Universidade de Brasília — Faculdade UnB Gama

FGA0314 – TESTES DE SOFTWARE (2025.2)

Atividade 4 – TDD

Validador de Expressões Matemáticas

Turma	01	Semestre	2025.2

Equipe	Ferrugem

Nome	**Matrícula**
Davi Marques do Egito Coelho	231030421
Luiz Gustavo Silva de Almeida	221022669
Vitor Guilherme Lustosa de Carvalho	232014342
Artur Handow Krauspenhar	231034082
Lucas de Oliveira Rodrigues	202017684
Mauricio Ferreira de Araujo	222007021

1. Linguagem e framework de teste unitário utilizados.

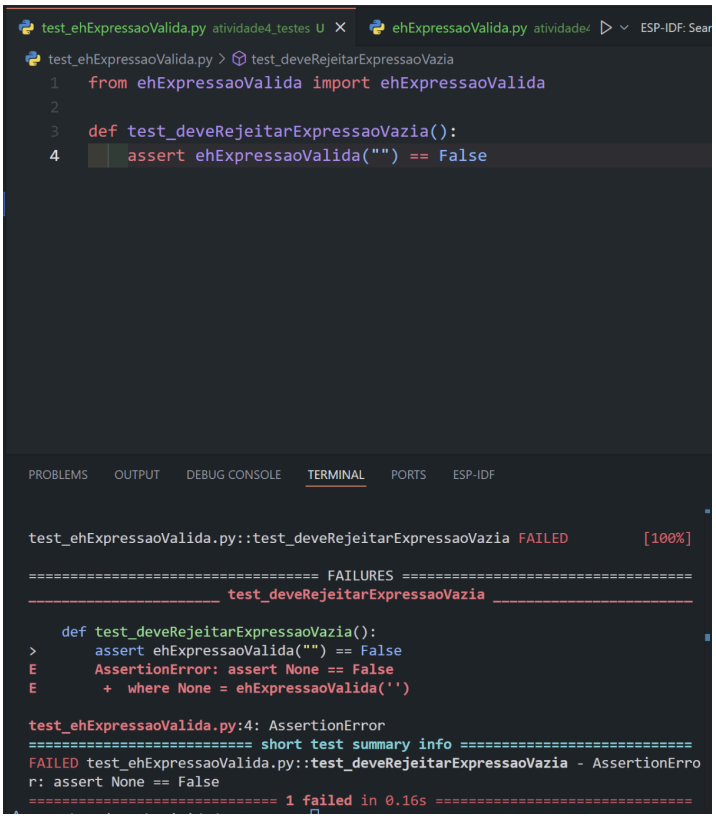
A linguagem de programação escolhida para a implementação do TDD validador de expressões matemáticas foi o Python. O framework de teste unitário utilizado foi o **(pytest ou unittest)**.

2. Ciclos de TDD (Red-Green-Refactor).

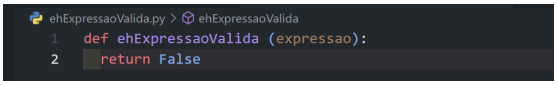
Ciclo 1: O caso mais básico (Inválido).

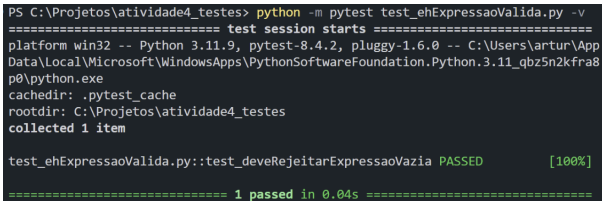
- **Teste:** `deveRejeitarExpressaoVazia()`

- RED: foco em lidar com a string vazia. Resultado: teste falhado.



- GREEN: função mínima para passar. Resultado: teste bem-sucedido.



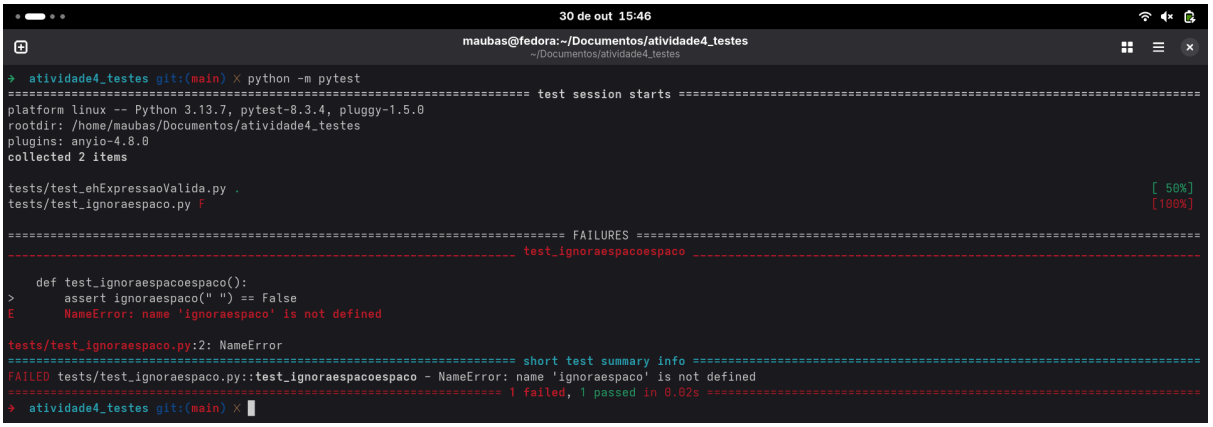


Ciclo 2: Regra 6 (Ignorar Espaços).

- **Teste:** `deveRejeitarExpressaoApenasComEspacos()`
- RED: sem implementação, teste falhando.

File: test_ignoraespaco.py

```
1  def test_ignoraespacoespaco():
2      pass
```



- GREEN: função mínima para passar, teste bem-sucedido.

File: ehExpressaoValida.py

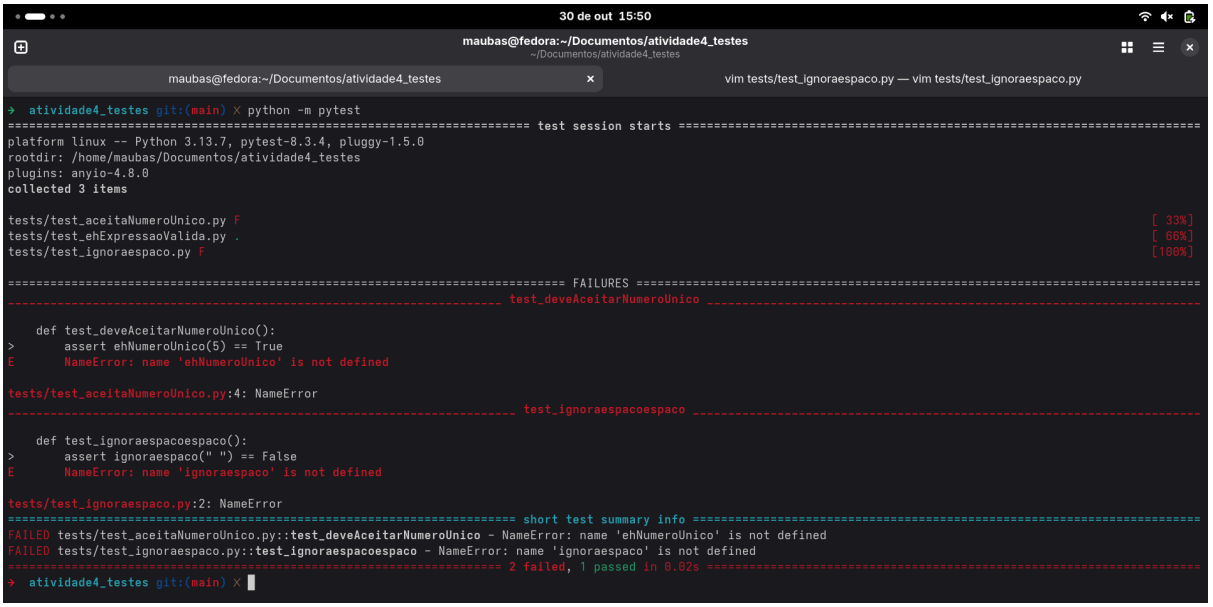
```
1  def ehExpressaoValida(expressao):
2      # remoção de espaços da expressão
3      expr_sem_espaco = expressao.replace(" ", "")
4
5      # se após remover os espaços a string estiver vazia, tem que retornar fa
   lso
6      if expr_sem_espaco == "":
7          return False
8
9      # retorne falso pra qualquer outra expressão
10     return False
```

```
→  atividade4_testes git:(main) ✗ python -m pytest tests/test_ehExpressaoValida.py::t
est_deveRejeitarExpressaoApenasComEspacos -v
================================= test session starts ==================================
platform linux -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- /usr/bin/python
cachedir: .pytest_cache
rootdir: /home/archsphinx/git/atividade4_testes
collected 1 item

tests/test_ehExpressaoValida.py::test_deveRejeitarExpressaoApenasComEspacos PASSED [1
00%]

================================== 1 passed in 0.01s ===================================
```

