`**Aula de Revisão _ Apresentação:*`

> https://docs.google.com/presentation/d/19zB8li61lZS1DSSRtDniO5bl47sKL2mnhgDiBP1ORLY/edit?usp=sharing

Este notebook apresenta uma solução para um problema de gestão de notas de alunos, evoluindo de lógica básica para princípios estruturados de engenharia de software. Abrange três abordagens principais:

Pseudocódigo (Estruturado): Apresenta a lógica central usando 'Portugol' para funções como adicionar, listar, atualizar, excluir alunos e calcular estatísticas.

Fluxograma (Lógica Visual): Fornece uma representação visual do fluxo do programa usando as convenções do Flowgorithm, detalhando o loop principal do menu e as sub-rotinas, incluindo uma lógica detalhada para o cálculo das estatísticas.

Linguagem Algorítmica (Python): Implementa uma aplicação Python completa usando uma lista de dicionários (turma) para armazenar os dados dos alunos. Inclui um sistema CRUD (Criar, Ler, Atualizar, Excluir) completo, um relatório estatístico detalhado e recursos fáceis de usar, como limpeza de tela e pausa de entrada. A solução Python enfatiza a modularização, tratamento de erros (try/except) e técnicas 'Pythonic' como list comprehensions para manipulação eficiente de dados.

```python
# Sistema de Notas - Desafio

# Entrada
n = int(input("Digite a quantidade de alunos: "))
notas = []

for i in range(n):
    nota = float(input(f"Digite a nota do aluno {i+1}: "))
    notas.append(nota)

# Processamento
media = sum(notas) / n
maior_nota = max(notas)
aprovados = sum(1 for nota in notas if nota >= 7.0)

# Saída
print("\n--- Resultados ---")
print("Média da turma:", media)
print("Maior nota da turma:", maior_nota)
print("Quantidade de aprovados:", aprovados)

```

```
Digite a quantidade de alunos: 2
Digite a nota do aluno 1: 5
Digite a nota do aluno 2: 8

--- Resultados ---
Média da turma: 6.5
Maior nota da turma: 8.0
Quantidade de aprovados: 1
```

```python
# Sistema de Notas - Desafio versão expandida

# Média com duas casas decimais: usando f"{media:.2f}".
#Lista de alunos com maior nota: se mais de um aluno tiver a mesma nota máxima, todos são exibidos.
#Organização: cada resultado aparece de forma clara e separada.

# Entrada
n = int(input("Digite a quantidade de alunos: "))
alunos = []  # lista de dicionários

for i in range(n):
    nome = input(f"Digite o nome do aluno {i+1}: ")
    nota = float(input(f"Digite a nota de {nome}: "))
    alunos.append({"nome": nome, "nota": nota})

# Processamento
notas = [aluno["nota"] for aluno in alunos]
media = sum(notas) / n
maior_nota = max(notas)
aluno_maior_nota = [aluno["nome"] for aluno in alunos if aluno["nota"] == maior_nota]
aprovados = sum(1 for nota in notas if nota >= 7.0)

# Saída
print("\n--- Resultados ---")
print(f"Média da turma: {media:.2f}")  # duas casas decimais
print("Maior nota da turma:", maior_nota)
print("Aluno(s) com maior nota:", ", ".join(aluno_maior_nota))
print("Quantidade de aprovados:", aprovados)
```

```python
# Sistema de Notas - Desafio versão expandida com relatório individual

