Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5
“Модульное тестирование в Python”

Выполнил:
студент группы ИУ5-34Б:
Малютин И.Д.
Подпись и дата:

Проверила:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022

Описание задания:

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
   - TDD – фреймворк 3 теста.
   - BDD – фреймворк 3 теста.
   - Создание Mock-объектов.

Код программы.
main.py

```
import random
import time as t


def gen_random(num_count, begin, end):
    mylist = []
    try:
        for i in range(num_count):
            mylist.append(random.randint(begin,end))
    except:
        return []
    return mylist


def cm_timer(sleeping_time):
    start = t.time()
    try:
        sleeping_time = int(sleeping_time)
    except:
        return 0
    t.sleep(sleeping time)
    return int(t.time()-start)


def sort(data):
    try:
        mas = sorted(data, key=lambda x: abs(x), reverse=True)
    except:
        return 0
    return mas
```

TDD.py

```
import unittest
import sys, os

sys.path.append(os.getcwd()) #current working directory
from main import *

#================================================================================
==============================

class TestGenRandom(unittest.TestCase):
    def test_gen_random_returns_list(self):
        self.assertEqual(list(set(gen_random(100,1,3))),[1,2,3])
        self.assertEqual(list(set(gen_random(0, 4, 6))), [])

    def test_gen_random_receives_not_integer_returns_empty(self):
        self.assertEqual(gen_random(1.1, 2, 3), [])
        self.assertEqual(gen_random(1, 12.3, 4), [])
        self.assertEqual(gen_random(15, 2, 14.11), [])

    def test_gen_random_receives_alpha_string_returns_empty(self):
        self.assertEqual(gen_random('a', 2, 3), [])
        self.assertEqual(gen_random(1, 'a', 4), [])
        self.assertEqual(gen_random(15, 2, 'a'), [])

#================================================================================
==============================

class TestCmTimer(unittest.TestCase):
    def test_cm_timer_returns_same(self):
        self.assertEqual(3, 3)
        self.assertEqual(4, 4)
        self.assertEqual(5, 5)

    def test_cm_timer_returns_integer(self):
        self.assertIsInstance(cm_timer(3),int)

    def test_cm_timer_receives_string_returns_integer(self):
        self.assertEqual(cm_timer('2'),2)

    def test_cm_timer_receives_alpha_string_returns_zero(self):
        self.assertEqual(cm_timer('asdadassda'),0)

#================================================================================
==============================

class TestSort(unittest.TestCase):
    def test_sort_returns_sorted_list(self):
        self.assertEqual(sort([4, -30, 100, -100, 123, 1, 0, -1, -4]),[123, 100,
-100, -30, 4, -4, 1, -1, 0])

    def test_sort_receives_not_list(self):
        self.assertEqual(sort(1),0)
        self.assertEqual(sort(1.11), 0)
        self.assertEqual(sort('a'), 0)

if __name__=='__main__':
    unittest.main()
```

test_cm_timer.py

```
from pytest_bdd import scenarios, scenario, given, when, then
from pathlib import Path
import pytest
import sys,os

sys.path.append(os.getcwd()) #current working directory
from main import cm_timer

featureFileDir='myfeatures'
featureFile='cm_timer.feature'
BASE_DIR=Path(__file__).resolve().parent
FEATURE_FILE = BASE_DIR.joinpath(featureFileDir).joinpath(featureFile)


@scenario(FEATURE_FILE,'The program will fall asleep for the time specified by
the user')
def testing_cm_timer():
    pass

@given('I have the number 3 - it is sleeping time',target_fixture='params')
def params():
    return 3

@when('Program will fall asleep with cm_timer and return sleeping
time',target_fixture='created_timer')
def created_timer(params):
    return cm_timer(params)

@then('I expect the result to be same seconds as user specified')
def created_timer(created_timer):
    assert created_timer == 3
```

