**Alta Qualidade em Software**

**Ciência de Dados e Inteligência Artificial – Senai Cimatec**

**Contexto de Aprendizagem**

**Parada Obrigatória 2**

**Docente:** Sergio Martins Fernandes

**Discente:** Guilherme Tourinho Embiruçu

**Índice**

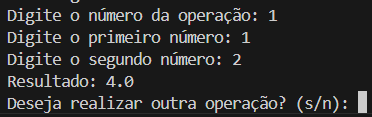
1. **Link GitHub..........................................................................................3**
2. **Relatório de Bugs...................................................................................3**
   * **Bug 1: Soma errada na função adição........................................3**
   * **Bug 2: Permitindo divisão por 0.................................................5**
   * **Bug 3: Erro ao calcular fatorial....................................................7**
   * **Bug 4: Erro ao calcular o logaritmo natural................................9**
   * **Bug 5: Erro ao calcular o logaritmo base 10.............................11**
   * **Bug 6: Função de seno não implementada................................13**
3. **Metodologia TDD...............................................................................15**
   * **Bug 1: Soma errada na função adição.......................................15**
   * **Bug 2: Permitindo divisão por 0................................................17**
4. **Observações Finais..............................................................................17**

**Link GitHub**

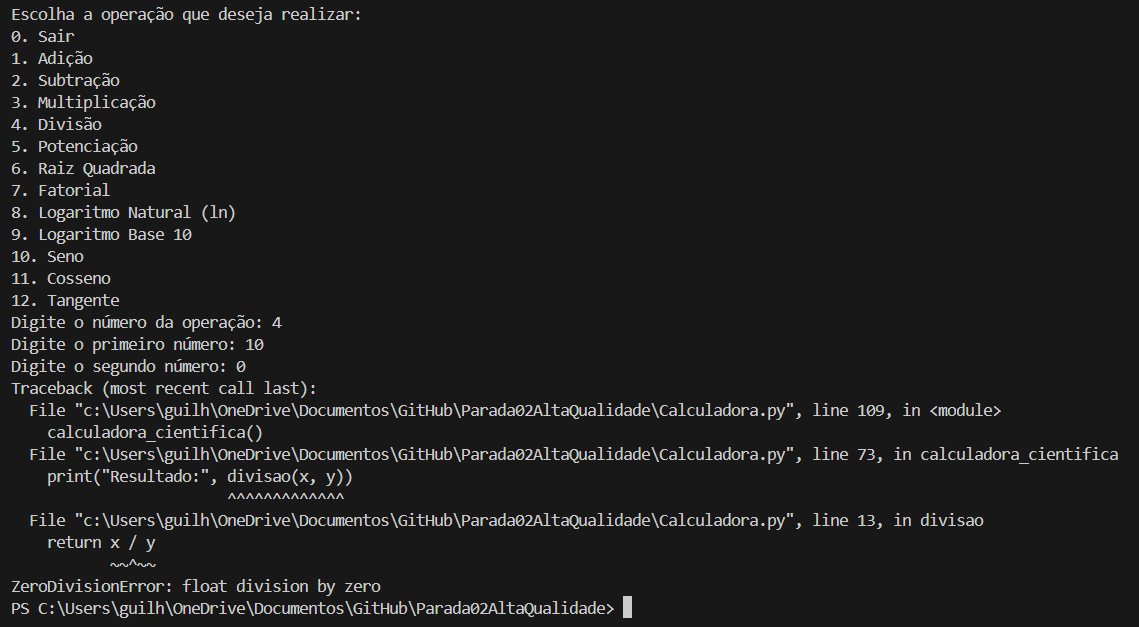
* [**https://github.com/GTECimatec/Parada02AltaQualidade.git**](https://github.com/GTECimatec/Parada02AltaQualidade.git)

