



Sistema de Informação

Júlio Geovani Oliveira Guimarães
Atividade 2 - Testes de Mutação

Universidade Federal de Sergipe / São Cristóvão
3 de Setembro de 2024

Repositório: <https://github.com/valdas-v1/A-Simple-Python-Calculator/tree/main>

Link para vídeo:

<https://drive.google.com/file/d/1FifUl8tLfDjR9UdBGn2nSleK6pGrTP9u/view?usp=drivesdk>

Descrição do repositório: Utilizei um projeto básico no github que executa funções básicas de uma calculadora, nesse caso, soma, subtração, multiplicação, divisão, potência de 2 e raiz quadrada.

Tecnologias utilizadas

- Sistema Operacional
- IDE: Visual Studio Code
- Ferramentas: Venv(Ambiente Virtua), pytes, pytest-cov e mutmut.

1. Configurações feitas no ambiente

1.1 Em primeiro, um clone no repositório:

Código: `"git clone https://github.com/valdas-v1/A-Simple-Python-Calculator.git"`

1.2 Em segundo, a instalação do venv:

Código: `"python -m venv venv"`

`"python3-m venv meu-diretório lvenv"`

1.3 Em terceiro, ativação do venv (Ambiente Virtual):

Código: `"venv/Scripts/activate"`

1.4 Em quarto, instalação dos pacotes já disponíveis no repositório:

Código: `"pip3 install -r requirements.txt"`

> **Observação:** Adição dos pacotes pytest-cov e mutmut ao requirements.txt.

2. Execução dos testes:

2.1 Execução do arquivo de teste utilizando o pytest:

Código: `"pytest -w meu-diretório\tests\test_calculator.py"`

Métodos executados: test_sum(), test_subtract(), test_multiply(), test_division(), test_square_root(), teste_invalide_value), test_memory.

Resultado: Todos os testes foram realizados e todos passaram corretamente.

```
A-Simple-Python-Calculator\tests\test_calculator.py::test_sum PASSED [ 14%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_subtract PASSED [ 28%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_multiply PASSED [ 42%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_division PASSED [ 57%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_square_root PASSED [ 71%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_invalid_value PASSED [ 85%]
A-Simple-Python-Calculator\tests\test_calculator.py::test_memory PASSED [100%]

===== 7 passed in 0.06s =====
```

2.2 Execução do teste de cobertura:

Código: `"pytest -vv meu-diretório\tests\test_calculator.py --cov=calculator"`

```
----- coverage: platform win32, python 3.12.5-final-0 -----
Name                                                    Stmts  Miss  Cover
-----
A-Simple-Python-Calculator\calculator\__init__.py         1     0   100%
A-Simple-Python-Calculator\calculator\calculator.py       58     0   100%
-----
TOTAL                                                    59     0   100%

===== 7 passed in 0.08s =====
```

Resultado: Todos os testes foram realizados e recebi o retorno que 100% do código está coberto de testes.

2.3 Gerando html com as informações do teste de cobertura:

Código: `"pytest-vv meu-diretório\tests\test_calculator.py --cov=calculator --cov-branch --cov-report html"`

Informações extraídas com a visualização na página web:

Coverage report: 100%

Files

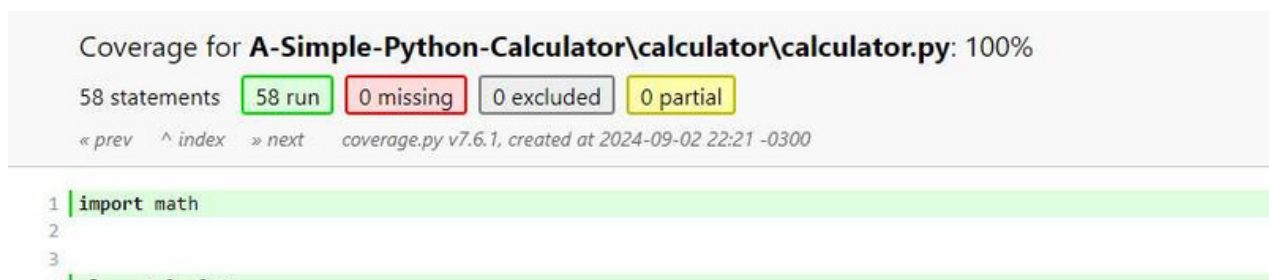
Functions

Classes

coverage.py v7.6.1, created at 2024-09-02 22:21 -0300

File ▲	statements	missing	excluded	branches	partial	coverage
A-Simple-Python-Calculator\calculator__init__.py	1	0	0	0	0	100%
A-Simple-Python-Calculator\calculator\calculator.py	58	0	0	14	0	100%
Total	59	0	0	14	0	100%

coverage.py v7.6.1, created at 2024-09-02 22:21 -0300



- Relatório de cobertura em 100%, ou seja, todas as linhas de código foram devidamente testadas.

> **Observação:** 59 linhas de código foram executadas, 0 ausentes, 0 excluídas e 0 parciais. Ou seja, êxito em 100% em relação aos testes de cobertura!

2.4 Execução do teste de mutação:

Código: `"mutmut run --paths-to-mutate=meu-diretório\calculator\calculator.py --tests-dir=meu-diretório\tests"`

```
Results are stored in .mutmut-cache.
Print found mutants with `mutmut results`.

Legend for output:
🦋 Killed mutants.   The goal is for everything to end up in this bucket.
🕒 Timeout.         Test suite took 10 times as long as the baseline so were killed.
🧐 Suspicious.      Tests took a long time, but not long enough to be fatal.
😬 Survived.         This means your tests need to be expanded.
🚫 Skipped.         Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
.: 38/38 🦋 35 🕒 0 🧐 0 😬 3 🚫 0
```

Resultados:

- 38 mutants foram inseridos no código.
- 35 foram mortos.
- Nenhum teve o tempo expirado.
- Nenhum suspeito.
- 3 sobreviveram.
- Nenhum foi ignorado.

