**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

«Тестирование в Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б: |  | преподаватель каф. ИУ5 |
| Семчёнок Владимир Дмитриевич |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022 г.

**Задание**

**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).

**Листинг**

sort.py

#data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

def abs\_sort(data):

return sorted(data, key=abs, reverse=True)

def abs\_sort\_lambda(data):

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

tdd\_test.py

import unittest

import sys

sys.path.append('../')

from sort import \*

class TestAbsSort(unittest.TestCase):

def test\_type(self):

self.assertIsInstance(abs\_sort([4, -30, 100, -100, 123, 1, 0, -1, -4]), list)

self.assertIsInstance(abs\_sort([4]), list)

def test\_default\_case(self):

self.assertEqual(abs\_sort([4, -30, 100, -100, 123, 1, 0, -1, -4]), [123, 100, -100, -30, 4, -4, 1, -1, 0])

self.assertEqual(abs\_sort([4]), [4])

self.assertEqual(abs\_sort([]), [])

class TestAbsLambdaSort(unittest.TestCase):

def test\_type(self):

self.assertIsInstance(abs\_sort\_lambda([4, -30, 100, -100, 123, 1, 0, -1, -4]), list)

self.assertIsInstance(abs\_sort\_lambda([4]), list)

def test\_default\_case(self):

self.assertEqual(abs\_sort\_lambda([4, -30, 100, -100, 123, 1, 0, -1, -4]), [123, 100, -100, -30, 4, -4, 1, -1, 0])

self.assertEqual(abs\_sort\_lambda([4]), [4])

self.assertEqual(abs\_sort\_lambda([]), [])

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

unittest.main()

bdd\_test.feature

Feature: Test Sort Function

Scenario: Abs Sort default

When I sort data [4, -30, 100, -100, 123, 1, 0, -1, -4]

Then I get result [123, 100, -100, -30, 4, -4, 1, -1, 0]

Scenario: Abs Sort one number

When I sort data [4]

Then I get result [4]

Scenario: Abs Sort with lambda default

When I lambda sort data [4, -30, 100, -100, 123, 1, 0, -1, -4]

Then I lambda get result [123, 100, -100, -30, 4, -4, 1, -1, 0]

Scenario: Abs Sort with lambda one number

When I lambda sort data [4]

Then I lambda get result [4]

test.py

from behave import given, when, then

import sys

sys.path.append('../')

from sort import \*

def get\_list(inp):

inp\_int = []

now\_str = ""

for i in inp:

if (i != ',' and i != ' ' and i != '[' and i != ']'):

now\_str += i

elif (i != ' ' and i != '['):

inp\_int.append(int(now\_str))

now\_str = ""

else:

now\_str = ""

return inp\_int

@when(u'I sort data {inp}')

def step\_impl(context, inp):

inp\_list = get\_list(inp)

print(u'STEP: When I sort data {}'.format(inp\_list))

context.result = abs\_sort(inp\_list)

@then(u'I get result {out}')

def step\_impl(context, out):

out\_list = get\_list(out)

print(u'STEP: Then I get result {}'.format(out))

assert context.result == out\_list, 'Expected {}, got {}'.format(out, context.result)

@when(u'I lambda sort data {inp}')

def step\_impl(context, inp):

inp\_list = get\_list(inp)

print(u'STEP: When I sort data {}'.format(inp\_list))

context.result = abs\_sort\_lambda(inp\_list)

@then(u'I lambda get result {out}')

def step\_impl(context, out):

out\_list = get\_list(out)

print(u'STEP: Then I get result {}'.format(out))

assert context.result == out\_list, 'Expected {}, got {}'.format(out, context.result)

Тесты:



