MUTATION TESTING

OR WHO IS GOING TO TEST YOUR TESTS ?

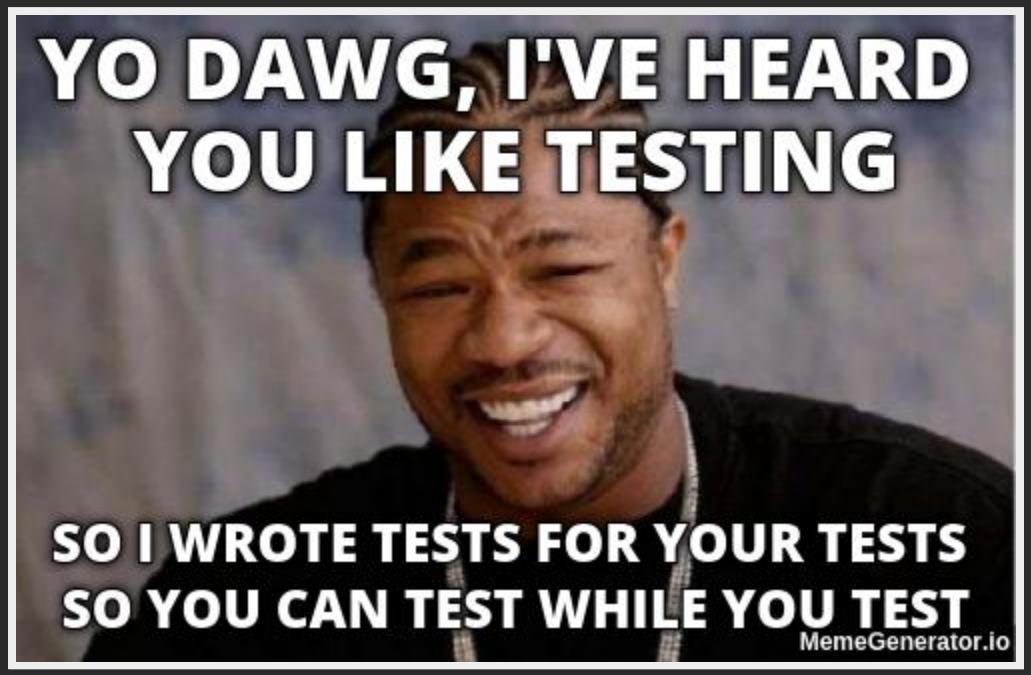
1st Developers@CERN Forum

Created by Sebastian Witowski

DO YOU WRITE CODE ?

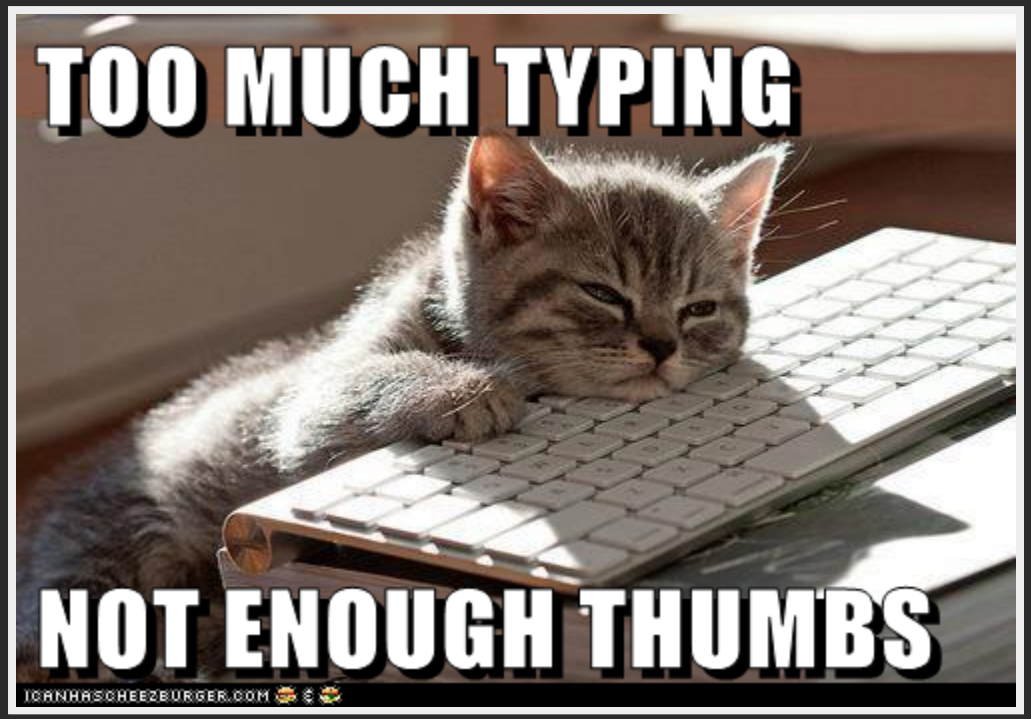
DO YOU TEST YOUR CODE ?