```

```python
# Entrada
n = int(input("Digite a quantidade de alunos: "))
alunos = []  # lista de dicionários

for i in range(n):
    nome = input(f"Digite o nome do aluno {i+1}: ")
    nota = float(input(f"Digite a nota de {nome}: "))
    alunos.append({"nome": nome, "nota": nota})

# Processamento
notas = [aluno["nota"] for aluno in alunos]
media = sum(notas) / n
maior_nota = max(notas)
alunos_maior_nota = [aluno["nome"] for aluno in alunos if aluno["nota"] == maior_nota]
aprovados = sum(1 for nota in notas if nota >= 7.0)

# Saída geral
print("\n--- Resultados da Turma ---")
print(f"Média da turma: {media:.2f}")  # duas casas decimais
print("Maior nota da turma:", maior_nota)
print("Aluno(s) com maior nota:", ", ".join(alunos_maior_nota))
print("Quantidade de aprovados:", aprovados)

# Relatório individual
print("\n--- Relatório Individual ---")
for aluno in alunos:
    situacao = "Aprovado" if aluno["nota"] >= 7 else "Reprovado"
    print(f"{aluno['nome']} - Nota: {aluno['nota']:.2f} - {situacao}")

```

A1_ Para atender aos **Critérios da Ficha de Avaliação** (que exigem Listas, Dicionários, Funções e CRUD), foi expandido o escopo do desafio original.

Abaixo, apresento a solução nas três formas solicitadas.

---

1. Pseudocódigo (Estruturado)

Esta etapa foca na lógica pura, independente da sintaxe da linguagem de programação. Utilizamos o estilo "Portugol".

```
INÍCIO
    DEFINIR lista_alunos COMO VAZIA

    FUNÇÃO adicionar_aluno():
        LER nome
        LER nota
        CRIAR dicionário (chave="nome": nome, chave="nota": nota)
        ADICIONAR dicionário À lista_alunos

    FUNÇÃO listar_alunos():
        PARA CADA aluno EM lista_alunos:
            ESCREVER "Nome: " + aluno.nome + " | Nota: " + aluno.nota

    FUNÇÃO estatisticas_turma():
        SE lista_alunos ESTIVER VAZIA:
            RETORNAR "Sem dados"

        soma_notas = 0
        maior_nota = -1
        aprovados = 0

        PARA CADA aluno EM lista_alunos:
            soma_notas = soma_notas + aluno.nota
            SE aluno.nota > maior_nota ENTÃO maior_nota = aluno.nota
            SE aluno.nota >= 7.0 ENTÃO aprovados = aprovados + 1

        media = soma_notas / TAMANHO(lista_alunos)
        ESCREVER media, maior_nota, aprovados

    ENQUANTO VERDADEIRO:
        ESCREVER "Menu: 1.Criar, 2.Ler, 3.Atualizar, 4.Deletar, 5.Estatísticas, 6.Sair"
        LER opcao
        ESCOLHA opcao:
            CASO 1: CHAMAR adicionar_aluno()
            CASO 2: CHAMAR listar_alunos()
            CASO 3: CHAMAR atualizar_nota()
            CASO 4: CHAMAR excluir_aluno()
            CASO 5: CHAMAR estatisticas_turma()
            CASO 6: PARE O LAÇO
FIM
```

---

2. Fluxograma (Lógica Visual)

Como o sistema possui um menu interativo, o fluxograma opera em um *loop* principal que ramifica para sub-rotinas.

2. Fluxograma (Lógica Visual)

Como o sistema possui um menu interativo, o fluxograma opera em um *loop* principal que ramifica para sub-rotinas.

Descrição da Lógica do Fluxo:

1. **Início:** O programa carrega as variáveis iniciais.
2. **Nó de Decisão (Menu):** O usuário insere uma opção numérica.
3. **Ramos de Processamento:**
   - Se **Opção 1 (Create):** Recebe dados → Valida → `Append` na lista.
   - Se **Opção 3 (Update):** Busca aluno pelo nome → Se encontrar: Altera valor da chave 'nota'.
   - Se **Opção 5 (Stats):** Percorre a lista (Iteração) → Acumula valores → Calcula Média ($\frac{\sum notas}{N}$) → Exibe Resultados.
4. **Retorno:** Todos os ramos voltam para o Menu Principal, exceto a opção "Sair".

Descrição da Lógica do Fluxo:

2.1 Fluxograma no Flowgorithm

O **Flowgorithm** é uma ferramenta que padroniza a lógica usando formas geométricas universais (norma ISO 5807).

