

**Московский государственный технический  
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

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

Выполнил:  
студент группы ИУ5-32Б  
Казицин Алексей

Проверил:  
преподаватель кафедры  
Гапанюк Ю.Е.

Подпись и дата:

Подпись и дата:

Москва, 2022 г.

**Цель лабораторной работы:** изучение возможностей модульного тестирования в языке Python.

## **Требования к отчету:**

Отчет по лабораторной работе должен содержать:

1. титульный лист;
2. описание задания;
3. текст программы;
4. экранные формы с примерами выполнения программы.

## **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:

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()
```

Sort.py

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

def sort_1(data):
    result = sorted(data, key=abs, reverse=True)
    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()
```

## 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]
```

## 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]
```

## Test3.py

```
from behave import *
import sort

@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]
```

## 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]

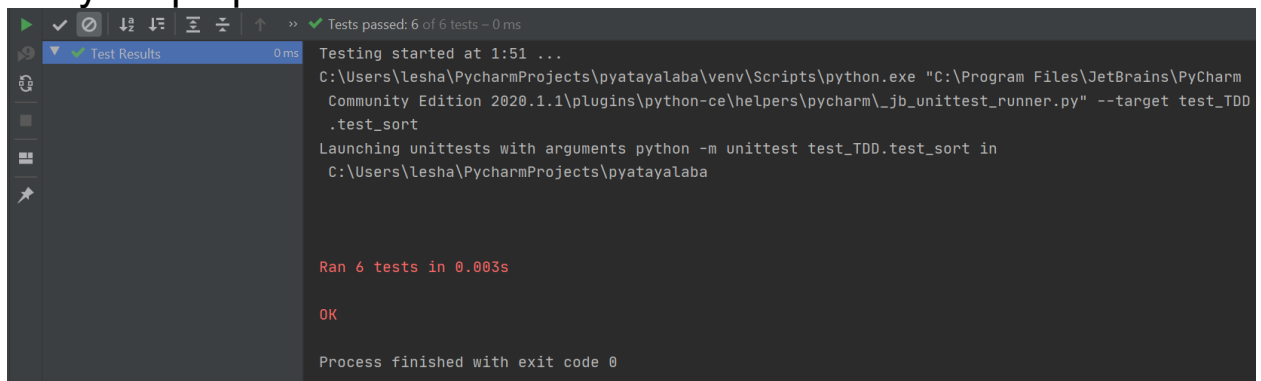
Scenario: Sort Abs3

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]

## Запуск программы:



The screenshot shows the PyCharm test runner interface. The top bar indicates 'Tests passed: 6 of 6 tests - 0 ms'. The 'Test Results' tab is active, showing a list of tests with a green checkmark and '0ms' duration. The main pane displays the test execution details: 'Testing started at 1:51 ...', the command used to run the tests, and the result 'Ran 6 tests in 0.003s' with an 'OK' status. The process finished with exit code 0.

```
(venv) C:\Users\lesha\PycharmProjects\pyatayalaba>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:5
  When the list is sorted # features/steps/test1.py:10
  Then the new list is [5, -4, 3, 1, 0] # features/steps/test1.py:15

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:15

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:15

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```