DO YOU TEST YOUR TESTS ?



DO YOU TEST THE TESTS FOR YOUR TESTS ?

TESTING TESTS ?



Richard Lipton, *Fault Diagnosis of Computer Programs*, 1971

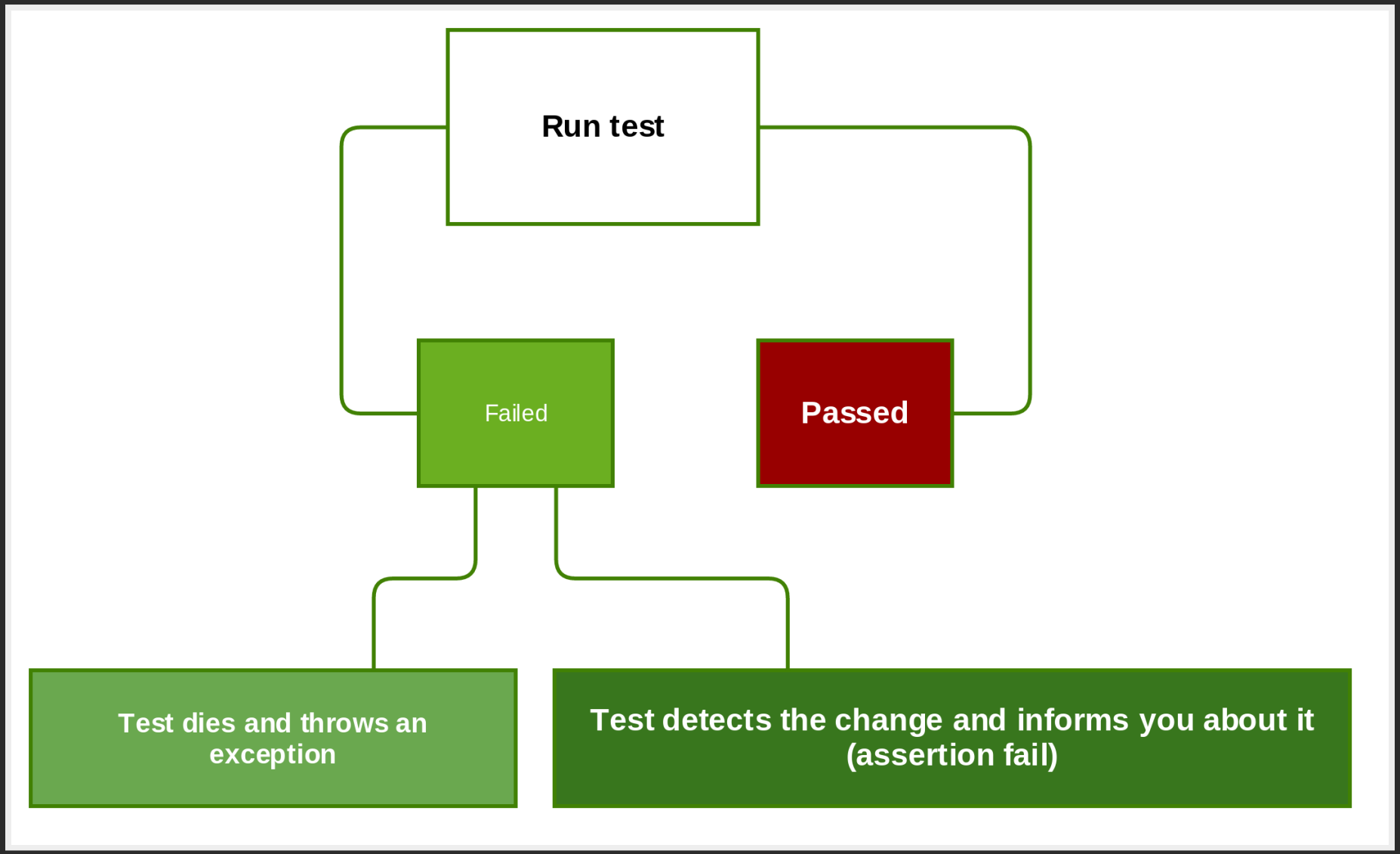
HOW DOES THIS MUTATION TESTING WORKS ?