Abaixo, a lógica principal do programa simulando exatamente como ficaria no Flowgorithm.

A estrutura usa um **Laço Principal (Main Loop)** que mantém o menu ativo até que o usuário escolha "Sair".

Fluxograma Principal (Estilo Flowgorithm)

```
flowchart TD
    %% Definição de Estilos para parecer Flowgorithm
    classDef terminal fill:#f9f,stroke:#333,stroke-width:2px,rx:10,ry:10;
    classDef process fill:#ff9,stroke:#333,stroke-width:2px;
    classDef io fill:#9cf,stroke:#333,stroke-width:2px,skew:-10;
    classDef decision fill:#f96,stroke:#333,stroke-width:2px,shape:diamond;
    classDef call fill:#fff,stroke:#333,stroke-width:4px;

    Start([Início]) --> Init[Inicializar Lista 'turma']
    Init --> LoopStart{Opção != 6?}

    %% Loop do Menu
    LoopStart -- Sim --> MenuOut[/Exibir Menu: 1.Criar, 2.Ler, 3.Upd, 4.Del, 5.Stats, 6.Sair/]
    MenuOut --> InputOp[/Ler Opção/]

    InputOp --> Decisao{Qual a Opção?}

    %% Casos de Uso (Chamadas de Função)
    Decisao -- "1" --> Func1[[Chamar Adicionar_Aluno]]
    Decisao -- "2" --> Func2[[Chamar Listar_Alunos]]
    Decisao -- "3" --> Func3[[Chamar Atualizar_Nota]]
    Decisao -- "4" --> Func4[[Chamar Excluir_Aluno]]
    Decisao -- "5" --> Func5[[Chamar Estatisticas]]
    Decisao -- "Outro" --> MsgErro[/Exibir 'Opção Inválida'/]

    %% Retorno ao Loop
    Func1 --> LoopStart
    Func2 --> LoopStart
    Func3 --> LoopStart
    Func4 --> LoopStart
    Func5 --> LoopStart
    MsgErro --> LoopStart

    %% Fim do Programa
    LoopStart -- Não (Opção == 6) --> End([Fim])

    %% Aplicação das Classes
    class Start,End terminal;
    class Init process;
    class MenuOut,InputOp,MsgErro io;
    class LoopStart,Decisao decision;
    class Func1,Func2,Func3,Func4,Func5 call;
```

Legenda das Formas (Padrão Flowgorithm)

Para interpretar o desenho acima, utilize este guia:

1. **Elipse (Rosa):** Representa o **Início** e o **Fim** do algoritmo.
2. **Paralelogramo (Azul):** Representa **Entrada** (Ler do teclado) e **Saída** (Escrever na tela).
3. **Losango (Laranja):** Representa uma **Decisão** (`IF/ELSE`) ou a condição de um laço de repetição (`WHILE`). Observe que o fluxo se divide aqui.
4. **Retângulo com borda grossa (Branco):** Representa uma **Chamada de Função**. Como seu código é modular (tem funções como `adicionar_aluno`), usamos este símbolo para manter o fluxo principal limpo.

Detalhe da Lógica de Estatísticas

Se você der um "zoom" (duplo clique) no bloco **Chamar Estatísticas** dentro do Flowgorithm, a lógica interna seria esta:

1. **Verificar:** A lista está vazia?
    - *Sim:* Exibir "Sem dados" $\rightarrow$ Fim da Função.
    - *Não:* Continuar.
2. **Inicializar:** `Soma = 0`, `Maior = 0`, `Contador_Aprovados = 0`.
3. **Loop (Para cada Aluno):**
    - `Soma = Soma + Nota`
    - `Se Nota > Maior` $\rightarrow$ `Maior = Nota`
    - `Se Nota >= 7.0` $\rightarrow$ `Contador_Aprovados = Contador_Aprovados + 1`
4. **Cálculo Final:** `Media = Soma / Total_Alunos`.
5. **Saída:** Exibir Média, Maior e Aprovados.

```
'''
3. Linguagem Algorítmica (Python)
Esta implementação atende aos critérios de **Lista de Dicionários**, **Modularização** e **CRUD Completo**.
'''

import os

# Variável Global para armazenar os dados (Lista de Dicionários)
# Estrutura: [{'nome': 'Ana', 'nota': 8.5}, {'nome': 'João', 'nota': 6.0}]
turma = []

def limpar_tela():
    """Limpa o console para melhorar a apresentação."""
    os.system('cls' if os.name == 'nt' else 'clear')

# --- CRUD: CREATE ---
def adicionar_aluno():
    print("--- Adicionar Aluno ---")
    nome = input("Nome do aluno: ")
    try:
        nota = float(input("Nota do aluno (0-10): "))
        if 0 <= nota <= 10:
            # Cria o dicionário e adiciona à lista
            aluno = {"nome": nome, "nota": nota}
            turma.append(aluno)
            print(f"Sucesso: {nome} adicionado.")
        else:
            print("Erro: A nota deve ser entre 0 e 10.")
    except ValueError:
        print("Erro: Digite um valor numérico para a nota.")
    input("Pressione Enter para continuar...")

# --- CRUD: READ ---
def listar_alunos():
    print("--- Lista de Alunos ---")
    if not turma:
        print("A turma está vazia.")
    else:
        print(f"{'NOME':<20} | {'NOTA':<5} | {'STATUS'}")
        print("-" * 40)
        for aluno in turma:
            status = "Aprovado" if aluno['nota'] >= 7.0 else "Reprovado"
            print(f"{aluno['nome']:<20} | {aluno['nota']:<5.1f} | {status}")
    input("Pressione Enter para continuar...")

# --- CRUD: UPDATE ---
def atualizar_nota():
    print("--- Atualizar Nota ---")
    nome_busca = input("Nome do aluno para atualizar: ")
    encontrado = False

    for aluno in turma:
        if aluno['nome'].lower() == nome_busca.lower():
            try:
                nova_nota = float(input(f"Nova nota para {aluno['nome']}: "))
                aluno['nota'] = nova_nota
                print("Nota atualizada com sucesso.")
                encontrado = True
                break
            except ValueError:
                print("Erro: Valor inválido.")

    if not encontrado:
        print("Aluno não encontrado.")
    input("Pressione Enter para continuar...")

# --- CRUD: DELETE ---
def excluir_aluno():
    print("--- Excluir Aluno ---")
    nome_busca = input("Nome do aluno para excluir: ")

    # Uso de List Comprehension para filtrar (criar nova lista sem o aluno)
    # Poderíamos usar 'remove' ou 'pop', mas filtrar é seguro contra erros de índice
    global turma
    tamanho_inicial = len(turma)
    turma = [aluno for aluno in turma if aluno['nome'].lower() != nome_busca.lower()]

    if len(turma) < tamanho_inicial:
        print("Aluno removido.")
```

```
        print("Aluno não encontrado.")
    input("Pressione Enter para continuar...")

# --- LÓGICA E ESTATÍSTICAS (O Desafio Original) ---
def calcular_estatisticas():
    print("--- Estatísticas da Turma ---")
    if not turma:
        print("Sem dados para calcular.")
    else:
        # Extrai apenas as notas para uma lista auxiliar
        notas = [aluno['nota'] for aluno in turma]

        # Cálculos
        media = sum(notas) / len(notas)
        maior_nota = max(notas)
        # Conta quantos aprovaram (nota >= 7.0)
        aprovados = sum(1 for n in notas if n >= 7.0)

        print(f"Total de Alunos: {len(turma)}")
        print(f"Média da Turma:  {media:.2f}")
        print(f"Maior Nota:      {maior_nota:.1f}")
        print(f"Aprovados:       {aprovados} ({(aprovados/len(turma))*100:.1f}%)")

    input("Pressione Enter para continuar...")

def menu():
    while True:
        limpar_tela()
        print("=== SISTEMA DE GESTÃO DE NOTAS ===")
        print("1. Adicionar Aluno")
        print("2. Listar Alunos")
        print("3. Atualizar Nota")
        print("4. Excluir Aluno")
        print("5. Relatório Estatístico (Desafio)")
        print("6. Sair")

        opcao = input("Escolha uma opção: ")

        if opcao == '1': adicionar_aluno()
        elif opcao == '2': listar_alunos()
        elif opcao == '3': atualizar_nota()
        elif opcao == '4': excluir_aluno()
        elif opcao == '5': calcular_estatisticas()
        elif opcao == '6': break
        else:
            print("Opção inválida!")
            input()

# Execução
if __name__ == "__main__":
    menu()
```

```
3. Atualizar Nota
4. Excluir Aluno
5. Relatório Estatístico (Desafio)
6. Sair
Escolha uma opção: 2
--- Lista de Alunos ---
NOME                 | NOTA  | STATUS
----------------------------------------
Kadidja              | 9.0   | Aprovado
Kerlla               | 8.0   | Aprovado
Pressione Enter para continuar...
=== SISTEMA DE GESTÃO DE NOTAS ===
1. Adicionar Aluno
2. Listar Alunos
3. Atualizar Nota
```

