Universidade Federal de Sergipe - UFS
Centro de Ciências Exatas e Tecnologia - CCET
Departamento de Computação - DCOMP

Bruno Macedo da Silva

Teste de Software

Atividade 1

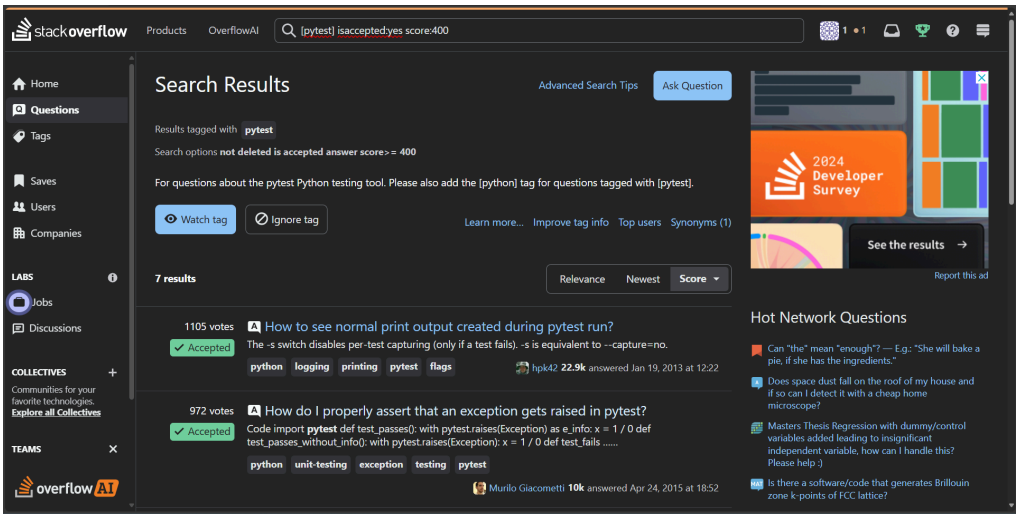
São Cristóvão - SE
2024

Localizar pergunta no StackOverflow

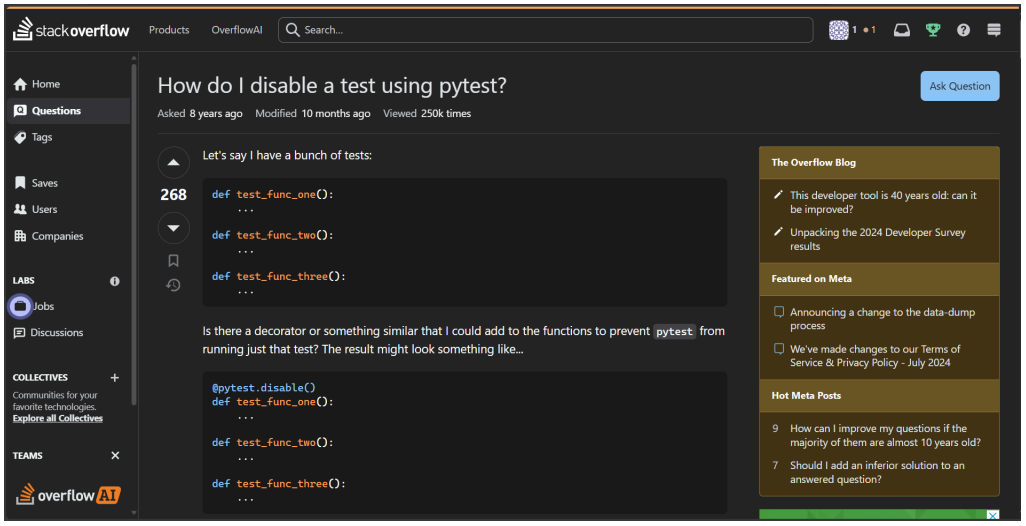
Para a atividade proposta, foi necessário localizar uma pergunta no site StackOverflow. As condições propostas pela atividade foram:

1. Problema relacionado a testes de unidade;
2. String de busca conter “[unit-testing] or [junit] or [pytest]”;
3. Ordenar as perguntas pela pontuação (Score);
4. A pergunta deve ter sido solucionada;
5. Ter no mínimo 400 votos

