



Executar Teste e Implantação de Aplicativos Computacionais

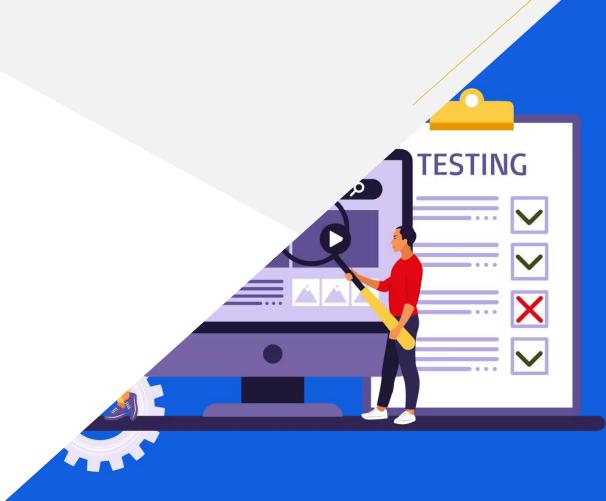
SENAC PE

25 de Outubro de 2024



Teste de Mutação

Automação com MutPy





Teste de Mutação

Conceito

• O teste de mutação é um **critério baseado defeitos** ou seja, utiliza conhecimento sobre defeitos típicos que podem ocorrer ao escrevermos programa para nos ajudar a criar bons conjuntos de testes.



Teste de Mutação

MutPy

• <u>MutPy</u> é uma ferramenta de teste de mutação para código-fonte Python 3.3+. MutPy suporta módulo unittest padrão, gera relatórios YAML/HTML e tem saída colorida. Ele aplica mutação no nível AST. Você pode impulsionar seu processo de teste de mutação com mutações de alta ordem (HOM) e análise de cobertura de código.



Exemplo • MutPy



Instalação

- Este módulo não vem embutido com Python. Para instalá-lo, digite o comando abaixo no terminal.
- · Caso utilize o Google Colab, execute o comando com uma! na frente do pip em um bloco de código.





Código Exemplo

usaremos a biblioteca MutPy para executar casos de teste escritos para um programa simples.
 Realizamos testes de mutação para um programa que verifica se um número é primo ou não.

Código: isPrime.py

```
def isPrime(num):
   if num > 1:
   for i in range(2,num):
        if (num % i) == 0:
            return False
   else:
        return True
   else:
   return False;
```



Casos de Teste

- Agora, precisamos escrever casos de teste para o programa acima usando a biblioteca Pytest. Os
 casos de teste devem ser escritos com uma abordagem para matar todos os mutantes, ou seja, os
 casos de teste devem ser eficazes o suficiente para dar uma boa pontuação de mutação.
- Os casos de teste são escritos usando assert que é uma asserção de teste geralmente usada para determinar se um caso de teste passou ou falhou. Escrevemos três funções de teste abaixo para verificar os três tipos de entrada:
 - 1. A entrada é primo
 - 2. A entrada não é primo
 - 3. A entrada é inválida



Código de Teste

- O nome da função e o nome do arquivo de teste devem sempre começar com a palavra 'test'.
- Código: test_isPrime.py

```
from isPrime import isPrime
def test_nonprime():
   assert isPrime(12) == False
def test_prime():
   assert isPrime(19) == True
def test_invalid():
   assert isPrime(-1) == False
```



Executando o teste

 Para executar esses casos de teste, precisamos criar dois arquivos separados isPrime.py e test_isPrime.py em uma única pasta e executar o seguinte comando no prompt de comando:

```
mut.py --target isPrime --unit-test test_isPrime -m --runner pytest
or
python <path\mut.py> --target isPrime --unit-test test_isPrime -m
```

- No comando acima, temos que especificar três coisas:
 - **Destino**: o arquivo de destino no qual os casos de teste serão executados, que no nosso caso é isPrime.py
 - Teste de unidade: o arquivo que contém os testes de unidade que devem ser executados, ou seja, test_isPrime.py no nosso caso.
 - runner: pytest ou unittest



Saida

A saída será um conjunto de mutantes junto com detalhes como pontuação de mutação, número de mutações eliminadas, sobrevividas, etc.

```
- test_isPrime [3.07469 s]
         if num > 1:
             for i in range(2, num):
                 if num % i == 0:
                     return False
             else:
[1.45151 s] killed by testing program mutpy/test_isPrime.py::CalculatorTest::test_nonprime
  2: def isPrime(num):
         if num > 1:
         if not (num > 1):
             for i in range(2, num):
                     return False
```



Saida

• Podemos ver na saída acima, 6 mutantes foram mortos e apenas 3 mutantes conseguiram sobreviver. Além disso, uma pontuação de mutação de 66,7% foi alcançada. Podemos melhorar ainda mais essa pontuação de mutação analisando os mutantes que sobreviveram aos casos de teste e escrevendo novos ou modificando casos de teste para matar os mutantes que sobreviveram.



Exercícios

MutPy



Exercício

- Verifique o programa smart_calc.py:
 - a) Crie um conjunto de testes (baseado em outros critérios estudados) com 8 testes
 - b) Teste o programa de acordo com o resumo da aula (utilizando o MutPy)
 - c) Documente os resultados (casos de teste)





Senac Pernambuco Educação Profissional Recife

Thiago Dias Nogueira

Instrutor Técnico

(81) 9 9627-0419

thiago.nogueira@pe.senac.br