STEP 1. CHANGE YOU CODE IN A SMALL WAY:

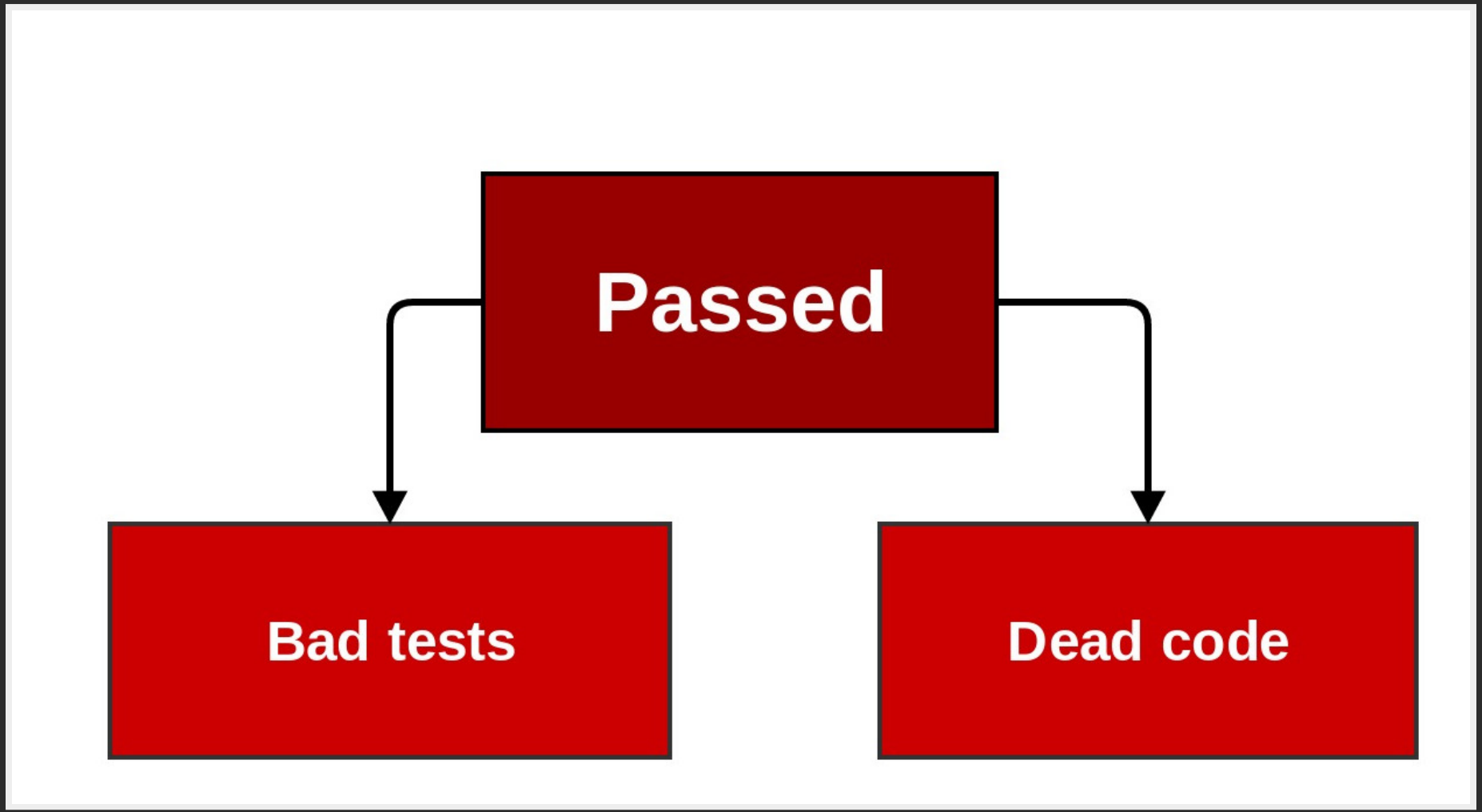
- a + b ---> a * b
- a + 1 ---> a + 2
- a + b ---> a + c
- a + 1 ---> a + 1
  a + 1
- (if a == 1 and b > 1) ---> (if a == 1 or b > 1)

Changes similar to small, programming errors.

