🧮 Tutorial: Criando uma Calculadora Python Modular

📋 Índice

🎯 Introdução

Bem-vindo ao tutorial para criar uma **calculadora Python modular**! Este projeto é perfeito para iniciantes que querem aprender os conceitos fundamentais da programação Python de forma prática e organizada.

O que você vai aprender:

- ✅ Como organizar código em módulos
- ✅ Funções e suas aplicações
- ✅ Estruturas condicionais (if/else, match/case)
- ✅ Loops (while)
- ✅ Tratamento de erros
- ✅ Manipulação de listas e dicionários
- ✅ Importação de módulos

🗂️ Estrutura do Projeto

Primeiro, vamos criar a estrutura de pastas do nosso projeto:

```
calculadora/
├── modulos/
│   ├── __init__.py
│   ├── operacoes.py
│   ├── interfaces.py
│   └── memoria.py
└── main.py
```

> 💡 **Dica:** O arquivo `__init__.py` (vazio) é necessário para que Python reconheça a pasta `modulos` como um pacote.

⚙️ Módulo de Operações

📄 Arquivo: `modulos/operacoes.py`

Este módulo contém todas as **funções matemáticas** da nossa calculadora.

🔢 Operações Básicas

```
def somar(a, b):
    """Retorna a soma de dois números."""
    resultado = a + b
    return resultado

def subtrair(a, b):
    """Retorna a subtração de dois números."""
    resultado = a - b
    return resultado

def multiplicar(a, b):
    """Retorna a multiplicação de dois números."""
    resultado = a * b
    return resultado
```

🔍 Operação com Tratamento de Erro

```
def dividir(a, b):
    """Retorna a divisão de dois números."""
    if b == 0:
        print("⚠️ Erro: Divisão por zero!")
        return None

    resultado = a / b
    return resultado
```

Conceitos aprendidos:

- **Funções**: Blocos de código reutilizáveis
- **Parâmetros**: Valores de entrada (`a`, `b`)
- **Return**: Valor de saída da função
- **Condicionais**: Verificação de erros com `if`
- **Docstrings**: Documentação das funções

🚀 Operações Avançadas

```
def potenciar(a, b):
    """Retorna a potenciação de dois números."""
```

```
    resultado = a ** b
    return resultado

def raiz_n(a, n):
    """Retorna a raiz de um número."""
    if n == 0:
        print("⚠️ Erro: Raiz de zero!")
        return None

    if a < 0 and n % 2 == 0:
        print("⚠️ Erro: Raiz de número negativo!")
        return None

    resultado = a ** (1 / n)
    return resultado
```

🔬 Algoritmos Matemáticos

```
def mdc(a, b):
    """Encontra o maior número que divide os dois números"""
    while b != 0:
        resto = a % b
        a = b
        b = resto

    return a

def mmc(a, b):
    """Encontra o menor número que é múltiplo dos dois números"""
    produto = a * b
    if produto < 0:
        produto = -produto

    divisor_comum = mdc(a, b)
    resultado = produto // divisor_comum
    return resultado
```

Conceitos aprendidos:

- **Loops while**: Repetição com condição
- **Algoritmo de Euclides**: Para calcular MDC
- **Operadores matemáticos**: `**`, `%`, `//`

🖥️ Módulo de Interfaces

📄 Arquivo: `modulos/interfaces.py`

Este módulo gerencia toda a **interação com o usuário**.

📋 Função do Menu

```python
def menu():
    """Exibe o menu da calculadora e retorna a opção escolhida."""
    print("\nMenu da calculadora:")
    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicar")
    print("4. Dividir")
    print("5. Potenciar")
    print("6. Raiz N")
    print("7. Modulo")
    print("8. MDC")
    print("9. MMC")
    print("10. Exibir histórico")
    print("0. Sair")

    opcao = int(input("Digite a sua escolha: "))
    return opcao
```

📝 Entrada de Dados

```python
def obter_numeros():
    """Solicita e retorna dois números do usuário."""
    num1 = float(input("Digite o primeiro número: "))
    num2 = float(input("Digite o segundo número: "))
    return num1, num2
```

📊 Exibição de Resultados

```python
def exibir_resultado(operacao, num1, num2, resultado):
    """Exibe o resultado da operação."""
    if resultado is not None:
        print(f"A {operacao} de {num1} e {num2} é: {resultado}")
    else:
        print("Operação não pôde ser realizada.")
```

Conceitos aprendidos:

- **Input/Output**: Entrada e saída de dados
- **Conversão de tipos**: `int()`, `float()`
- **F-strings**: Formatação de strings (`f"texto {variavel}"`)
- **Múltiplos returns**: Retornar vários valores

💾 Módulo de Memória

📄 Arquivo: `modulos/memoria.py`

Este módulo gerencia o **histórico** das operações realizadas.