6. Sair

```
6. Sair
Escolha uma opção: 4
--- Excluir Aluno ---
Nome do aluno para excluir: kadidja
Aluno removido.
Pressione Enter para continuar...5
=== SISTEMA DE GESTÃO DE NOTAS ===
1. Adicionar Aluno
2. Listar Alunos
3. Atualizar Nota
4. Excluir Aluno
5. Relatório Estatístico (Desafio)
6. Sair
Escolha uma opção:
```

Análise da Solução (Baseada na Ficha de Avaliação)

1. **Lógica e Estrutura:** O código é totalmente modular. O "programa principal" é apenas um laço `while` que chama funções específicas.
2. **Manipulação de Listas e Dicionários:** A variável `turma` é uma lista. Dentro dela, cada item é um dicionário (`{'nome': 'X', 'nota': Y}`).
3. **Eficiência:** O uso de *List Comprehension* na função `excluir_aluno` e nas estatísticas (`sum(1 for...)`) demonstra avanço (requer domínio) da linguagem Python (Pythonic code).
4. **Apresentação:** O uso da função `limpar_tela()` e menus formatados com `f-strings` (ex: `{media:.2f}`) garante uma experiência de usuário organizada.

```python
import os  # Importa a biblioteca do sistema operacional para comandos como limpar a tela

# ------------------------------------------------------------
# VARIÁVEIS GLOBAIS
# ------------------------------------------------------------
# Lista que servirá como nosso "Banco de Dados" em memória.
# Cada item desta lista será um dicionário representando um aluno.
# Exemplo de estrutura: [{'nome': 'Ana', 'nota': 9.0}, {'nome': 'Beto', 'nota': 5.5}]
turma = []

# ------------------------------------------------------------
# FUNÇÕES AUXILIARES
# ------------------------------------------------------------
def limpar_tela():
    """
    Função responsável por limpar o console.
    Verifica o sistema operacional (Windows ou Unix/Linux/Mac)
    para executar o comando correto.
    """
    # 'nt' é o nome interno do Windows. Se for ele, usa 'cls', senão 'clear'
    os.system('cls' if os.name == 'nt' else 'clear')

def pausar():
    """Pausa a execução para que o usuário possa ler a mensagem na tela."""
    input("\nPressione [Enter] para continuar...")

# ------------------------------------------------------------
# MÓDULOS DO SISTEMA (CRUD + LÓGICA)
# ------------------------------------------------------------

def adicionar_aluno():
    """
    (CREATE) Responsável por criar um novo registro na lista.
    """
    limpar_tela()
    print("--- 1. ADICIONAR ALUNO ---")

    # Etapa 1: Coleta de dados
    nome = input("Digite o nome do aluno: ")

    # Etapa 2: Tratamento de erro (Try/Except)
    # Tentamos converter a entrada para float. Se falhar, capturamos o erro.
    try:
        nota = float(input("Digite a nota do aluno (0 a 10): "))

        # Etapa 3: Validação de regra de negócio
        if 0 <= nota <= 10:
            # Cria um dicionário com os dados validados
            novo_aluno = {
                "nome": nome,
                "nota": nota
            }

            # Adiciona o dicionário à lista principal (append)
            turma.append(novo_aluno)
            print(f"\nSucesso! Aluno '{nome}' cadastrado com nota {nota}.")
        else:
            print("\nErro: A nota deve ser um valor entre 0 e 10.")

    except ValueError:
        # Executado se o usuário digitar letras ao invés de números na nota
        print("\nErro: Por favor, digite apenas números válidos para a nota (ex: 8.5).")