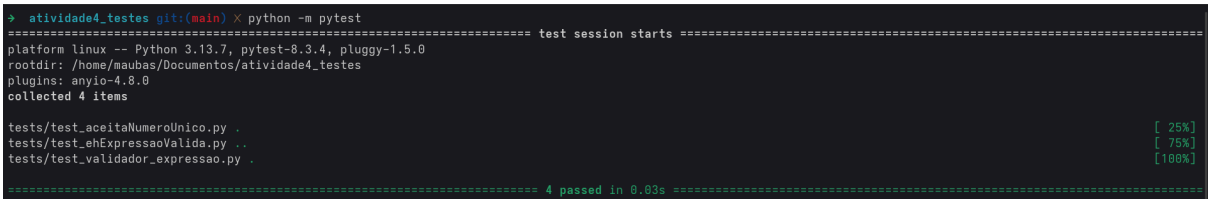
Ciclo 3: O caso válido mais simples.

- **Teste:** `deveAceitarNumeroUnico()`
- RED: sem implementação, teste falhando.

```python
from src.ehExpressaoValida import ehExpressaoValida

def test_deveAceitarNumeroUnico():
    assert ehExpressaoValida(5) == True
```



- GREEN: teste passando após a função implementada.



- BLUE: refatoração.

```python
def ehExpressaoValida(expressao):
    if isinstance(expressao, int):
        return True


    # remoção de espaços da expressão
    expr_sem_espaco = expressao.replace(" ", "")

    # se após remover os espaços a string estiver vazia, tem que retornar falso
    if expr_sem_espaco == "":
        return False
    # retorne falso pra qualquer outra expressão
    return False
```

Ciclo 4: Regra 3 (Começar/Terminar com Operador).

- **Teste:** `deveRejeitarSeComecarOuTerminarComOperador()`
- RED: código do teste implementado, com o teste falhado.

test_validador_expressao.py × validador_expressao.py

tests > test_validador_expressao.py

```python
import sys, os
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "src"))

from src.validador_expressao import eh_expressao_valida

def test_deveRejeitarSeComecarOuTerminarComOperador():
    casos_invalidos = ["+1", "1*", "/"]
    for expr in casos_invalidos:
        assert eh_expressao_valida(expr) is False

```

```
C:\Users\lalme\OneDrive\Desktop\AAE-4\atividade4_testes>python -m pytest tests/test_validador_expressao.py
========================== test session starts ==========================
platform win32 -- Python 3.12.0, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\lalme\OneDrive\Desktop\AAE-4\atividade4_testes
plugins: anyio-4.11.0, socket-0.7.0, xdist-3.8.0
collected 1 item

tests\test_validador_expressao.py F                                    [100%]

=================================== FAILURES ===================================
_____________ test_deveRejeitarSeComecarOuTerminarComOperador _____________

    def test_deveRejeitarSeComecarOuTerminarComOperador():
        casos_invalidos = ["+1", "1*", "/"]
        for expr in casos_invalidos:
>           assert eh_expressao_valida(expr) is False
E           AssertionError: assert True is False
E            +  where True = eh_expressao_valida('+1')

tests\test_validador_expressao.py:9: AssertionError
======================== short test summary info ========================
FAILED tests/test_validador_expressao.py::test_deveRejeitarSeComecarOuTerminarComOperador - AssertionError: assert True is False
========================== 1 failed in 0.19s ===========================
```

- GREEN: código da funcionalidade implementada no ciclo, com o teste bem-sucedido.

test_validador_expressao.py 1, U | validador_expressao.py U

src > validador_expressao.py > eh_expressao_valida

```python
def eh_expressao_valida(expressao: str) -> bool:

    expr = expressao.replace(" ", "")
    if not expr:
        return False

    operadores = {"+", "-", "*", "/"}
    if expr[0] in operadores or expr[-1] in operadores:
        return False

    return True
```

```
C:\Users\lalme\OneDrive\Desktop\AAE-4\atividade4_testes>python -m pytest tests/test_validador_expressao.py
============================= test session starts =============================
platform win32 -- Python 3.12.0, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\lalme\OneDrive\Desktop\AAE-4\atividade4_testes
plugins: anyio-4.11.0, socket-0.7.0, xdist-3.8.0
collected 1 item

tests\test_validador_expressao.py .                                      [100%]

============================== 1 passed in 0.04s ==============================
```

Ciclo 5: Regra 2 (Operadores Seguidos).