**Relatório de Bugs:**

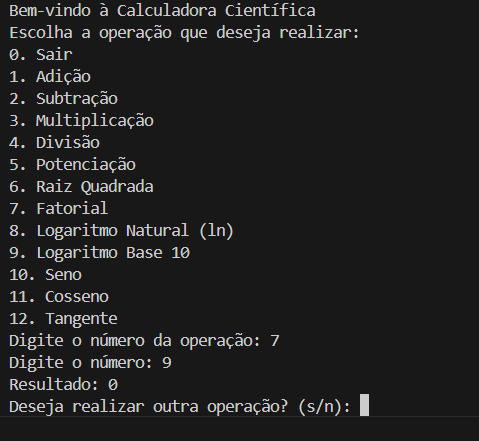
**Bug 1: Soma errada na função adição**

* **Identificação do Bug**
  + **Título do Bug:**Soma errada na função adição
  + **Localização do Bug:**Calculadora.py , função adição() , linha3
* **Descrição Detalhada**
  + **Descrição Clara do Bug:**Soma o primeiro valor de entrada (x) 2 vezes, o que resultará em um valor incorreto na soma de x+y
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado:**
    - Iniciar código calculadora
    - Escolher operação de soma (comando 1)
    - Digitar o primeiro valor (exemplo: 1 )
    - Digitar segundo valor (exemplo: 2)
  + **Consistência:**O Bugg acontece sempre que a função de soma é chamada, independente dos valores de entradas**.** O primeiro valor inserido sempre é chamado 2 vezes.
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado:** A soma dos dois valores inseridos (exemplo: 3)**.**
  + **Resultado Obtido:** O primeiro valor vezes dois mais o segundo valor (exemplo: 4)
* **Evidências**
  + ***Screenshots* e Vídeos:**
  + ****
* **Severidade e Impacto**
  + **Severidade do Bug:**Alto: Uma das funções de operações mais básicas não funciona corretamente para quaisquer valores de entrada**.**
  + **Impacto do Bug:**Resultados incorretos para somas simples. Não gera problemas caso a operação de soma não seja realizada.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções:** Alterar função para que soma do primeiro valor ocorra apenas uma vez
* **Registro e Acompanhamento**
  + **Data de Identificação:** 16/11/2024**.**
  + **Status do Bug:** resolvido

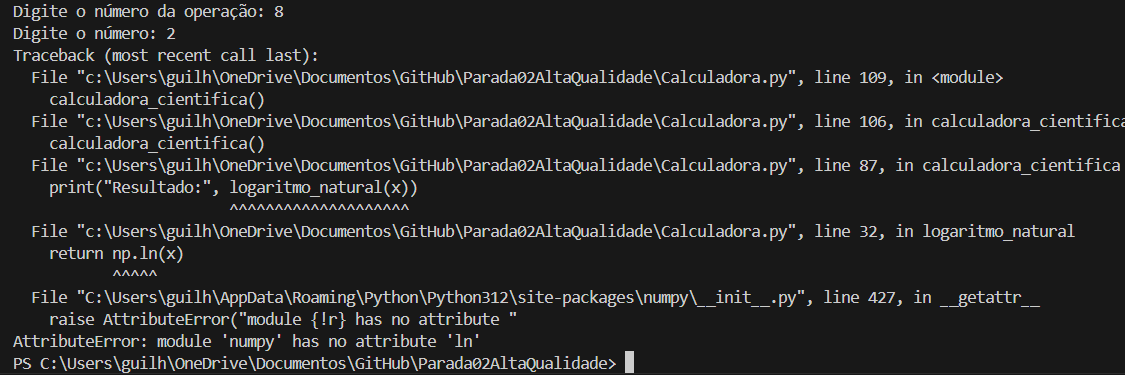
**Bug 2:** **Permitindo divisão por 0**

* **Identificação do Bug**
  + **Título do Bug:**Permitindo divisão por 0
  + **Localização do Bug:**Calculadora.py , função divisao() , linha12
* **Descrição Detalhada**
  + **Descrição Clara do Bug:**Caso o denominador seja 0, operação apresenta erro de cálculo
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado:**
    - Iniciar código calculadora
    - Escolher operação de soma (comando 4)
    - Digitar o primeiro valor (exemplo: 10 )
    - Digitar segundo valor (exemplo: 0)
  + **Consistência:**O Bugg acontece quando o segundo valor de entrada na função de divisão é 0. Função não aparenta apresentar erro com demais valores
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado:** Mensagem de erro: “Valor indefinido”**.**
  + **Resultado Obtido:** Erro ao calcular retorno
* **Evidências**
  + ***Screenshots* e Vídeos:**
  + ****
* **Severidade e Impacto**
  + **Severidade do Bug:**Baixo.
  + **Impacto do Bug:**Resultado errado para divisões com zero (deveria apresentar mensagem de erro mais específica). Bug não impacta demais valores na própria função, nem no software em geral.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções:** Colocar uma condicional na função, que trate valores divididos por zero
* **Registro e Acompanhamento**
  + **Data de Identificação:** 16/11/2024**.**
  + **Status do Bug:** resolvido

