



**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 №4

«Шаблоны проектирования и модульное тестирование в Python»

Выполнил: студент группы ИУ5-32Б  
Овчинников Данила Алексеевич

Москва, 2021 г.

## Описание задания:

- 1) Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
- 2) В модульных тестах необходимо применить следующие технологии:
  - a. TDD - фреймворк.
  - b. BDD - фреймворк.
  - c. Создание Моск-объектов.

## Текст программы:

Файл main.py:

```
import sys
import math
import os

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 прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
    except:
        # Вводим с клавиатуры
        print(prompt)
        coef_str = input()
        # Переводим строку в действительное число
    try:
        coef = float(coef_str)
    except ValueError:
        print('У вас ошибка, попробуйте ввести правильные данные(число)')
    return get_coef(index, prompt)

def get_roots(a, b, c):
    """
    Вычисление корней квадратного уравнения
    Args:
    """
```

```

        a (float): коэффициент A
b (float): коэффициент B          c
(float): коэффициент C
    Returns:          list[float]:
Список корней
    ...

result = []          if
a == 0:              if
b != 0:
    root = -c / b
if root == 0:
    result.append(root)
elif root > 0:
    result.append(-round(math.sqrt(root), 2))
result.append(round(math.sqrt(root), 2))          return
result          D = b*b -4*a*c          if D == 0.0:
    root = -b / (2.0*a)
if root == 0:
    result.append(0.0)
elif root > 0:
    result.append(-round(math.sqrt(root), 2))
result.append(round(math.sqrt(root), 2))          elif D >
0.0:
    sqD = math.sqrt(D)
root1 = (-b + sqD) / (2.0*a)
root2 = (-b - sqD) / (2.0*a)
if root1 == 0.0:
    result.append(root1)
elif root1 > 0.0:
    result.append(-round(math.sqrt(root1), 2))
result.append(round(math.sqrt(root1), 2))          if
root2 == 0.0:
    result.append(root2)
elif root2 > 0.0:
    result.append(-round(math.sqrt(root2), 2))
result.append(round(math.sqrt(root2), 2))          return
result
def
main():
    ...

    Основная функция    '''          a =
get_coef(1, 'Введите коэффициент A:')          b =
get_coef(2, 'Введите коэффициент B:')          c =
get_coef(3, 'Введите коэффициент C:')          if
a == 0 and b == 0 and c == 0:
    print('Бесконечное число корней')
else:
    # 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]))

```

```

# Если сценарий запущен из командной строки if
__name__ == "__main__":
    main()

```

```

# Пример запуска
# qr.py 1 0 -4

```

Файл test1.py:

```

import
unittest import
sys, os

sys.path.append(os.getcwd()) from
main import *
test1 = [0] test2 = [-
1.41, 1.41] test3 = [-
2.0, 2.0, 0.0]
class
TestGetRoots(unittest.TestCase):
    def test_get_roots1(self):
        self.assertEqual(get_roots(1, 0, 0), test1)
def test_get_roots2(self):
    self.assertEqual(get_roots(1, 0, -4), test2)
    def test_get_roots3(self):
        self.assertEqual(get_roots(1, -4, 0), test3)
if __name__ ==
"__main__":
    unittest.main()

```

Файл test2.py:

```
from behave import Given, When, Then from
main import get_roots

@Given(u'Calculator app is run') def
step_impl(context):
    pass

@When(u'I input "{a}", "{b}" and "{c}" to calculator') def
step_impl(context, a, b, c):
    context.result = str(get_roots(float(a), float(b), float(c)))

@Then(u'I get result "{out}"')
def step_impl(context, out):
    if (context.result == out):
        pass
    else :
        raise Exception ("Invalid roots")
```

Файл test2.feature:

```
Feature: Test

    Scenario:Test1
    Given Calculator app is run
    When I input "1", "0" and "0" to calculator
    Then I get result "[0.0]"

    Scenario:Test2
    Given Calculator app is run
    When I input "1", "0" and "-4" to calculator
    Then I get result "[-1.41, 1.41]"

    Scenario:Test3
    Given Calculator app is run
    When I input "1", "-4" and "0" to calculator
    Then I get result "[-2.0, 2.0, 0.0]"
```

## Примеры выполнения программы:

```
PS C:\Users\J4ngle\Desktop\lab4> & c:/Users/J4ngle/Desktop/lab4/.env/Scripts/Activate.ps1
(.venv) PS C:\Users\J4ngle\Desktop\lab4> & c:/Users/J4ngle/Desktop/lab4/.env/Scripts/python.exe c:/Users/J4ngle/Desktop/lab4/main.py
Введите коэффициент A:
1
Введите коэффициент B:
0
Введите коэффициент C:
0
Один корень: 0.0
```

## TDD:

```
PS C:\Users\J4ngle\Desktop\lab4> & c:/Users/J4ngle/Desktop/lab4/.venv/Scripts/Activate.ps1
(.venv) PS C:\Users\J4ngle\Desktop\lab4> & c:/Users/J4ngle/Desktop/lab4/.venv/Scripts/python.exe c:/Users/J4ngle/Desktop/lab4/test1.py
...
-----
Ran 3 tests in 0.000s
OK
```

## BDD:

```
(.venv) PS C:\Users\J4ngle\Desktop\BKIT_2021-main\lab4\features> behave test2.feature
Feature: Test # test2.feature:2

  Scenario: Test1 # test2.feature:4
    Given Calculator app is run # ../steps/test2.py:4
    When I input "1", "0" and "0" to calculator # ../steps/test2.py:8
    Then I get result "[0.0]" # ../steps/test2.py:12

  Scenario: Test2 # test2.feature:9
    Given Calculator app is run # ../steps/test2.py:4
    When I input "1", "0" and "-4" to calculator # ../steps/test2.py:8
    Then I get result "[-1.41, 1.41]" # ../steps/test2.py:12

  Scenario: Test3 # test2.feature:14
    Given Calculator app is run # ../steps/test2.py:4
    When I input "1", "-4" and "0" to calculator # ../steps/test2.py:8
    Then I get result "[-2.0, 2.0, 0.0]" # ../steps/test2.py:12

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.001s
```