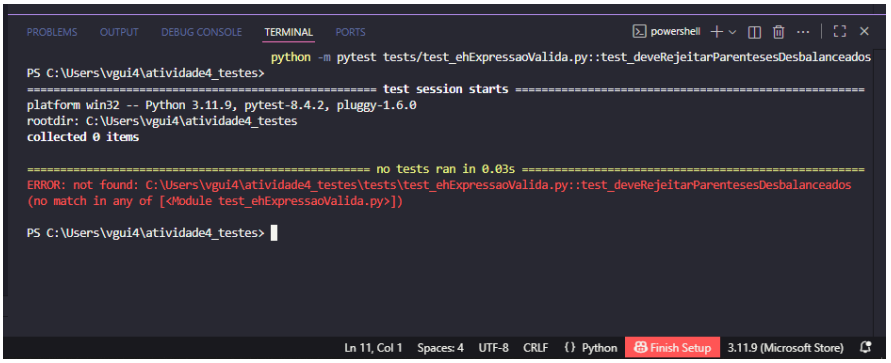
- **Teste:** deveRejeitarOperadoresDuplos()
- **RED:** código do teste implementado, com o teste falhando.

```
PS C:\Users\lukin\Downloads\att4_testes> C:/Users/lukin/Downloads/att4_testes/.venv/Scripts/python.exe -m pytest tests/test_validador_expressao.py -v
============================= test session starts =============================
platform win32 -- Python 3.13.0, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\lukin\Downloads\att4_testes\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lukin\Downloads\att4_testes
collected 1 item

tests/test_validador_expressao.py::test_deveRejeitarOperadoresDuplos FAILED [100%]

================================== FAILURES ===================================
_____________________ test_deveRejeitarOperadoresDuplos _____________________

    def test_deveRejeitarOperadoresDuplos():
        # Test case 1: Adjacent plus operators
>       assert not validador_expressao("1++2"), "Should reject expression with consecutive plus operators"
E       AssertionError: Should reject expression with consecutive plus operators
E       assert not True
E        +  where True = validador_expressao('1++2')

tests\test_validador_expressao.py:6: AssertionError
=========================== short test summary info ===========================
FAILED tests/test_validador_expressao.py::test_deveRejeitarOperadoresDuplos - AssertionError: Should reject expression with consecutive plus operators
============================== 1 failed in 0.25s ==============================
PS C:\Users\lukin\Downloads\att4_testes>
```

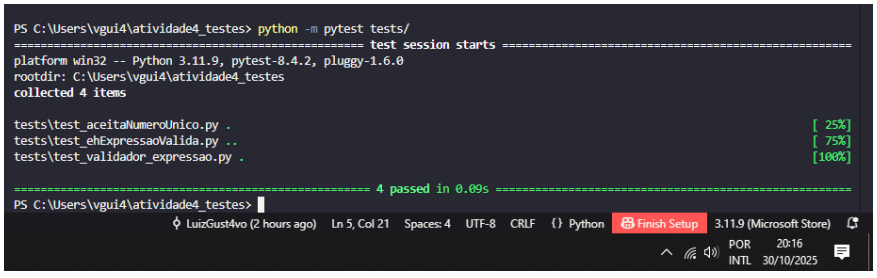
```
=================================== FAILURES ===================================
_____ test_deveRejeitarOperadoresDuplos _____

    def test_deveRejeitarOperadoresDuplos():
        # Test case 1: Adjacent plus operators
        assert not validador_expressao("1++2"), "Should reject expression with consecutive plus operators"

        # Test case 2: Adjacent division and multiplication operators
>       assert not validador_expressao("5 / * 3"), "Should reject expression with consecutive division and multiplication operators"
E       AssertionError: Should reject expression with consecutive division and multiplication operators
E       assert not True
E        +  where True = validador_expressao('5 / * 3')

tests\test_validador_expressao.py:9: AssertionError
=========================== short test summary info ============================
FAILED tests/test_validador_expressao.py::test_deveRejeitarOperadoresDuplos - AssertionError: Should reject expression with consecutive division and multiplication operators
```