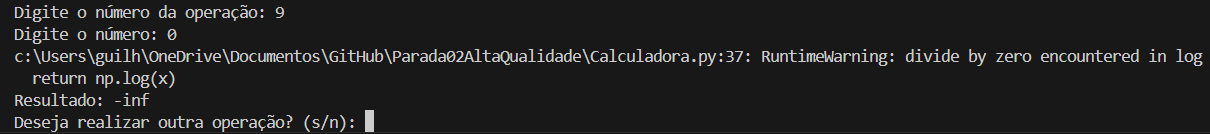
**Bug 3:** **Erro ao calcular fatorial**

* **Identificação do Bug**
  + **Título do Bug:**Erro ao calcular fatorial
  + **Localização do Bug:**Calculadora.py , função fatorial(), linha23
* **Descrição Detalhada**
  + **Descrição Clara do Bug:**Valor da operação fatorial apresenta sempre valor zero como resultado, independente da entrada. Aparentemente o valor zero está sendo incluído na multiplicação, zerando o valor de retorno.
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado:**
    - Iniciar código calculadora
    - Escolher operação de soma (comando 7)
    - Digitar o primeiro valor (exemplo: 9)
  + **Consistência:**O Bugg acontece sempre que a operação de fatorial é chamada, independente do valor de entrada
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado:** 362.880
  + **Resultado Obtido:** 0
* **Evidências**
  + ***Screenshots* e Vídeos:**
  + ****
* **Severidade e Impacto**
  + **Severidade do Bug:**Médio.
  + **Impacto do Bug:**Uma das operações da calculadora não funciona para nenhum valor inserido.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções:** Corrigir loop de repetição, iniciando-o a partir de 1
* **Registro e Acompanhamento**
  + **Data de Identificação:** 16/11/2024**.**
  + **Status do Bug:** pendente

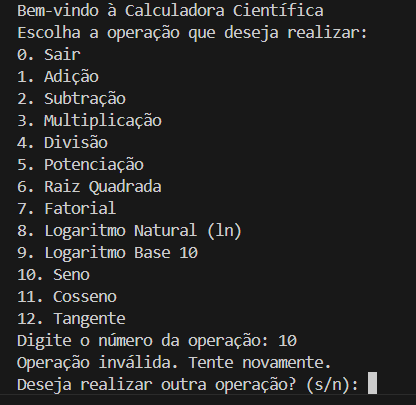
**Bug 4: Erro ao calcular o logaritmo natural**

* **Identificação do Bug**
  + **Título do Bug**: Erro ao calcular o logaritmo natural
  + **Localização do Bug**: Calculadora.py, função logaritmo\_natural(), linha 31
* **Descrição Detalhada**
  + **Descrição Clara do Bug**: A função utiliza o método np.ln, que não existe no módulo NumPy, resultando em erro sempre que a operação é chamada.
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado**:
    1. Iniciar o código da calculadora.
    2. Escolher operação de logaritmo natural (comando 8).
    3. Digitar um número positivo (exemplo: 2).
  + **Consistência**: O bug ocorre sempre que a função é chamada.
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado**: 0.693.
  + **Resultado Obtido**: module 'numpy' has no attribute 'ln'.
* **Evidências**
  + **Screenshots e Vídeos**:
  + 
* **Severidade e Impacto**
  + **Severidade do Bug**: Alto.
  + **Impacto do Bug**: A operação não funciona para nenhuma entrada válida.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções**: Substituir np.ln por np.log.
* **Registro e Acompanhamento**
  + **Data de Identificação**: 17/11/2024.
  + **Status do Bug**: Pendente