STEP 2. RUN YOUR TESTS



STEP 2. RUN YOUR TESTS



STEP 3. GET THE MUTATION SCORE

$$\text{Mutation score} = \frac{\text{number or mutants killed}}{\text{number of mutants created}}$$

STEP 4. PROFIT

EXAMPLE TIME

```
def multiply(a, b):
    return a * b
```

```
def multiply(a, b):
    return a * b
```

```
class CalculatorTest(TestCase):
    def test_multiply(self):
        self.assertEqual(multiply(2, 2), 4)
```

```
self.assertEqual(multiply(2, 2), 4)
```

```
self.assertEqual(multiply(3, 3), 9)
```


STEM CELL
RESEARCH
LAB
WHY ARE WE NOT FUNDING THIS?!
quickmeme.com

EQUIVALENT MUTANT PROBLEM

```
index = 0
while True:
    do_stuff()
    index = index + 1
    if index == 10:
        break
```

if index == 10:

vs.

if index >= 10:

WRAP UP

THE GOOD PARTS

- Detect problems with your tests
- They discover dead code
- They are automatic
- How else would you test your tests ?
- (Semi-)Automatic tool for testing ? I'm in !

THE NOT SO GOOD PARTS

- Mutation testing is slow:
  (TIME = ALL MUTANTS x ALL TESTS)
- Handful of libraries
- Equivalent Mutant Problem
- Writing complex mutant tests is difficult

MUTANT TESTING LIBRARIES

- Mutant - Ruby (last updated September 2015)
- VisualMutator - C# (last updated September 2015)
- Pitest - Java (last updated August 2015)
- Humbug - PHP (last updated May 2015)
- MuCheck - Haskell (last updated January 2015)
- MutPy - Python3 (last updated January 2014)
- Mutator - commercial solution for Java, Ruby, JavaScript and PHP

EXAMPLE

MutPy (requires Python3)

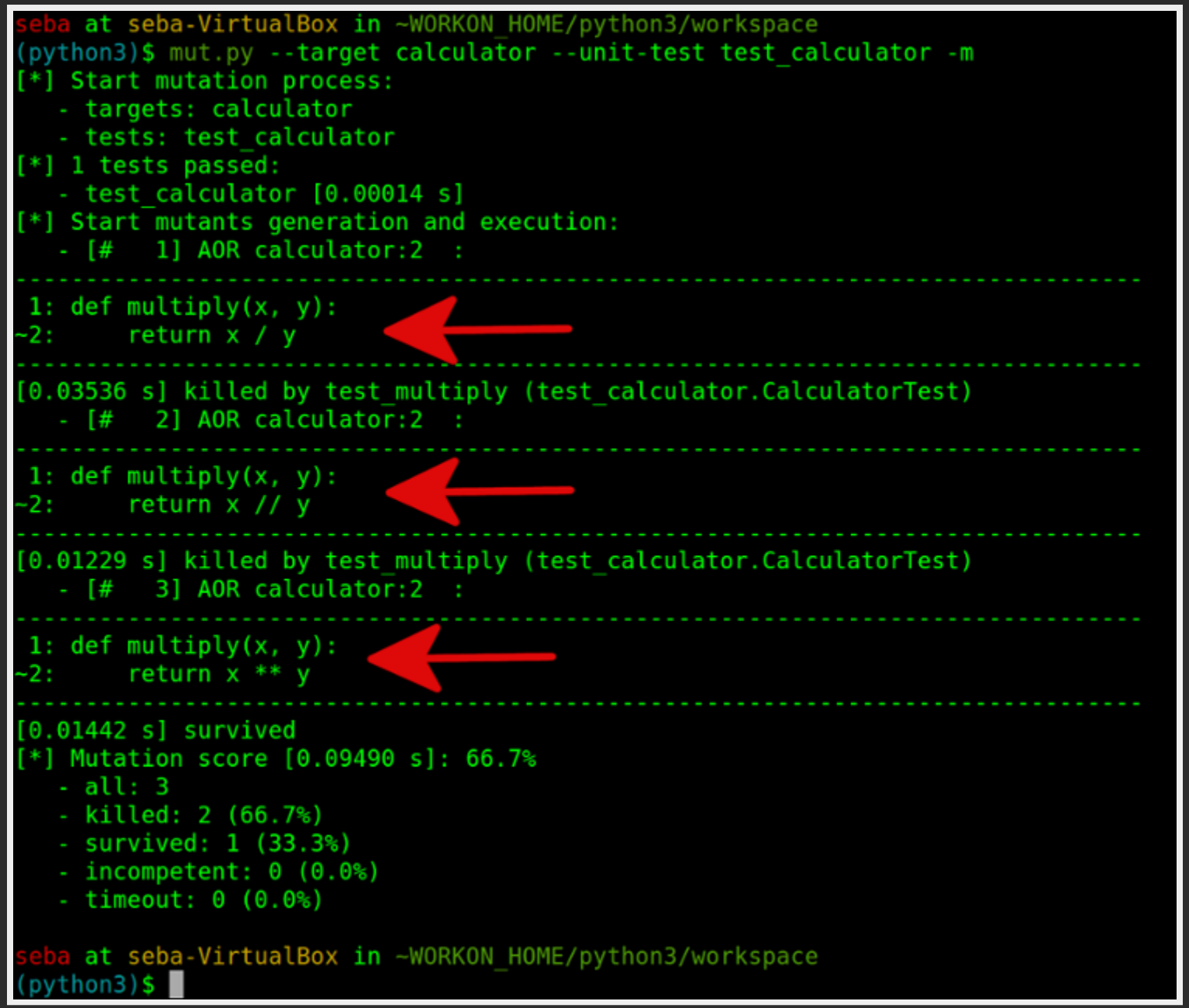
calculator.py

```
def multiply(a, b):
    return a * b
```