    pausar()

def listar_alunos():
    """
```

```
    (READ) Responsável por ler e exibir os dados armazenados.
    """
    limpar_tela()
    print("--- 2. LISTA DE ALUNOS ---")

    # Verifica se a lista está vazia antes de tentar processar
    if not turma:
        print("A turma está vazia. Nenhum aluno cadastrado.")
    else:
        # Cabeçalho da tabela com formatação de espaçamento
        print(f"{'NOME':<20} | {'NOTA':<6} | {'SITUAÇÃO'}")
        print("-" * 45) # Linha separadora visual

        # Etapa de Iteração: Percorre cada dicionário na lista
        for aluno in turma:
            nome_atual = aluno['nome']
            nota_atual = aluno['nota']

            # Lógica condicional em linha (Operador Ternário) para definir status
            situacao = "APROVADO" if nota_atual >= 7.0 else "REPROVADO"

            # Exibe os dados formatados
            # :<20 alinha à esquerda com 20 espaços
            # :.1f formata o número float com 1 casa decimal
            print(f"{nome_atual:<20} | {nota_atual:<6.1f} | {situacao}")

    pausar()

def atualizar_nota():
    """
    (UPDATE) Responsável por alterar um dado existente.
    """
    limpar_tela()
    print("--- 3. ATUALIZAR NOTA ---")

    # Coleta o nome de busca
    nome_busca = input("Qual aluno deseja alterar? ")
    aluno_encontrado = False # Variável de controle (flag)

    # Percorre a lista procurando pelo nome
    for aluno in turma:
        # Compara os nomes em minúsculo (lower) para evitar erros de digitação (Ana vs ana)
        if aluno['nome'].lower() == nome_busca.lower():
            print(f"Aluno encontrado: {aluno['nome']} (Nota atual: {aluno['nota']})")

            try:
                nova_nota = float(input("Digite a NOVA nota: "))
                if 0 <= nova_nota <= 10:
                    # Atualiza o valor da chave 'nota' no dicionário
                    aluno['nota'] = nova_nota
                    print("\nNota atualizada com sucesso!")
                    aluno_encontrado = True
                    break # Para o laço pois já encontramos o aluno
                else:
                    print("Erro: Nota fora do intervalo 0-10.")
            except ValueError:
                print("Erro: Valor não numérico.")

            # Se houve erro de validação dentro do loop, marcamos que encontramos o aluno
            # mas a atualização falhou. O break acima só ocorre no sucesso.
            aluno_encontrado = True
            break

    if not aluno_encontrado:
        print("\nAluno não encontrado na base de dados.")

    pausar()

def excluir_aluno():
    """
    (DELETE) Responsável por remover um registro da lista.
    """
    limpar_tela()
    print("--- 4. EXCLUIR ALUNO ---")

    nome_busca = input("Nome do aluno a ser excluído: ")

    # Precisamos referenciar a variável global pois vamos reescrevê-la
    global turma

    # Guarda o tamanho atual para verificar depois se algo foi removido
    tamanho_antes = len(turma)

    # --- TÉCNICA DE FILTRAGEM (List Comprehension) ---
    # Recria a lista 'turma' mantendo apenas quem tem o nome DIFERENTE do buscado.
    # Isso efetivamente "apaga" quem tiver o nome igual.
    turma = [
        aluno for aluno in turma
        if aluno['nome'].lower() != nome_busca.lower()