**Bug 5: Erro ao calcular o logaritmo base 10**

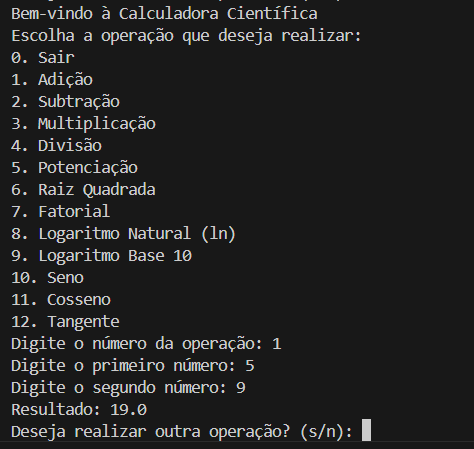
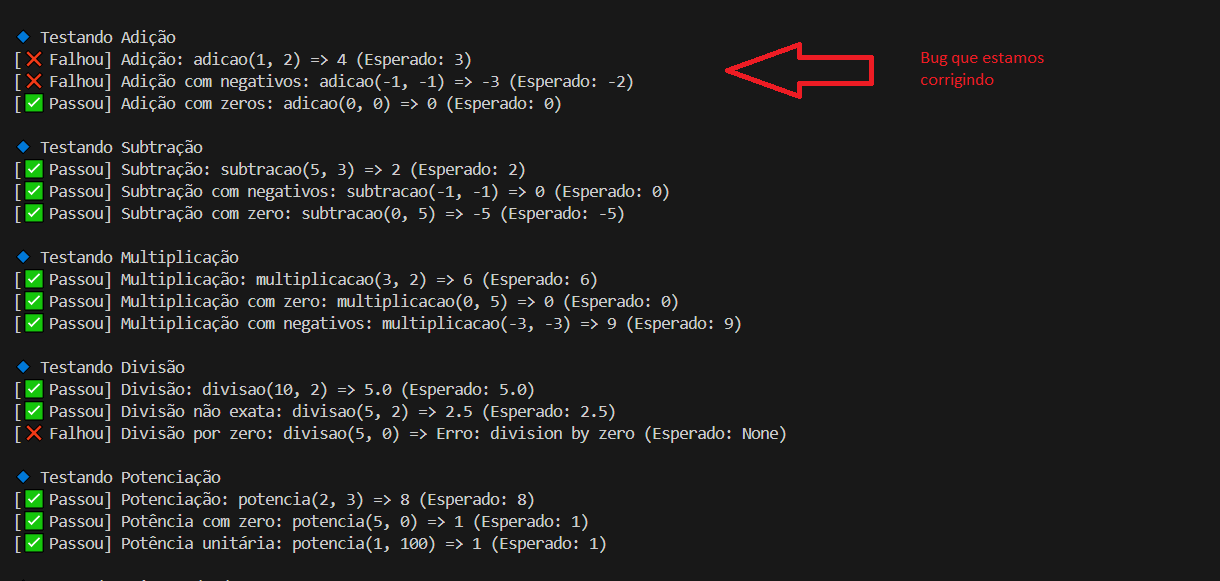
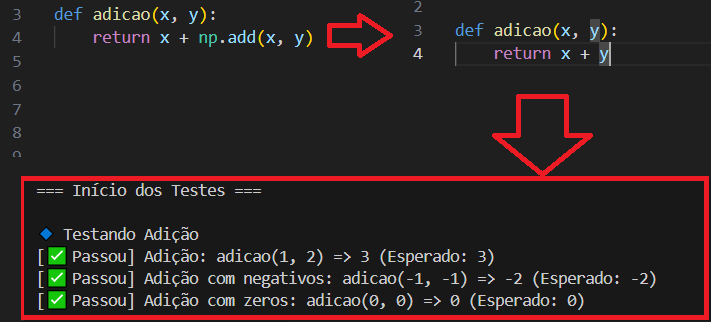
* **Identificação do Bug**
  + **Título do Bug**: Erro ao calcular o logaritmo base 10
  + **Localização do Bug**: Calculadora.py, função logaritmo\_base10(), linha 34
* **Descrição Detalhada**
  + **Descrição Clara do Bug**: A função não valida corretamente entradas iguais a zero.
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado**:
    1. Iniciar o código da calculadora.
    2. Escolher operação de logaritmo base 10 (comando 9).
    3. Digitar um número negativo (exemplo: 0).
  + **Consistência**: O bug ocorre sempre que a função é chamada com uma entrada menor ou igual a zero.
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado**: "Erro: Logaritmo de número não positivo".
  + **Resultado Obtido**: -inf.
* **Evidências**
  + **Screenshots e Vídeos**:
  + 
* **Severidade e Impacto**
  + **Severidade do Bug**: Médio.
  + **Impacto do Bug**: A operação não retorna resultados adequados para entradas inválidas.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções**: Adicionar validação para entrada zero.
* **Registro e Acompanhamento**
  + **Data de Identificação**: 17/11/2024.
  + **Status do Bug**: Pendente.

