Live de Python #42

Testes de mutação

Roteiro

- O que são testes de mutação?
- Tipos de mutação
- Mutpy
  - Linha de comando
- Code !!!

Testes de mutação

Teste de mutação é uma técnica baseada em **defeitos**, hã? Ela vai trocar operadores, trocas assignment, ela vai modificar o seu código e os seus testes e vai ver como eles reagem a isso. Ou seja, partindo dos problemas do seu código ele vai funcionar.

Testes de mutação

Sim, tudo fica mais fácil com exemplos.

```
In [1]: def soma(x, y):
   ...:     return x + y
   ...:

In [2]: """Iniciando os testes."""
Out[2]: 'Iniciando os testes.'

In [3]: assert soma(0, 0) == 0

In [4]: assert soma(1, 0) == 1
```

Testes de mutação

Trocamos o operador principal da função (+ por -) e ainda assim os testes continuam funcionando

```
In [5]: def soma(x, y):
   ...:     return x - y
   ...:

In [6]: assert soma(0, 0) == 0

In [7]: assert soma(1, 0) == 1
```

Testes de mutação

Tá, me e agora?

O que fizemos agora é uma técnica chamada **AOR** (arithmetic operator replacement) ou Substituição de operador aritmético.

Isso pode parecer uma coisa extremamente simples, e de fato é, mas representa uma parcela boa dos nossos testes.

Testes de mutação

Ou seja, os testes de mutação tem o objetivo de nos ajudar a escrever código melhor, testes melhores. Ele vai apontar falhas no desenvolvimento baseado nos testes de escrevemos. Isso pode não parecer muito, mais se usarmos técnicas de mutação misturadas, vamos ter um grande campo de correções e “defeitos”

Tipos de mutação (Aritméticas)

- **AOD** - arithmetic operator deletion
- **AOR** - arithmetic operator replacement

```
def invert(x):
    return -x
```

AOD: Vai deletar o operador, ou seja, vai retornar somente ‘x’

```
def soma(x, y):
    return x + y
```

AOR: Vai trocar o operador ‘+’ e tentar com outros operadores (-, *, **, /, //)

Tipos de mutação (Atribuição)

- **ASR** - assignment operator replacement

```
def sum_(*vals):
    result = 0
    for val in vals:
        result += val
    return result
```

ASR: Vai trocar o operador de atribuição (-=, *=, ..)

Tipos de mutação (Constantes)

- **CRP** - constant replacement

```
def sum_(*vals):
    result = 0
    for val in vals:
        result += val
    return result
```

CRP: Vai mudar o valor da constante (result) para qualquer outro valor

Tipos de mutação (Condicional)

- **COD** - conditional operator deletion
- **COI** - conditional operator insertion

```
def is_number(x):
    if not isinstance(x, Number):
        return False
    return True
```

COD: Vai deletar o not (vai tentar executar sem a condicional)

Tipos de mutação (Condicional)

- **COD** - conditional operator deletion
- **COI** - conditional operator insertion

```
def is_number(x):
    if not isinstance(x, Number):
        return False
    return True
```

COI: Vai inserir um operador, por exemplo (not, not) o que seria uma deleção também

Tipos de mutação (Op. Lógicas)

- **LCR** - logical connector replacement
- **LOD** - logical operator deletion
- **LOR** - logical operator replacement

```
def check_numbers(x, y):
    if is_number(x) and is_number(y):
        return True
    return False
```

LCR: Vai trocar o conectivo (and) por ‘or’

Para saber mais

Description of muJava's Method-level Mutation Operators

Yu-Seung Ma
Electronics and Telecommunications Research Institute, Korea
ysma@etri.re.kr

Jeff Offutt
Software Engineering
George Mason University
offutt@gmu.edu

November 29, 2005
Correction December 2011
Update July 2016