📚 Estrutura de Dados

```
memoria = []
```

💾 Salvando Resultados

```
def salvar_resultado(operacao, num1, num2, resultado):
    if resultado is not None:
        memoria.append({
            "operacao": operacao,
            "num1": num1,
            "num2": num2,
            "resultado": resultado
        })
    else:
        print("Operação não pôde ser realizada.")
```

📖 Exibindo Histórico

```
def exibir_resultados():
    for registro in memoria:
        print("\n")
        print("=" * 30)
        print(f"\n{str(registro['operacao'].upper())}\n{registro['num1']} e
{registro['num2']}\nResultado: {registro['resultado']}\n")
```

Conceitos aprendidos:

- **Listas**: Estrutura de dados mutável
- **Dicionários**: Estrutura chave-valor
- **Método append()**: Adicionar elementos à lista
- **Loop for**: Iteração sobre elementos
- **Formatação de strings**: `upper()`, multiplicação de strings

🚀 Arquivo Principal

📄 Arquivo: `main.py`

Este é o **coração** do programa, onde tudo se conecta.

📦 Importações

```
from modulos.interfaces import *
from modulos.operacoes import *
from modulos.memoria import *
```

🔄 Loop Principal

```
opcao_de_parada = 0
opcao = menu()

while opcao != opcao_de_parada:
    match opcao:
        case 1:
            num1, num2 = obter_numeros()
            resultado = somar(num1, num2)
            exibir_resultado("soma", num1, num2, resultado)
            salvar_resultado("soma", num1, num2, resultado)

        case 2:
            num1, num2 = obter_numeros()
            resultado = subtrair(num1, num2)
            exibir_resultado("subtração", num1, num2, resultado)
            salvar_resultado("subtração", num1, num2, resultado)

        # ... outros cases ...

        case 10:
            exibir_resultados()

        case _:
            print("⚠️ Opção inválida!")

    opcao = menu()

print("Programa finalizado")
```

Conceitos aprendidos:

- **Importações**: Usar código de outros módulos
- **Match/Case**: Estrutura de controle moderna (Python 3.10+)
- **Loop while**: Repetição com condição
- **Padrão de programa**: Inicialização → Loop → Finalização

▶️ Executando o Programa

1. Preparação

```
# Navegue até a pasta do projeto
cd calculadora

# Execute o programa
python3 main.py

# ou

py main.py
```

2. Fluxo de Execução

1. **Inicialização**: Programa exibe o menu
2. **Entrada**: Usuário escolhe uma opção
3. **Processamento**: Programa executa a operação
4. **Saída**: Resultado é exibido e salvo
5. **Repetição**: Menu é exibido novamente
6. **Finalização**: Usuário escolhe sair (0)

🎓 Conceitos Aprendidos

🧮 Estrutura e Organização

- **Modularização**: Separação de responsabilidades
- **Pacotes e módulos**: Organização do código
- **Importações**: Reutilização de código

🔧 Programação Básica

- **Funções**: Definição, parâmetros, retorno
- **Variáveis**: Tipos, escopo, conversões
- **Estruturas condicionais**: `if/else`, `match/case`
- **Loops**: `while`, `for`

📊 Estruturas de Dados

- **Listas**: Criação, manipulação, iteração
- **Dicionários**: Chaves, valores, acesso
- **Strings**: Formatação, métodos

🛡️ Tratamento de Erros

- **Validação de entrada**: Verificação de divisão por zero
- **Retorno condicional**: `None` para erros
- **Mensagens de erro**: Feedback ao usuário

💡 Boas Práticas

- **Documentação**: Docstrings nas funções
- **Nomes descritivos**: Variáveis e funções claras
- **Separação de responsabilidades**: Cada módulo tem sua função
- **Reutilização**: Funções chamadas múltiplas vezes

🎉 Parabéns!

Você criou uma calculadora Python completa e modular! Este projeto demonstra conceitos fundamentais da programação e serve como base para projetos mais complexos.

🚀 Próximos Passos

- Adicione mais operações matemáticas
- Implemente uma interface gráfica
- Adicione salvamento em arquivo
- Crie testes unitários
- Melhore o tratamento de erros

Continue programando e explorando! 🐍✨