    ]

    tamanho_depois = len(turma)

    # Verifica se o tamanho diminuiu (significa que alguém foi removido)
    if tamanho_depois < tamanho_antes:
        print(f"\nAluno '{nome_busca}' removido com sucesso.")
    else:
        print("\nAluno não encontrado. Ninguém foi excluído.")

    pausar()

def estatisticas_turma():
    """
    (LÓGICA EXTRA) Realiza cálculos matemáticos sobre os dados.
    """
```

```
    limpar_tela()
    print("--- 5. ESTATÍSTICAS DA TURMA ---")

    if not turma:
        print("Não há dados suficientes para gerar estatísticas.")
    else:
        # Cria uma lista simples contendo apenas as notas (ex: [8.0, 5.5, 9.0])
        lista_notas = [aluno['nota'] for aluno in turma]

        # 1. Cálculo da Média (Soma das notas / Quantidade de notas)
        media = sum(lista_notas) / len(lista_notas)

        # 2. Maior Nota (Função max)
        maior_nota = max(lista_notas)

        # 3. Contagem de Aprovados
        # Gera 1 para cada nota >= 7 e soma tudo
        qtd_aprovados = sum(1 for nota in lista_notas if nota >= 7.0)

        # 4. Cálculo da Porcentagem
        porcentagem_aprov = (qtd_aprovados / len(turma)) * 100

        print(f"Total de Alunos:  {len(turma)}")
        print(f"Média da Turma:   {media:.2f}")
        print(f"Maior Nota:       {maior_nota:.1f}")
        print(f"Total Aprovados:  {qtd_aprovados} alunos ({porcentagem_aprov:.1f}%)")

    pausar()

# ------------------------------------------------------------
# PROGRAMA PRINCIPAL (MENU)
# ------------------------------------------------------------
def menu_principal():
    """
    Função controladora que exibe o menu e gerencia o fluxo.
    """
    while True: # Loop Infinito (roda até encontrar um 'break')
        limpar_tela()
        print("="*30)
        print("   SISTEMA DE GESTÃO ESCOLAR")
        print("="*30)
        print("1. Adicionar Aluno")
        print("2. Listar Alunos")
        print("3. Atualizar Nota")
        print("4. Excluir Aluno")
        print("5. Ver Estatísticas")
        print("6. Sair do Sistema")
        print("-" * 30)

        opcao = input("Escolha uma opção: ")

        # Estrutura de decisão para chamar a função correta
        if opcao == '1':
            adicionar_aluno()
        elif opcao == '2':
            listar_alunos()
        elif opcao == '3':
            atualizar_nota()
        elif opcao == '4':
            excluir_aluno()
        elif opcao == '5':
            estatisticas_turma()
        elif opcao == '6':
            print("Encerrando o sistema...")
            break # Quebra o While e encerra o programa
        else:
            print("Opção inválida!")
            pausar()

# Verifica se o script está sendo executado diretamente
if __name__ == "__main__":
    menu_principal()
```

Comentários:

Estruturas de Dados: Expliquei como a lista de dicionários funciona visualmente ([{chave:valor}]).

Controle de Fluxo: Comentários explicam o break (parada de laço) e o while True (loop infinito).

Pythonic Way: Na função excluir_aluno, expliquei detalhadamente como a* List Comprehension* filtra os dados, pois essa é uma técnica mais avançada que iniciantes costumam ter dúvida.

Tratamento de Erros: Destaquei onde o código protege contra erros do usuário (digitar letras em campo de números) usando try/except.