Conclusão: De forma geral, os casos de teste se comportaram bem em relação ao teste de mutação, eliminaram praticamente todos os 38 mutants e apenas 3 sobreviveram.

2.5 Analisando mutants que sobreviveram:

- Imprimindo os mutantes:

Código: "mutmut results"

```
To apply a mutant on disk:
  mutmut apply <id>

To show a mutant:
  mutmut show <id>

Survived 😬 (3)

---- C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py (3) ----
1, 52-53
```

- Checando mais informações sobre o mutant de acordo com seu id:

Código: "mutmut show 1"

```
@@ -6,7 +6,7 @@
    """Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

    def __init__(self, memory=0):
-        self.memory = memory
+        self.memory = None

    def reset_memory(self):
        """
```

Resultado: No caso deste teste de mutação, foi passado para o self.memory o valor None em vez do 0 inicialmente. Como se trata de uma calculadora com métodos para execução de operações aritméticas entre números, valores nulos podem causar erros de execução. No código não existe uma tratativa na classe Calculator que trata None da mesma forma que trata 0 ou outros valores iniciais.

> Visualização do html gerado pelo código "mutmut html":

Mutation testing report

Killed 35 out of 38 mutants

File	Total	Skipped	Killed	% killed	Survived
C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py	38	0	35	92.11	3

Survived

Survived mutation testing. These mutants show holes in your test suite.

Mutant 1

```
--- C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
+++ C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
@@ -5,7 +5,7 @@
"""Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

- def __init__(self, memory=0):
+ def __init__(self, memory=1):
    self.memory = 0

    def reset_memory(self):
```

Mutant 59

```
--- C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
+++ C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
@@ -6,7 +6,7 @@
"""Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

    def __init__(self, memory=0):
        self.memory = 0
+        self.memory = 1

    def reset_memory(self):
        """
```

Mutant 60

```
--- C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
+++ C:\Users\JulioG\Downloads\testeSoftware\A-Simple-Python-Calculator\calculator\calculator.py
@@ -6,7 +6,7 @@
"""Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

    def __init__(self, memory=0):
        self.memory = 0
+        self.memory = None

    def reset_memory(self):
        """
```


3. Melhorias realizadas:

3.1 Garantir que self.memory receba o valor 0, do memory já iniciado no init.

Código que o teste foi realizado:

```
def __init__(self, memory=0):  
    self.memory = 0  
  
def reset_memory(self):  
    """  
    Resets the memory back to zero  
    """  
  
    self.memory = 0
```

Código com a alteração:

```
def __init__(self, memory=0):  
    self.memory = memory  
  
def reset_memory(self):  
    """  
    Resets the memory back to zero  
    """  
  
    self.memory = 0
```

Alteração de melhoria:

3.2 Verificar para não realizar raiz quadrada de número negativo.

Código que o teste foi realizado:

```
def sqrt(self, a=None):  
    |  
    try:  
        if a == None:  
            a = self.memory  
  
        self.memory = math.sqrt(float(a))  
        return self.memory  
    except:  
        return 'Invalid input'
```

Código com alteração:

```
try:  
    if a == None:  
        a = self.memory  
    a = float(a)  
    if a < 0:  
        return 'Invalid Input'  
    |  
    self.memory = math.sqrt(float(a))  
    return self.memory  
except:  
    return 'Invalid input'
```

```
assert c.sqrt(256) == 16  
assert c.sqrt() == 4  
assert c.sqrt() == 2  
assert c.sqrt(-4) == 'Invalid input'
```

Resultado com as alterações:

```
🔪 Killed mutants. The goal is for everything to end up in this bucket.  
🕒 Timeout. Test suite took 10 times as long as the baseline so were killed.  
😟 Suspicious. Tests took a long time, but not long enough to be fatal.  
😟 Survived. This means your tests need to be expanded.  
🚫 Skipped. Skipped.  
  
1. Using cached time for baseline tests, to run baseline again delete the cache file  
  
2. Checking mutants  
.: 37/37 🚫 35 🕒 0 😟 0 😟 2 🚫 0
```


Survived

Survived mutation testing. These mutants show holes in your test suite.

Mutant 1

```
--- C:\Users\JulioG\Downloads\projetoTS\A-Simple-Python-Calculator\calculator\calculator.py
+++ C:\Users\JulioG\Downloads\projetoTS\A-Simple-Python-Calculator\calculator\calculator.py
@@ -5,7 +5,7 @@
    """Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

-    def __init__(self, memory=0):
+    def __init__(self, memory=1):
        self.memory = memory

    def reset_memory(self):
```

Mutant 2

```
--- C:\Users\JulioG\Downloads\projetoTS\A-Simple-Python-Calculator\calculator\calculator.py
+++ C:\Users\JulioG\Downloads\projetoTS\A-Simple-Python-Calculator\calculator\calculator.py
@@ -6,7 +6,7 @@
    """Calculator class with methods to sum, subtract, multiply, divide square and find square root"""

    def __init__(self, memory=0):
-        self.memory = memory
+        self.memory = None

    def reset_memory(self):
        """
```

Resultado: Um dos 3 mutants foi eliminado, gerando assim um melhor resultado em relação aos testes de mutação. Dessa forma, sobraram apenas dois mutants.

Novos casos de teste resultaram em uma melhor detecção de mutantes, melhorando a confiabilidade do código.