



# Executar Teste e Implantação de Aplicativos Computacionais

SENAC PE

25 de Outubro de 2024

# Teste de Mutaç o

- Automa  o 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

- MutPy 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

# Exemplo

## 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.



```
pip install mutpy
```

# Exemplo

## 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

```
# define a function
def isPrime(num):

    if num > 1:

        # check for factors
        for i in range(2,num):
            if (num % i) == 0:
                return False

        else:
            return True

    # if input number is less than
    # or equal to 1, it is not prime
    else:
        return False;
```

# Exemplo

## 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



# Exemplo

## 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

```

# importar bibliotecas necessárias
from isPrime import isPrime

# caso de teste para verificar números não primos
def test_nonprime():
    assert isPrime(12) == False

# caso de teste para verificar números primos
def test_prime():
    assert isPrime(19) == True

# caso de teste para verificar entrada inválida
def test_invalid():
    assert isPrime(-1) == False

```

# Exemplo

## 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*

# Exemplo

## Saida

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

```
[*] Start mutation process:
- targets: isPrime
- tests: test_isPrime
[*] 3 tests passed:
- test_isPrime [3.07469 s]
[*] Start mutants generation and execution:
- [# 1] AOR isPrime:
-----
3:
4:     if num > 1:
5:
6:         for i in range(2, num):
7:             if num % i == 0:
- 7:             if num * i == 0:
+ 7:             if num * i == 0:
8:                 return False
9:         else:
10:
11:             return True
-----
[1.45151 s] killed by testing program mutpy/test_isPrime.py::CalculatorTest::test_nonprime
- [# 2] COI isPrime:
-----
1:
2: def isPrime(num):
3:
- 4:     if num > 1:
+ 4:     if not (num > 1):
5:
6:         for i in range(2, num):
7:             if num % i == 0:
8:                 return False
```

# Exemplo

## 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](mailto:thiago.nogueira@pe.senac.br)