test_calculator.py

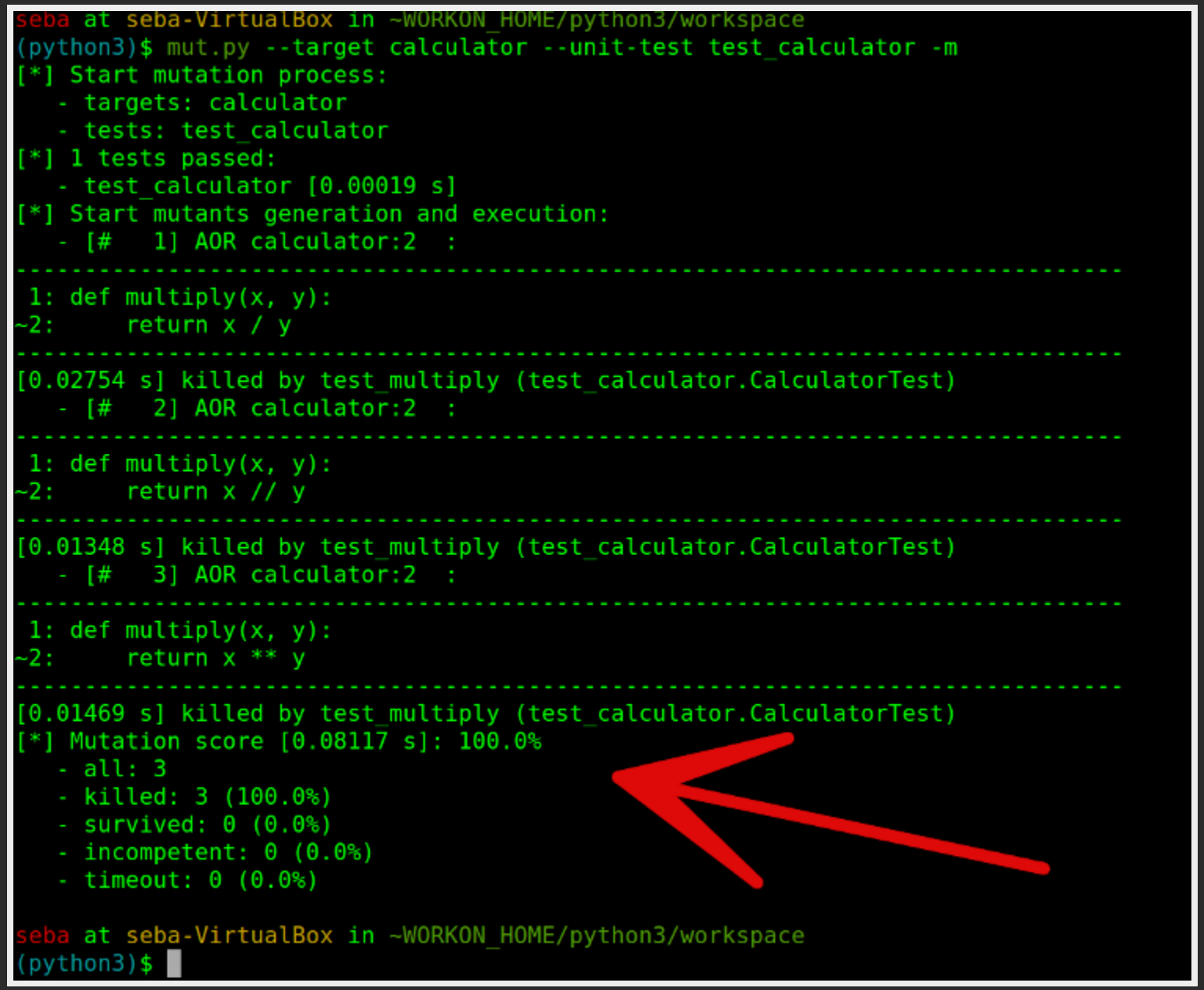
```
from unittest import TestCase
from calculator import multiply

class CalculatorTest(TestCase):
    def test_multiply(self):
        self.assertEqual(multiply(2, 2), 4)
```


