

## Contexto de Aprendizagem - Parada Obrigatória 2: Alta Qualidade em Software

Tiago Alexandre Leme Barbosa

### Relatório de Bug do arquivo Calculadora.py

Durante a execução da tarefa foram identificados alguns bugs no código disponibilizado. Com base inclusive em ferramentas de revisão de código, ao todo, foram encontrados 10 bugs no código. Elenco abaixo, três deles que abordam as principais funções da calculadora. Primeiro, no *bug 01* resolvo o problema da adição que está retornando valores errados; *bug 02* trato as divisões por 0 e por fim *bug 3* trato os aspectos relacionados ao input de dados.

Além da discussão de quais são os bugs encontrados e quais foram as soluções implementadas, no trabalho apresento o código corrigido e também um arquivo com os testes realizados.

#### Bug 01- Adição retorna valor equivocado

**Título do Bug:** Adição soma incorretamente

**Localização do Bug:** função adicao.

**Descrição Clara do Bug:** A função adicao(x, y) implementa return x + np.add(x, y), o que soma x duas vezes mais y. Ou seja, para x=2, y=3, o resultado é  $2 + (2+3) = 7$  em vez de 5.

**Contexto:** Ambiente padrão Python 3.x com numpy disponível. Impacta qualquer uso da função adicao em código ou testes.

**Passos para Reproduzir:**

Abrir um terminal/REPL Python com o diretório contendo Calculadora.py.

Importar a função: from Calculadora import adicao.

Executar adicao(2, 3).

**Consistência:** O bug ocorre sempre (determinístico).

Resultado Esperado: 5. Resultado Obtido: 7.

Evidências: saída do REPL >>> adicao(2,3) 7.

**Severidade:** Alta — afeta operação matemática básica.

**Impacto:** Quebra a confiança nas operações aritméticas e invalida quaisquer cálculos que dependam dessa função.

**Sugestões para Correção:** Mudar a implementação para return x + y.

**Data de Identificação:** 09/11/2025

**Status:** Resolvido.

### Bug 02 — divisão não trata divisão por zero

**Título do Bug:** Divisão causa ZeroDivisionError

**Localização:** divisao.

**Descrição:** return x / y sem checagem de y == 0. Chamadas com y=0 lançarão ZeroDivisionError e encerram o programa.

**Passos para reproduzir:** *divisao(10, 0) -> exception.*

**Resultado Esperado:** Mensagem de erro tratada ou retorno controlado. Resultado Obtido: ZeroDivisionError não tratado.

**Severidade:** Média-Alta.

**Sugestões:** envolver em try/except ZeroDivisionError ou checar antes de dividir.

**Status:** Resolvido.

### Bug 03 — Entrada e validação de tipos frágil

**Título do Bug:** Falta validação robusta de entradas do usuário

**Localização:** em múltiplas partes de calculadora\_cientifica() onde float(input(...)) e int(input(...)) são usados sem try/except.

**Descrição:** Entradas inválidas (texto, strings vazias) levantam ValueError e encerram o programa.

**Sugestões:** implementar funções utilitárias input\_float() e input\_int() com laço/try/except.

**Severidade:** Média.

**Status:** Resolvido.

### Resoluções dos Bugs

**Bug 1:** Corrigir adicao(x, y) para return x + y. (Bug 01)

**Bug 2:** Atualizar divisao(x, y) para checar y == 0 e retornar uma exceção tratada (lançar ValueError com mensagem clara) ou retornar mensagem —

**Bug 3** - Adicionar utilitários input\_float(prompt) e input\_int(prompt) que repetem o prompt até o usuário inserir um número válido (tratamento de ValueError).