test_gen_random.py

```
from pytest_bdd import scenarios, scenario, given, when, then
from pathlib import Path
import pytest
import sys,os

sys.path.append(os.getcwd()) #current working directory
from main import gen_random

featureFileDir='myfeatures'
featureFile='gen_random.feature'
BASE_DIR=Path(__file__).resolve().parent
FEATURE_FILE = BASE_DIR.joinpath(featureFileDir).joinpath(featureFile)


@scenario(FEATURE_FILE,'A new array will be created from random numbers provided
by the user')
def testing_gen_random():
    pass

@given('I have the numbers 10, 1, 3',target_fixture='params')
def params():
```

```
    return 10,1,3

@when('Array get created with gen_random',target_fixture='created_array')
def created_array():
    return list(set(gen_random(10,1,3)))

@then('I expect the result to be array with random numbers 1-3 which set will be
[1,2,3]')
def created_array(created_array):
    assert created_array==[1,2,3]
```

test_sort.py

```
from pytest_bdd import scenario, given, when, then
from pathlib import Path
import pytest
import sys,os

sys.path.append(os.getcwd()) #current working directory
from main import sort

featureFileDir='myfeatures'
featureFile='sort.feature'
BASE_DIR=Path(__file__).resolve().parent
FEATURE_FILE = BASE_DIR.joinpath(featureFileDir).joinpath(featureFile)


@scenario(FEATURE_FILE,'Data need to be sorted by abs')
def testing_sort():
    pass

@given('Some data',target_fixture='data')
def data():
    return [4, -30, 100, -100, 123, 1, 0, -1, -4]

@when('Data get sorted with sort',target_fixture='using_sort')
def using_sort(data):
    return sort(data)

@then('Data is sorted')
def using_sort(using_sort):
    assert using_sort==[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

sort_feature.feature

```
Feature: Sorting elements in data
  Scenario: Data need to be sorted by abs
    Given Some data
    When Data get sorted with sort
    Then Data is sorted
```

gen_random_feature.feature

```
Feature: Creating a new array with random numbers
  Scenario: A new array will be created from random numbers provided by the user
    Given I have the numbers 10, 1, 3
    When Array get created with gen_random
    Then I expect the result to be array with random numbers 1-3 which set will
be [1,2,3]
```

cm_timer_feature.feature

```
Feature: Creating timer by lib time
  Scenario: The program will fall asleep for the time specified by the user
    Given I have the number 3 - it is sleeping time
    When Program will fall asleep with cm_timer and return sleeping time
    Then I expect the result to be same seconds as user specified
```

Работа тестов:

```
collecting ... collected 9 items

TDD.py::TestGenRandom::test_gen_random_receives_alpha_string_returns_empty
TDD.py::TestGenRandom::test_gen_random_receives_not_integer_returns_empty
TDD.py::TestGenRandom::test_gen_random_returns_list
TDD.py::TestCmTimer::test_cm_timer_receives_alpha_string_returns_zero PASSED [ 11%]PASSED [ 22%]PASSED             [ 33%]
TDD.py::TestCmTimer::test_cm_timer_receives_string_returns_integer
TDD.py::TestCmTimer::test_cm_timer_returns_integer
TDD.py::TestCmTimer::test_cm_timer_returns_same
TDD.py::TestSort::test_sort_receives_not_list PASSED [ 44%]PASSED [ 55%]PASSED              [ 66%]PASSED                [ 77%]PASSED                 [ 88%]
TDD.py::TestSort::test_sort_returns_sorted_list PASSED                 [100%]

============================== 9 passed in 5.08s ==============================

Process finished with exit code 0
Launching pytest with arguments C:\Users\60TP\PycharmProjects\lab5\test_sort.py --no-header --no-summary -q in C:\Users\60TP\PycharmProjects\lab5

============================== test session starts ==============================
collecting ... collected 1 item

test_sort.py::testing_sort <- ..\..\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 0.02s ==============================

Process finished with exit code 0
Launching pytest with arguments C:\Users\60TP\PycharmProjects\lab5\test_gen_random.py --no-header --no-summary -q in C:\Users\60TP\PycharmProjects\lab5

============================== test session starts ==============================
collecting ... collected 1 item

test_gen_random.py::testing_gen_random <- ..\..\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 0.02s ==============================

Process finished with exit code 0
Launching pytest with arguments C:\Users\60TP\PycharmProjects\lab5\test_cm_timer.py --no-header --no-summary -q in C:\Users\60TP\PycharmProjects\lab5

============================== test session starts ==============================
collecting ... collected 1 item

test_cm_timer.py::testing_cm_timer <- ..\..\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 3.02s ==============================

Process finished with exit code 0
```