seba at seba-VirtualBox in ~WORKON_HOME/python3/workspace
(python3)$ mut.py --target calculator --unit-test test_calculator -m
[*] Start mutation process:
- targets: calculator
- tests: test_calculator
[*] 1 tests passed:
- test_calculator [0.00014 s]
[*] Start mutants generation and execution:
- [# 1] AOR calculator:2 :
1: def multiply(x, y):
~2: return x / y
[0.03536 s] killed by test_multiply (test_calculator.CalculatorTest)
- [# 2] AOR calculator:2 :
1: def multiply(x, y):
~2: return x // y
[0.01229 s] killed by test_multiply (test_calculator.CalculatorTest)
- [# 3] AOR calculator:2 :
1: def multiply(x, y):
~2: return x ** y
[0.01442 s] survived
[*] Mutation score [0.09490 s]: 66.7%
- all: 3
- killed: 2 (66.7%)
- survived: 1 (33.3%)
- incompetent: 0 (0.0%)
- timeout: 0 (0.0%)
seba at seba-VirtualBox in ~WORKON_HOME/python3/workspace
(python3)$

```
self.assertEqual(multiply(2, 2), 4)
```

```
self.assertEqual(multiply(3, 3), 9)
```


seba at seba-VirtualBox in ~WORKON_HOME/python3/workspace
(python3)$ mut.py --target calculator --unit-test test_calculator -m
[*] Start mutation process:
- targets: calculator
- tests: test_calculator
[*] 1 tests passed:
- test_calculator [0.00019 s]
[*] Start mutants generation and execution:
- [# 1] AOR calculator:2 :
1: def multiply(x, y):
~2: return x / y
[0.02754 s] killed by test_multiply (test_calculator.CalculatorTest)
- [# 2] AOR calculator:2 :
1: def multiply(x, y):
~2: return x // y
[0.01348 s] killed by test_multiply (test_calculator.CalculatorTest)
- [# 3] AOR calculator:2 :
1: def multiply(x, y):
~2: return x ** y
[0.01469 s] killed by test_multiply (test_calculator.CalculatorTest)
[*] Mutation score [0.08117 s]: 100.0%
- all: 3
- killed: 3 (100.0%)
- survived: 0 (0.0%)
- incompetent: 0 (0.0%)
- timeout: 0 (0.0%)
seba at seba-VirtualBox in ~WORKON_HOME/python3/workspace
(python3)$

THE FUTURE ?

THANK YOU !

ANY QUESTIONS ?

Happy ~~coding~~ testing !

This presentation is available on github, so you can see the slides on github pages