Para atender as condições 1, 2, 4 e 5, a string de busca utilizada foi a seguinte: “*[pytest] isaccepted:yes score:400*” . A pesquisa retornou 7 resultados, então para atender a condição número 3, o resultado foi ordenado pelo *Score* conforme imagem abaixo:

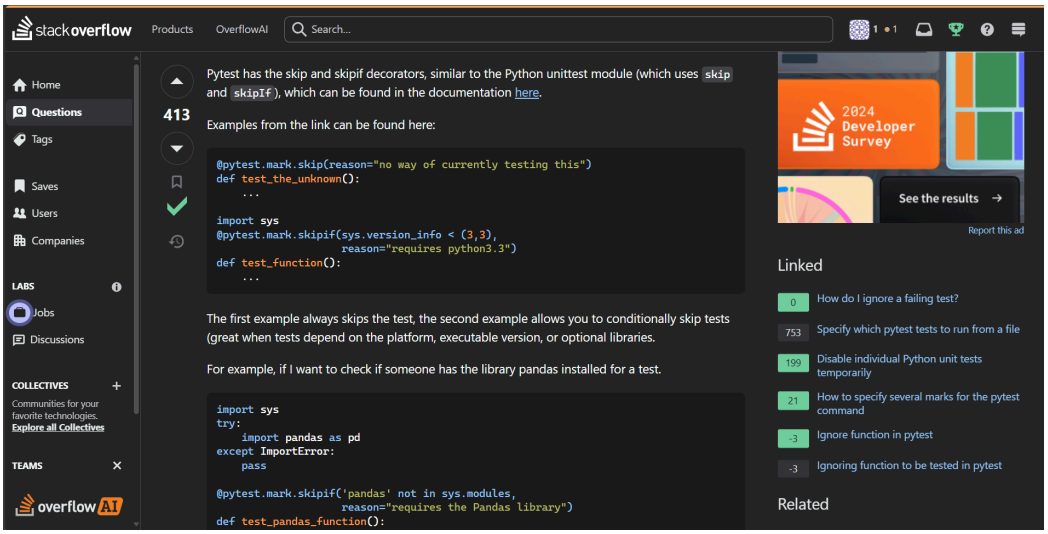


Um dos resultados foi uma pergunta com o seguinte título: *“How do I disable a test using pytest?”* disponível no endereço python - How do I disable a test using pytest? - Stack Overflow.



Essa pergunta foi escolhida pois pode ser importante para o processo de construção de testes automatizados.

Na pergunta, o usuário questiona se há uma forma de prevenir que o teste seja realizado por algum motivo. A resposta aceita e com mais votos positivos (um maior *Score*) está a o uso do decorator *@pytest.mark.skip()* e *@pytest.mark.splitif()*:



Essa condição criada pode ser utilizada em casos, por exemplo, para que não apresente erros em testes no qual o método ainda não foi implementado, ou utilize uma versão específica do python ou necessite de uma biblioteca que, por algum motivo, é interessante prevenir o erro caso ela não consiga ser importada.

Demonstração

Os arquivos estão disponíveis no repositório git no seguinte endereço: https://github.com/macedobruno/Teste_Software_2024_Silva_Bruno.git. Para continuar com o processo, é preciso que o sistema tenha o *python* e *git* instalado. Os passos a seguir são necessários para utilizar os arquivos criados:

1. **Clonar repositório:**

Para clonar o repositório git, é preciso executar os seguintes comandos no prompt:

git clone https://github.com/macedobruno/Teste_Software_2024_Silva_Bruno.git
cd Teste_Software_2024_Silva_Bruno

