Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Казовые компоненты интернет технологий»

Отчет по лабораторной работе №5
«Модульное тестирование в Python»

Выполнил:
студент группы ИУ5-33Б:

Лебедева С.К.

Руководитель:
преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2022 г.

Цель лабораторной работы:

Изучение возможностей модульного тестирования в Python.

Задание:

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 (не менее 3 тестов).
   - BDD - фреймворк (не менее 3 тестов).
   - Создание Mock-объектов (необязательное дополнительное задание).

Текст программы:

sort.py

```
data_1 = [4, -30, 100, -100, 123, 1, 0, -1, -4]

def sort_1(data):
    result = sorted(data, key = abs, reverse = True)
    #print(result)
    return result

def sort_2(data):
    result_with_lambda = sorted(data, key = lambda x: abs(x), reverse = True)
    return result_with_lambda

def main_s():
    result = sort_1(data_1)
    print (result)

    result_with_lambda = sort_2(data_1)
    print(result_with_lambda)

if __name__ == "__main__":
    main_s()
```

test_TDD.py

```
import unittest

import sort

class test_sort(unittest.TestCase):
    def test_sort_1_1(self):
```

```
        self.assertEqual(sort.sort_1([3, -4, 5, 0, 1]), [5, -4, 3, 1, 0])
    def test_sort_1_2(self):
        self.assertEqual(sort.sort_1([3, -4, 4, 5, 0, 1, -1, 17]), [17, 5, -4, 4,
3, 1, -1, 0])
    def test_sort_1_3(self):
        self.assertEqual(sort.sort_1([0, -100, 100, 67, -67, 67, 99, 15, 16, -
15]), [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0])

    def test_sort_2_1(self):
        self.assertEqual(sort.sort_2([3, -4, 5, 0, 1]), [5, -4, 3, 1, 0])
    def test_sort_2_2(self):
        self.assertEqual(sort.sort_2([3, -4, 4, 5, 0, 1, -1, 17]), [17, 5, -4, 4,
3, 1, -1, 0])
    def test_sort_2_3(self):
        self.assertEqual(sort.sort_2([0, -100, 100, 67, -67, 67, 99, 15, 16, -
15]), [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0])

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

features/tutorial.feature

```
Feature: Abs_sorting

    Scenario: Sort Abs
        Given the list is [3, -4, 5, 0, 1]
        When the list is sorted
        Then the new list is [5, -4, 3, 1, 0]

     Scenario: Sort Abs2
        Given the list is [3, -4, 4, 5, 0, 1, -1, 17]
        When the list is sorted2
        Then the new list is [17, 5, -4, 4, 3, 1, -1, 0]

    Scenario: Sort Abs3
        Given the list is [0, -100, 100, 67, -67, 67, 99, 15, 16, -15]
        When the list is sorted3
        Then the new list is [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0]
```

features/steps/sort.py

```
data_1 = [4, -30, 100, -100, 123, 1, 0, -1, -4]

def sort_1(data):
    result = sorted(data, key = abs, reverse = True)
    #print(result)
    return result

def sort_2(data):
    result_with_lambda = sorted(data, key = lambda x: abs(x), reverse = True)
    return result_with_lambda
```

```
def main_s():
    result = sort_1(data_1)
    print (result)

    result_with_lambda = sort_2(data_1)
    print(result_with_lambda)

if __name__ == "__main__":
    main_s()
```

features/steps/test1.py

```
from behave import *
import sort

@given('the list is [3, -4, 5, 0, 1]')
def step_impl(context):
    context.gdata = [3, -4, 5, 0, 1]

@when('the list is sorted')
def step_impl(context):
    context.gdata = sort.sort_1(context.gdata)

@then('the new list is [5, -4, 3, 1, 0]')
def step_impl(context):
    assert context.gdata == [5, -4, 3, 1, 0]
```

features/steps/test2.py

```
from behave import *
import sort


@given('the list is [3, -4, 4, 5, 0, 1, -1, 17]')
def step_impl(context):
    context.gdata = [3, -4, 4, 5, 0, 1, -1, 17]


@when('the list is sorted2')
def step_impl(context):
    context.gdata = sort.sort_1(context.gdata)

@then('the new list is [17, 5, -4, 4, 3, 1, -1, 0]')
def step_impl(context):
    assert context.gdata == [17, 5, -4, 4, 3, 1, -1, 0]
```

features/steps/test3.py

```
from behave import *
import sort
```

```python
@given('the list is [0, -100, 100, 67, -67, 67, 99, 15, 16, -15]')
def step_impl(context):
    context.gdata = [0, -100, 100, 67, -67, 67, 99, 15, 16, -15]


@when('the list is sorted3')
def step_impl(context):
    context.gdata = sort.sort_1(context.gdata)

@then('the new list is [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0]')
def step_impl(context):
    assert context.gdata == [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0]
```

Примеры выполнения:

```
PS C:\Users\sophi\Documents\Python\lab_5> & C:/Users/sophi/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/sophi/Documents/Python/lab_5/test_TDD.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\sophi\Documents\Python\lab_5>
```

```
PS C:\Users\sophi\Documents\Python\lab_5> C:\Users\sophi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\Scripts\behave
Feature: Abs_sorting # features/tutorial.feature:1

  Scenario: Sort Abs                    # features/tutorial.feature:3
    Given the list is [3, -4, 5, 0, 1]    # features/steps/test1.py:14
    When the list is sorted               # features/steps/test1.py:23
    Then the new list is [5, -4, 3, 1, 0] # features/steps/test1.py:30

  Scenario: Sort Abs2                             # features/tutorial.feature:8
    Given the list is [3, -4, 4, 5, 0, 1, -1, 17]    # features/steps/test2.py:5
    When the list is sorted2                         # features/steps/test2.py:10
    Then the new list is [17, 5, -4, 4, 3, 1, -1, 0] # features/steps/test2.py:14

  Scenario: Sort Abs3                                        # features/tutorial.feature:13
    Given the list is [0, -100, 100, 67, -67, 67, 99, 15, 16, -15]    # features/steps/test3.py:5
    When the list is sorted3                                 # features/steps/test3.py:10
    Then the new list is [-100, 100, 99, 67, -67, 67, 16, 15, -15, 0] # features/steps/test3.py:14

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.007s
```