

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

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторным работам №5
«Модульное тестирование в Python»

Выполнил:

студент группы ИУ5-32Б
Васькин М.И.

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

дата: 16.12.2022

Москва, 2022 г.

Задание

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.

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).

Файл lab5.py

```
import sys
import math
import random
def get_coef(index, prompt):
    """
    Читаем коэффициент из командной строки или вводим с клавиатуры
    Args:
        index (int): Номер параметра в командной строке
    prompt (str): Приглашение для ввода коэффициента
    Returns:
        float: Коэффициент
    квадратного уравнения
    """
    try:
        # Пробуем прочесть коэффициент из командной
        строки
        coef_str = sys.argv[index]
        coef =
        ""
        while not isinstance(coef, float):
    try:
        coef = float(coef_str)
    except:
        print("Некорректный ввод данных! Введите
        заново!")
        coef = float(coef_str)
        except:
            # Вводим с клавиатуры
            coef_str = ""
            coef = ""
            while not isinstance(coef, float):
                try:
                    print(prompt)
                    coef_str = input()
                    coef = float(coef_str)
                except:
                    print("Некорректный ввод данных! Введите заново!")
```

```

    # Переводим строку в действительное число
    return coef

def get_roots(a, b, c):

    result = []
    D = b*b - 4*a*c
    if D == 0.0:
        if ((-b / (2.0*a)) >= 0):
            root1 = math.sqrt(-b / (2.0*a))
            root2 = -root1;
            if root1==0:
                result.append(root1)
        else:
            result.append(root1)
    result.append(root2)
    elif D > 0.0:
        sqD = math.sqrt(D)
        if
        (((-b + sqD) / (2.0*a)) >= 0):
            root1 = math.sqrt((-b + sqD) /
            (2.0*a))
            root2 = -root1
            if
            root1==0:
                result.append(root1)
        else:
            result.append(root1)
    result.append(root2)
    if (((-b -
    sqD) / (2.0*a)) >= 0):
        root3 = math.sqrt((-b - sqD) /
        (2.0*a))
        root4 = -root3
        if
        root3==0:
            result.append(root3)
    else:
        result.append(root3)
    result.append(root4)

    return result

def
sort_with_lambda(lst):
    return sorted(lst, key=Lambda x: abs(x), reverse=True)

```

```

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randint(begin, end)

def main():
    a = get_coef(1,
'Введите коэффициент A:')
    b =
get_coef(2, 'Введите коэффициент B:')
    c
= get_coef(3, 'Введите коэффициент C:')

    # Вычисление корней
    roots = get_roots(a,b,c)
    # Вывод корней
    len_roots = len(roots)
    if len_roots == 0:
        print('Нет корней')
    elif len_roots == 1:
        print('Один корень: {}'.format(roots[0]))
    elif len_roots == 2:
        print('Два корня: {} и {}'.format(roots[0], roots[1]))
    elif len_roots == 3:
        print('Три корня: {}, {} и {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
        print('Четыре корня: {} , {} , {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))

__name__ == "__main__":
    main()

```

TDD – фреймворк (unittest) Файл

TDD.py

```

import unittest.mock
from lab5 import get_roots, sort_with_lambda, gen_random
import sys

def compare_list(lst1,
lst2):
    return len(lst1) == len(lst2) and set(lst1) == set(lst2)

class
TestFuncs(unittest.TestCase):
def test_calculate(self):
    tests = [
        ([1, 1, -2], [1, -1]),
        ([9, 8, -1], [0.3333333333333333, -0.3333333333333333]),
        ([20, -1, -1], [0.5, -0.5]),

```

```

        ([1, -40, 144], [6, -6, 2, -2]),
        ([5, -4, 1], []),
        ([-4, 16, 0], [2, -2, 0]),
        ([1, -18, 81], [3, -3]),
        ([256, -32, 1], [0.25, -0.25]),
        ([4, 0, 0], [0]),
        ([1, 0.1, 0], [0]),
    ]
    for
test in tests:
    roots = get_roots(*test[0])
self.assertTrue(compare_list(roots, test[1]))
def
test_sort_with_lambda(self):
    tests = [
        ([3, -40, 40, 4, 5], [-40, 40, 5, 4, 3]),
        ([1, 5, 3, -4, 2], [5, -4, 3, 2, 1]),
    ]
    for
test in tests:
    sequence = sort_with_lambda(test[0])
self.assertEqual(sequence, test[1])

#sequence = [1,5,3,-4,2]
#self.assertEqual(sort_with_lambda(sequence), [5,-4, 3, 2, 1])
def
test_gen_random(self):
    sequence = list(gen_random(5,1,3))
self.assertEqual(len(sequence),5)
if __name__ ==
"__main__":
unittest.main()
# Пример выполнения кода
[Running] python -u "c:\BKIT\lab5\TDD.py"
...
-----
Ran 3 tests in 0.000s

OK

[Done] exited with code=0 in 0.766 seconds

```

BDD – фреймворк (Behave)

Файл tutorial.feature

Feature: Sorting

Scenario: Seq1

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: Seq2

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: Seq3

Given the list is [0, -100, 100, 67, 67, 99, 15, 16, -15]

When the list is sorted3

Then the new list is [-100, 100, 99, 67, 67, 16, 15, -15, 0]

Файл sort.py

```
data = [0, -100, 100, 67, -67, 67, 99, 15, 16, -15]
def sort_1(lst):    result = sorted(lst, key =
abs, reverse = True)
    #print(result)
return result
def
sort_2(lst):
    result_with_lambda = sorted(lst, key = lambda x: abs(x), reverse = True)
return result_with_lambda
def
main_s():
    result = sort_1(data)
print (result)
    result_with_lambda =
sort_2(data)
print(result_with_lambda)
    if __name__ ==
"__main__":
        main_s()
```

Файл 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]
```

Файл 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_2(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]
```


Файл test3.py

```
from behave import
* import sort

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

@when('the list is sorted3') def
step_impl(context):
    context.gdata = sort.sort_2(context.gdata)
@then('the new list is [-100, 100, 99, 67, 67, 16, 15, -15, 0]')
def step_impl(context):
    assert context.gdata == [-100, 100, 99, 67, 67, 16, 15, -15, 0]
```

Пример выполнения программы

(venv) PS C:\BKIT\lab5> behave

Feature: Sorting # features/tutorial.feature:1

Scenario: Seq1 # features/tutorial.feature:3

Given the list is [3, -4, 5, 0, 1] # features/steps/test1.py:4

When the list is sorted # features/steps/test1.py:8

Then the new list is [5, -4, 3, 1, 0] # features/steps/test1.py:13

Scenario: Seq2 # 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: Seq3 #

features/tutorial.feature:13

Given the list is [0, -100, 100, 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, 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 0m0.010s