**TESTING IN PYTHON – UNIT TEST & SCRIPT**

**Version 2.0**

*This is a supplementary material of the Course “Software Quality Assurance – Testing”, Summer Semester-FPT University, 2021, delivered by* ***Tran Dinh Que****,* [*quetd2@fe.edu.vn*](mailto:quetd2@fe.edu.vn) *. This file will be updated regularly and displayed in class.*

The simplest tool for editing code is **Notepad**. You may utilize the other editors such as **visual studio**.

The purpose of this simple tutorial is to assist students to be familiar with Python:

* Writing functions in Python (if…elif…else, for)
* Call functions from some file
* Writing script for testing (unit and integration) in Python

There are two simple ways of unit testing SCRIPT with test cases in Python

* Using simple **assertion**
* Using library **unittest**

Students can download and implement PYTHON at the web page: <https://www.python.org/downloads/>

1. **Delicious smell of Python PYTHON**

Easy to access, easy to learn coding

<https://www.tutorialspoint.com/python/python_lists.htm>

* 1. **Definition of functions in python**

#Define sum of sequence of numbers

def sum(arg):

total = 0

for val in arg:

total += val

return total

#Define max of a sequence of numbers

def maxSequence(u):

max = u[0]

for elem in u:

if elem > max:

max = elem

return max

**If…elif….else in Python**

**1 def functCond1(x,y):**

2 if (1 > x > 0) and y > 0:

3 return x+y

4 elif (x > 1) and y < 0:

5 return x\*y

6 else:

7 return x-y

TC#1: (0.5, 1) lines 1-2-3 coverage 3/7

TC#2: (2,6) lines 1-2-4-6-7 coverage 5/7

TC#1 and TC#2: line 1,2,2,4,6,7 6/7

TC#3: (2,-1): 5 is executed

How to get 100%???

TC#1,2,3: 7/7=100%

* 1. **Call functions from functions**

**def max2(x,y):**

if x >= y:

return x

else:

return y

**def max3(x,y,z):**

t = max2(x,y);

u = max2(t,z);

return u

* 1. **Call functions from files**

Refer: <https://www.geeksforgeeks.org/python-call-function-from-another-file/>

You need to edit three files

# edit file number1: **display.py**

# function defined in display.py

def displayText():

print("\n Hello !")

# edit file number2: **calc.py**

# functions defined in calc.py

**def addNumbers(a, b):**

print("Sum is ", a + b)

**def subtractNumbers(a, b):**

print("Difference is ", a-b)

**def multiplyNumbers(a, b):**

print("Product is ", a \* b)

**def divideNumbers(a, b):**

print("Division is ", a / b)

**def modulusNumbers(a, b):**

print("Remainder is ", a % b)

#edit file number3: **fileTest.py**

# importing all the functions defined in **calc.py**

**from calc import \***

# importing required functions

# defined in display.py

**from display import displayText**

# calling functions defined

# in calc.py

**addNumbers(25, 6)**

**subtractNumbers(25, 6)**

**multiplyNumbers(25, 6)**

**divideNumbers(25, 6)**

**modulusNumbers(25, 6)**

# calling function defined

# in display.py

**displayText()**

1. **Testing – Test cases – Assertion**

**Example 1**

**def functCond1(x,y):**

if (1 > x > 0) and y > 0:

return x+y

elif (x > 1) and y < 0:

return x\*y

else:

return x-y

**#=================**

**#test case:(1,2)**

**#test case: (0, )**

**#.......**

**#======================**

**def test\_functCond1():**

assert functCond1(0, ) == ??, "Should be 6"

if \_\_name\_\_ == "\_\_main\_\_":

test\_functCond1()

print("Everything passed")

**Example 2**

1. Edit file **maxSeq.py** containing the following function

======

#Define max of a sequence of numbers

**def maxSequence(u):**

max = u[0]

for elem in u:

if elem > max:

max = elem

return max

1. Edit file for testing **maxSeqTesting.py** containing the following function

======

#syntax for testing testcases

#without library unittest

def test\_maxSequence():

assert maxSequence([1, 2, 3]) == 3, "Should be 3"

#**you replace 3 by 2 to see what occurs!**

**if \_\_name\_\_ == "\_\_main\_\_":**

**test\_maxSequence()**

**print("Everything passed")**

1. Write test cases and Testing. Happy with this
2. **Testing – Test cases – Unittest**

**Example 1**

**import unittest**

#Define sum of sequence of numbers

def sum(arg):

total = 0

for val in arg:

total += val

return total

#build test

**class TestSum(unittest.TestCase):**

*#inherit from unittest*

def test\_list\_int(self):

data = [1.5,-1,5.2,6,7]

result = sum(data)

self.***assertEqual(result, 6)***

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

**unittest.main()**

**Example 2: Using two types of testing. Enjoy ☺**

1. **Edit in one file** *maxD.py*

##Define max of a sequence of numbers

def maxDayso(u):

max = u[0]

for x in u:

if x >= max:

max = x

return max

1. Edit in one file *maxAserTest*

##syntax for testing testcases

##without library unittest

from maxD import \*

**def test\_maxDayso():**

**assert maxDayso([1, 2, 3]) == 3, "Should be 3"**

**if \_\_name\_\_ == "\_\_main\_\_":**

**test\_maxDayso()**

**print("Everything passed")**

1. **Edit in one file** *maxUnitTest*

**import unittest**

**from maxD import \***

**class TestmaxDay(unittest.TestCase):**

**#inherit from unittest**

**def test\_list\_int(self):**

**data = [1, 2, 3]**

**result = maxDayso(data)**

**self.assertEqual(result, 4)**

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

**unittest.main()**

**Running and enjoy ☺**

1. **Discover Python furthermore**

Data structures

<https://docs.python.org/3/tutorial/datastructures.html>

<https://pythonhosted.org/gchecky/unittest-pysrc.html>

<https://docs.python.org/3.0/library/unittest.html>

**Django**