2. **Instalar dependências**

As dependências necessárias estão listadas no arquivo *requirements.txt*. Para instalar, basta executar o comando:

pip install -r requirements.txt

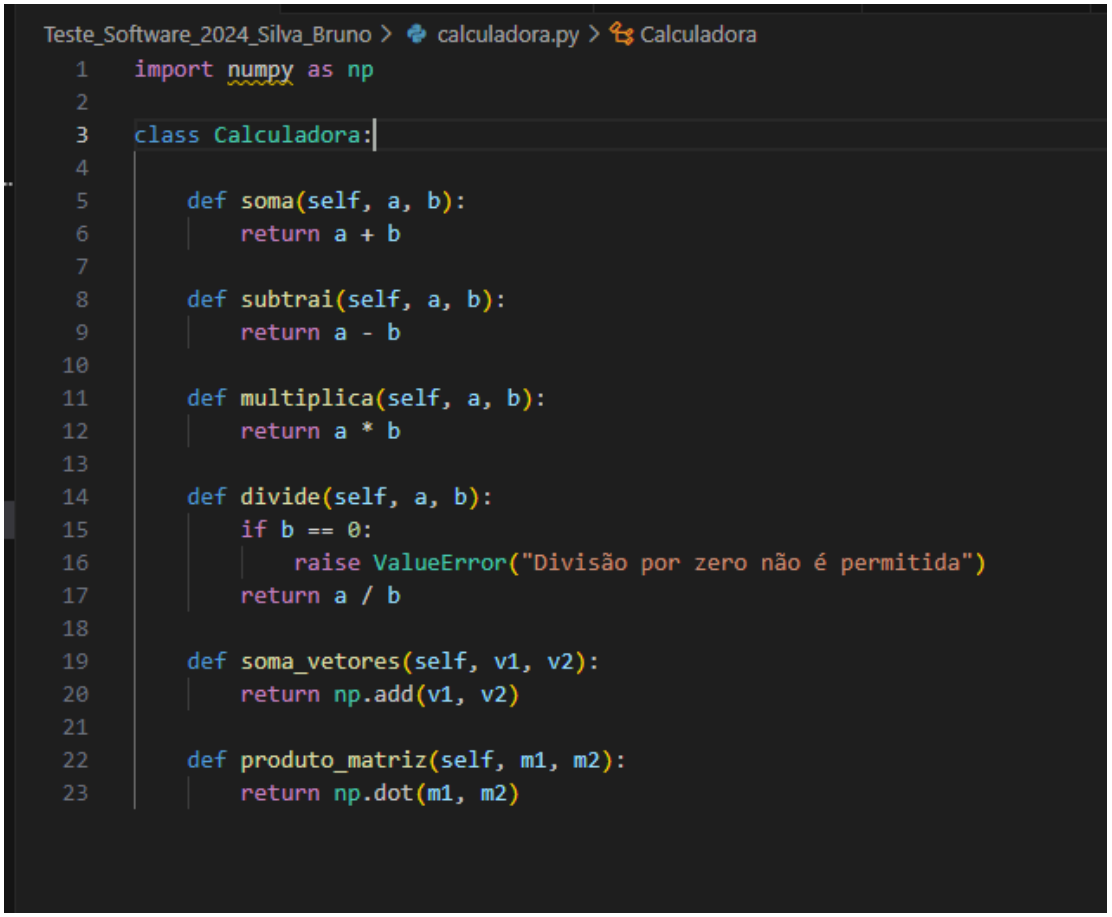
3. **Uso**

Para executar os testes criados, entre com o seguinte comando no prompt:

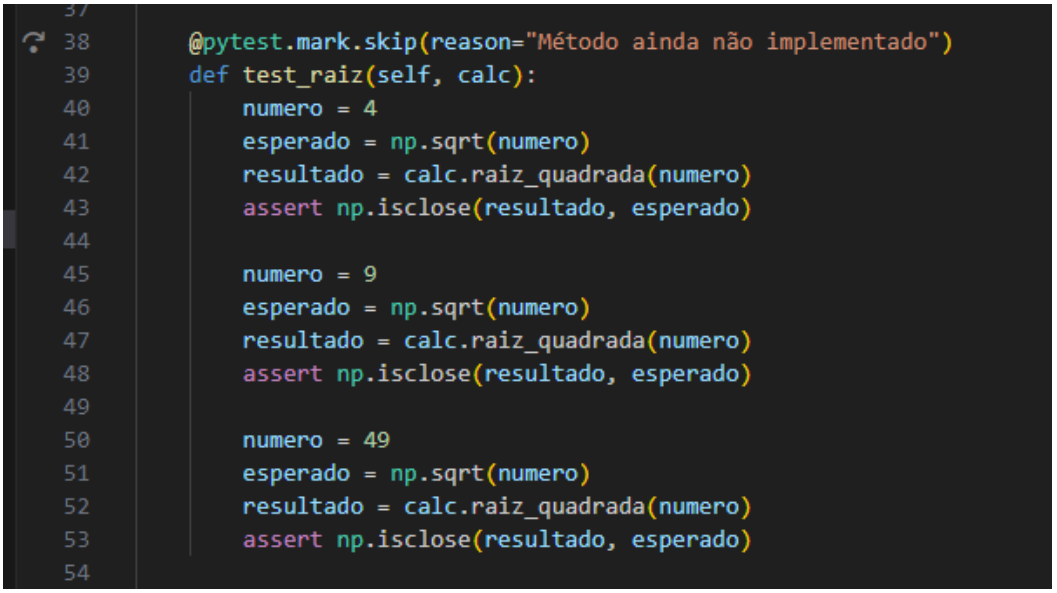
pytest test_calculadora.py

Aplicar solução

Para demonstrar a aplicação da solução foi criado uma classe chamada *calculadora.py* com funções básicas de uma calculadora.



Aplicando a primeira solução, foi implementada uma classe de teste chamada *test_calculadora.py*. Foi adicionado um teste para o método raiz_quadrada, no qual ainda não foi implementado na classe *calculadora.py*.



Assim, é possível evitar um falso negativo quanto ao resultado dos testes do método em questão.

Para demonstrar a utilidade do *@pytest.mark.skipif()*, foram criados dois métodos de teste: o *test_soma_vetores* e o *test_produto_matriz*; sendo o primeiro com a condição de pular o teste caso a biblioteca *numpy* não esteja entre os módulos instalados, e a segunda da mesma forma, mas usando a biblioteca *pandas*:

```python
@pytest.mark.skipif('numpy' not in sys.modules, reason="Biblioteca numpy é necessária para o teste.")
def test_soma_vetores(self, calc):
    vetor1 = np.array([1, 2, 3])
    vetor2 = np.array([4, 5, 6])
    resultado = calc.soma_vetores(vetor1, vetor2)
    esperado = np.add(vetor1, vetor2)
    assert all([r1 == r2 for r1, r2 in zip(resultado, esperado)])

    vetor1 = np.array([8, 5, 0])
    vetor2 = np.array([2, 2, 9])
    resultado = calc.soma_vetores(vetor1, vetor2)
    esperado = np.add(vetor1, vetor2)
    assert all([r1 == r2 for r1, r2 in zip(resultado, esperado)])

    vetor1 = np.array([3.1, 0.24, 1.1])
    vetor2 = np.array([2.9, 5.3, 9.22])
    resultado = calc.soma_vetores(vetor1, vetor2)
    esperado = np.add(vetor1, vetor2)
    assert all([r1 == r2 for r1, r2 in zip(resultado, esperado)])
```

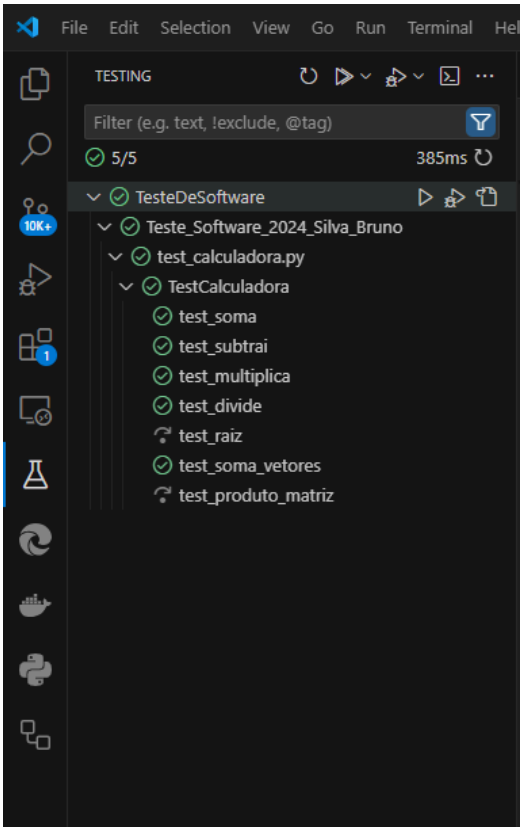
```python
@pytest.mark.skipif('pandas' not in sys.modules, reason="Biblioteca pandas não é necessária, mas isso é um teste")
def test_produto_matriz(self, calc):
    matriz1 = np.array([[1, 2], [3, 4]])
    matriz2 = np.array([[5, 6], [7, 8]])
    resultado = calc.produto_matriz(matriz1, matriz2)
    esperado = np.array([[19, 22], [43, 50]])
    assert all([m1 == m2 for m1, m2 in zip(resultado, esperado)])

    # Produto de uma matriz com uma matriz identidade
    matriz = np.array([[1, 2], [3, 4]])
    identidade = np.eye(2)
    resultado = calc.produto_matriz(matriz, identidade)
    esperado = matriz
    assert all([m1 == m2 for m1, m2 in zip(resultado, esperado)])

    # Produto de uma matriz com uma matriz de zeros
    zeros = np.zeros((2, 2))
    resultado = calc.produto_matriz(matriz, zeros)
    esperado = zeros
    assert all([m1 == m2 for m1, m2 in zip(resultado, esperado)])

    # Produto de duas matrizes não quadradas
    matriz1 = np.array([[1, 2, 3], [4, 5, 6]])
    matriz2 = np.array([[7, 8], [9, 10], [11, 12]])
    resultado = calc.produto_matriz(matriz1, matriz2)
    esperado = np.array([[58, 64], [139, 154]])
    assert all([m1 == m2 for m1, m2 in zip(resultado, esperado)])
```

Observando as bibliotecas no arquivo *requirements.txt*, estão o *pytest*, necessário para a implementação dos testes unitários, e o *numpy*, que auxilia em algumas das operações feitas. Então, o teste implementado, sempre será pulado, pois a biblioteca *pandas* não está instalada. Seguem resultados mostrados pela IDE Visual Studio Code:


File Edit Selection View Go Run Terminal
TESTING
Filter (e.g. text, !exclude, @tag)
5/5
385ms
TesteDeSoftware
Teste_Software_2024_Silva_Bruno
test_calculadora.py
TestCalculadora
test_soma
test_subtrai
test_multiplica
test_divide
test_raiz
test_soma_vetores
test_produto_matriz


PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
tware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_subtrai', '/home/bruno/UFS/2024.1/TesteDeSoftware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_multiplica', '/home/bruno/UFS/2024.1/TesteDeSoftware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_divide', '/home/bruno/UFS/2024.1/TesteDeSoftware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_raiz', '/home/bruno/UFS/2024.1/TesteDeSoftware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_soma_vetores', '/home/bruno/UFS/2024.1/TesteDeSoftware/Teste_Software_2024_Silva_Bruno/test_calculadora.py::TestCalculadora::test_produto_matriz']
========== test session starts ==========
platform linux -- Python 3.10.12, pytest-8.3.2, pluggy-1.5.0
rootdir: /home/bruno/UFS/2024.1/TesteDeSoftware
collected 7 items
Teste_Software_2024_Silva_Bruno/test_calculadora.py ....s.s [100%]
========== 5 passed, 2 skipped in 0.11s ==========
Finished running tests!
Test run at 8/6/2024, 10:38:13 PM
test_soma
test_subtrai
test_multiplica
test_divide
test_soma_vetores
test_raiz
test_produto_matriz
Test run at 8/6/2024, 10:31:08 PM
Test run at 8/6/2024, 10:30:30 PM
Test run at 8/6/2024, 10:29:43 PM
Test run at 8/6/2024, 10:27:40 PM
Test run at 8/6/2024, 10:27:14 PM
Ln 83, Col 11 Spaces: 4 UTF-8 LF Python 3.10.12 ('Teste_Software_2024_Silva_Bruno': venv)