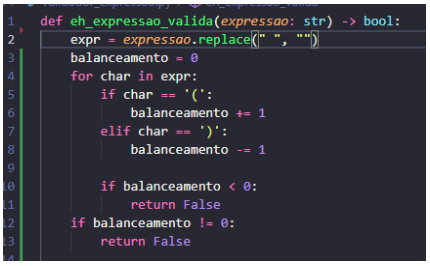
Ciclo 6: Regra 1 (Parênteses Balanceados).

- **Teste:** `deveRejeitarParentesesDesbalanceados()`
- RED: código do teste implementado, com o teste falhado.



- GREEN: código da funcionalidade implementada no ciclo, com o teste bem-sucedido.





```python
def eh_expressao_valida(expressao: str) -> bool:
    expr = expressao.replace(" ", "")
    balanceamento = 0
    for char in expr:
        if char == '(':
            balanceamento += 1
        elif char == ')':
            balanceamento -= 1

        if balanceamento < 0:
            return False
    if balanceamento != 0:
        return False
```

- BLUE/Refactor: código de produção e arquivo de testes refatorados.

validador_expressao.py

src > validador_expressao.py > validador_expressao

```python
def validador_expressao(expressao):

    # ciclo 1 - tipo inválido
    if expressao is None:
        return False

    # converte tudo pra string (caso número passe direto)
    if isinstance(expressao, int):
        return True

    # ciclo 2 - regra 6
    expr = expressao.replace(" ", "")
    if not expr:
        return False

    # ciclo 3 – número único válido
    if expr.isdigit():
        return True

    # ciclo 4 – regra 3
    operadores = {"+", "-", "*", "/"}
    if expr[0] in operadores or expr[-1] in operadores:
        return False

    # ciclo 6 - regra 1
    balanceamento = 0
    for char in expr:
        if char == '(':
            balanceamento += 1
        elif char == ')':
            balanceamento -= 1

        if balanceamento < 0:
            return False
    if balanceamento != 0:
        return False

    # demais casos ainda não implementados (ciclos 5+)
    return False
```

- https://github.com/daviegito/atividade4_testes/blob/main/src/validador_expressao.py

test_validador_expressao.py

tests > test_validador_expressao.py > ...

```python
from src.validador_expressao import validador_expressao

# ciclo 1
def test_deveRejeitarExpressaoVazia():
    assert validador_expressao("") is False

# ciclo 2
def test_deveRejeitarExpressaoApenasComEspacos():
    assert validador_expressao("") is False

# ciclo 3
def test_deveAceitarNumeroUnico():
    assert validador_expressao(5) is True

# ciclo 4
def test_deveRejeitarSeComecarOuTerminarComOperador():
    casos_invalidos = ["+1", "1*", "/"]
    for expr in casos_invalidos:
        assert validador_expressao(expr) is False

# ciclo 6
def test_deveRejeitarParentesesDesbalanceados():
    casos_invalidos = ["(1+2", "1+2)", ")("]
    for expr in casos_invalidos:
        assert validador_expressao(expr) is False

```

- https://github.com/LuizGust4vo/saleor_testes/blob/main/saleor/graphql/account/tests/mutations/authentication/test_token_create.py

3. **Reflexão sobre a prática realizada (o que funcionou bem? o que foi mais difícil?)**

Durante a prática de TDD, uma das maiores dificuldades que o grupo enfrentou foi a consecução dos diferentes ciclos, tendo cada um tendo de começar imediatamente após o término do outro para que o código gradualmente fosse incrementado. Isso colocou a comunicação do grupo à prova, uma vez que usamos um repositório para fazer essa atividade e tivemos de ter cuidado com as diferentes versões e commits feitos pelo outro para garantir que o código fosse gradualmente crescendo da versão inicial.

Tivemos algumas pequenas dificuldades iniciais com a importação de módulos em Python, sobre como seria feita a refatoração e a hierarquia de pastas que tivemos, mas eventualmente começamos a sanar esses problemas. Um dos nossos membros teve problemas com a questão da biblioteca Pytest, mas que eventualmente foi sanado também.