# Introduction

This document contains the example unit test to prove that unit tests are working for our language. We selected Python as development language, so the files are python files. We translated the code of the example unit test from <http://wiki.fernuni-hagen.de/eclipse/index.php/Einf%C3%BChrung_in_JUnit> to Python. As the check for infinite loops without a crash of the whole unit test is a lot more complicated in Python than in Java, the check for the square root function was not executed. If you remove the comment marks, the unit test will crash, so you are aware that there is an infinite loop.

In section 2 you can find the failing test examples and in section 3 the corrected examples are shown.

# Non-Working Unit Test

The calculator file (calculator.py):

class Calculator(object):

def \_\_init\_\_(self):

self.result = 0

def add(self, n):

self.result = self.result + n

def subtract(self, n):

self.result = self.result - 1

def multiply(self, n):

pass

def divide(self, n):

self.result = self.result / n

def square(self, n):

self.result = n \* n

def squareRoot(self, n):

while True:

pass

def clear(self):

self.result = 0

def switchOn(self):

self.result = 0

def switchOff(self):

pass

def getResult(self):

return self.result

The unit test file (calculatorTest.py):

import calculator

import unittest

class calculatorTest(unittest.TestCase):

def setUp(self):

print "\nSwitch on calculator"

self.calculator = calculator.Calculator()

self.calculator.switchOn()

self.calculator.clear()

def tearDown(self):

print "\nSwitch of calculator"

self.calculator.switchOff()

def test\_add(self):

self.calculator.add(1)

self.calculator.add(1)

self.assertEqual(self.calculator.getResult(), 2)

def test\_subtract(self):

self.calculator.add(10)

self.calculator.subtract(2)

self.assertEqual(self.calculator.getResult(), 8)

def test\_divide(self):

self.calculator.add(8)

self.calculator.divide(2)

self.assertEqual(self.calculator.getResult(), 4)

def test\_divideByZero(self):

self.assertRaises(ZeroDivisionError, self.calculator.divide, 0)

#def test\_squareRoot(self):

# self.calculator.squareRoot(100)

# self.assertEqual(calculator.getResult(), 10)

def test\_multiply(self):

self.calculator.add(10)

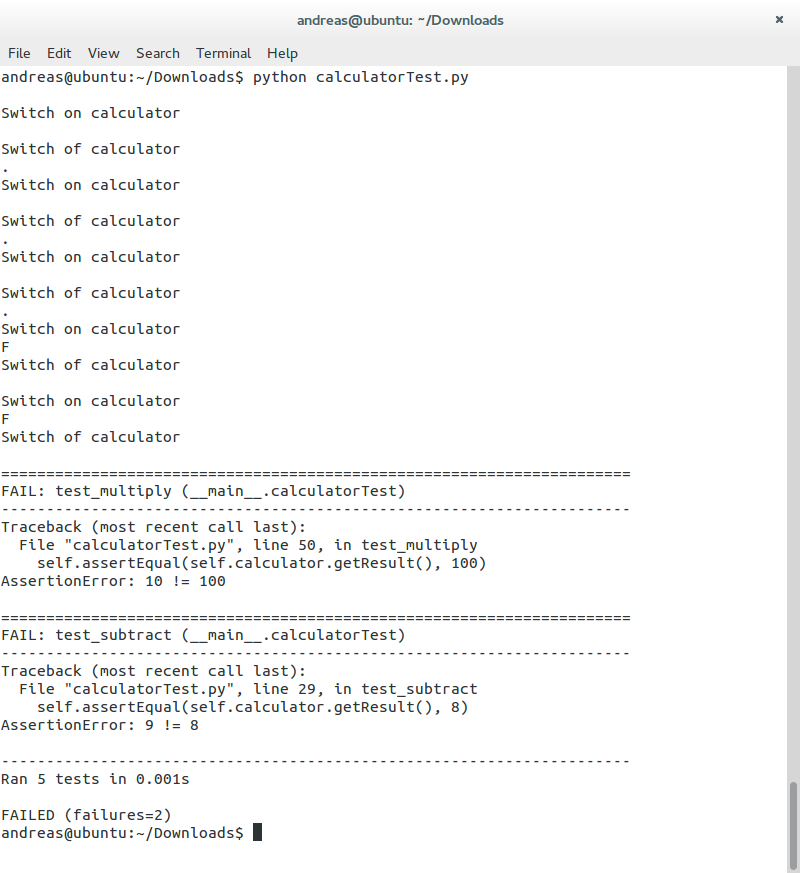
self.calculator.multiply(10)

self.assertEqual(self.calculator.getResult(), 100)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Test result:



# Working Unit Test

The calculator file (calculator.py) is corrected:

class Calculator(object):

def \_\_init\_\_(self):

self.result = 0

def add(self, n):

self.result = self.result + n

def subtract(self, n):

self.result = self.result - n

def multiply(self, n):

self.result = self.result \* n

def divide(self, n):

self.result = self.result / n

def square(self, n):

self.result = n \* n

def squareRoot(self, n):

while True:

pass

def clear(self):

self.result = 0

def switchOn(self):

self.result = 0

def switchOff(self):

pass

def getResult(self):

return self.result

The unit test file (calculatorTest.py) is unchanged:

import calculator

import unittest

class calculatorTest(unittest.TestCase):

def setUp(self):

print "\nSwitch on calculator"

self.calculator = calculator.Calculator()

self.calculator.switchOn()

self.calculator.clear()

def tearDown(self):

print "\nSwitch of calculator"

self.calculator.switchOff()

def test\_add(self):

self.calculator.add(1)

self.calculator.add(1)

self.assertEqual(self.calculator.getResult(), 2)

def test\_subtract(self):

self.calculator.add(10)

self.calculator.subtract(2)

self.assertEqual(self.calculator.getResult(), 8)

def test\_divide(self):

self.calculator.add(8)

self.calculator.divide(2)

self.assertEqual(self.calculator.getResult(), 4)

def test\_divideByZero(self):

self.assertRaises(ZeroDivisionError, self.calculator.divide, 0)

#def test\_squareRoot(self):

# self.calculator.squareRoot(100)

# self.assertEqual(calculator.getResult(), 10)

def test\_multiply(self):

self.calculator.add(10)

self.calculator.multiply(10)

self.assertEqual(self.calculator.getResult(), 100)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Test result:

