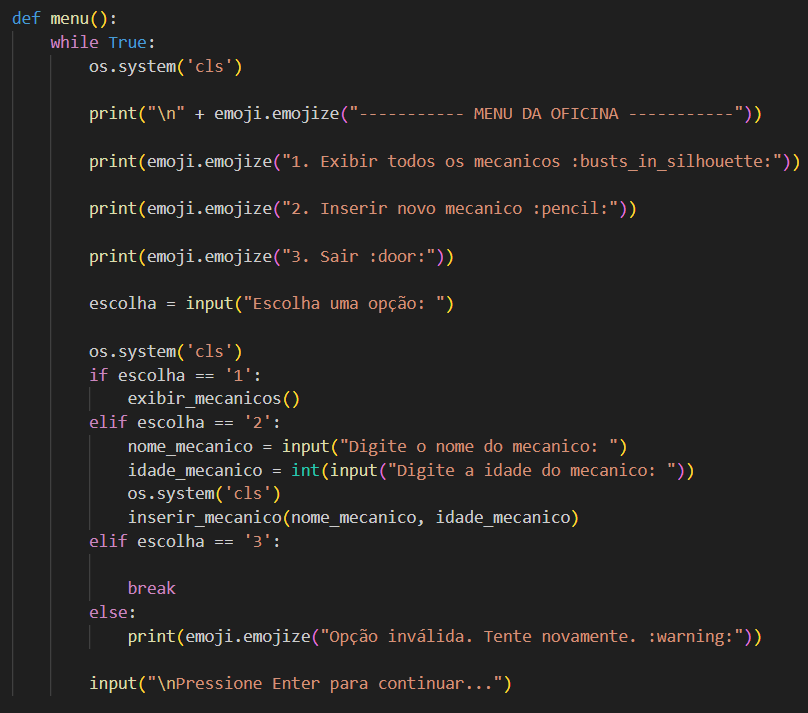
* Conecta ao banco de dados.
* Busca todos os mecânicos cadastrados na tabela MECANICOS.
* Exibe cada mecânico formatado com emojis para facilitar a visualização.
* Fecha a conexão ao final.

1. **Função para Inserir um Novo Mecânico**

****

* Conecta ao banco de dados.
* Insere um novo mecânico na tabela MECANICOS com os valores fornecidos (nome e idade).
* Confirma a operação com commit(), garantindo que os dados sejam salvos no banco.
* Exibe uma mensagem de sucesso.
* Fecha a conexão ao final.

1. **Função Principal: Menu Interativo**

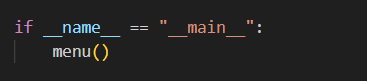
****

* Exibe um menu interativo para o usuário.
* Limpa a tela do terminal antes de exibir o menu (os.system('cls') no Windows).
* Oferece três opções:

1. Exibir todos os mecânicos → Chama exibir\_mecanicos().
2. Inserir um novo mecânico → Pede nome e idade e chama inserir\_mecanico().
3. Sair → Encerra o programa.

* Se a entrada for inválida, exibe um aviso.
* Aguarda a tecla "Enter" para continuar após cada operação.

**Execução do Programa**

****

* Verifica se o arquivo está sendo executado diretamente.
* Se sim, chama a função menu() para iniciar o programa.

**Considerações Finais**

O presente projeto propôs o desenvolvimento de um sistema simples de gerenciamento de dados de mecânicos utilizando a linguagem Python e banco de dados SQLite. O sistema foi estruturado para permitir a realização das operações essenciais de manipulação de dados, como criação, leitura, atualização e exclusão (CRUD), por meio de uma interface interativa e de fácil utilização no terminal.

Ao longo do desenvolvimento, foram utilizadas diversas bibliotecas para facilitar a execução das funcionalidades. A biblioteca sqlite3 foi crucial para a interação com o banco de dados, possibilitando a criação, modificação e consulta de registros de mecânicos. A inclusão da biblioteca emoji proporcionou uma interface mais amigável e dinâmica, tornando o terminal mais visualmente interessante. Além disso, a biblioteca os foi empregada para controlar aspectos do sistema operacional, como limpar a tela e verificar a existência de arquivos.

O código foi dividido em módulos com funcionalidades específicas, como a criação da tabela de mecânicos, a exclusão de registros, a leitura e atualização das informações armazenadas e a exibição de um menu interativo para o usuário. Essa estrutura modular garantiu clareza e organização no código, facilitando futuras modificações e manutenções.

Em termos de segurança e robustez, o sistema foi desenvolvido de forma a validar entradas e fornecer mensagens de erro apropriadas, o que contribui para uma experiência de usuário mais estável e confiável. Embora o sistema tenha sido pensado para operações básicas de manipulação de dados, ele oferece uma boa base para a expansão, podendo ser aprimorado com funcionalidades adicionais, como autenticação de usuários ou validação mais detalhada de dados.

Em conclusão, o projeto atingiu seus objetivos iniciais de forma eficaz, criando uma solução funcional para o gerenciamento de dados de mecânicos de uma oficina. A utilização de SQLite permitiu a persistência de dados de forma simples e eficiente, enquanto a interface baseada em terminal, enriquecida com emojis, tornou o uso do sistema mais agradável e intuitivo. O sistema é uma ferramenta prática e útil para gerenciar informações de mecânicos, podendo ser ampliado para atender a necessidades mais complexas no futuro.