**Bug 6: Função de seno não implementada**

* **Identificação do Bug**
  + **Título do Bug**: Função de seno não implementada
  + **Localização do Bug**: Calculadora.py, listada no menu, mas ausente no código.
* **Descrição Detalhada**
  + **Descrição Clara do Bug**: O menu da calculadora lista a opção de calcular o seno (comando 10), mas a função correspondente não está implementada. Quando o usuário escolhe essa opção, ocorre um erro indicando que a função não existe.
* **Passos para Reproduzir**
  + **Passo a Passo Detalhado**:
    1. Iniciar o código da calculadora.
    2. Escolher a operação de seno (comando 10).
  + **Consistência**: O bug ocorre sempre que a operação de seno é chamada.
* **Resultado Esperado vs. Obtido**
  + **Resultado Esperado**: Retornar o valor do seno do ângulo fornecido, por exemplo, 0.5 para 30 graus.
  + **Resultado Obtido**: Operação inválida.
* **Evidências**
  + **Screenshots e Vídeos**:
  + 
* **Severidade e Impacto**
  + **Severidade do Bug**: Médio.
  + **Impacto do Bug**: A operação está presente no menu, mas não pode ser utilizada.
* **Sugestões para Correção (Opcional)**
  + **Possíveis Soluções**: Implementar a função seno utilizando numpy.sin, convertendo o ângulo de graus para radianos:
* **Registro e Acompanhamento**
  + **Data de Identificação**: 17/11/2024.
  + **Status do Bug**: Pendente.

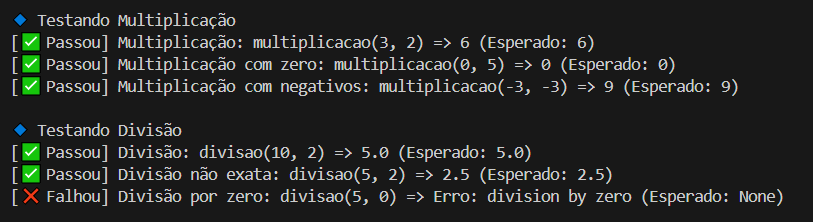
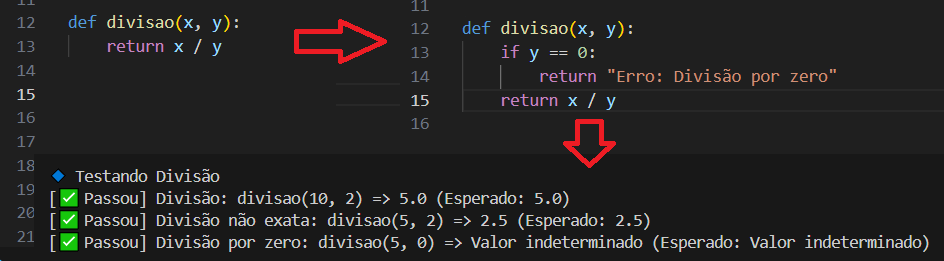
**Metodologia TDD:**

**Bug 1: Soma errada na função adição**

* **Introdução:** O relatório de bugs foi revisado e o bug reproduzido
  + 
* **Criação de Função Teste:** Criei uma função que realiza testes das operações, comparando valores esperados com obtidos. Para a mensagem de erro ficar mais detalhada, criei duas funções. Uma test\_operacoes() , que realiza testes de todas as operações e pode ser aprimorada caso novos testes ou novas opções precisem ser implementadas. Essa função, em cada resultado, chama a função testar\_operacao() , que formata a mensagem do resultado do teste num padrão mais organizado.
  + 
* **Código corrigido:** Função de soma foi corrigida e novo teste feito
  + ****
* **Documentação de correção:** A função de soma foi corrigida, substituindo a “x + np.add(x,y)” para “x + y”.
* **Status alterado:** Alterei o status do bug para “resolvido” no relatório

**Bug 2: Permitindo divisão por 0**

* **Introdução:** O relatório de bugs foi revisado e o bug reproduzido
  + **Texto

    Descrição gerada automaticamente**
* **Criação de Função Teste:** Rodei a função de teste, descrita no relatório anterior.
  + 
* **Código corrigido:** Função de divisão foi corrigida e novo teste feito
  + ****
* **Documentação de correção:** A função de divisao foi corrigida, adicionando uma condicional que retorna “valor indeterminado” caso y seja 0.
* **Status alterado:** Alterei o status do bug para “resolvido” no relatório

**Observações finais:** Olá professor, sei que a atividade pedia para escolhermos um dos erros, mas fiz com dois para demonstrar que o processo pode ser seguido sequencialmente, com os testes sendo sempre realizados para todas as operações, garantindo que uma correção não interfira em outra já realizada. Como essa calculadora é simples, isso não aconteceria, pois cada comando usa uma unção isolada, mas em um software mais complexo isso poderia ocorrer.