3Московский государственный технический университет имени Н.Э. Баумана

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет РТ Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по лабораторной работе № 5 по курсу**

**Базовые компоненты**

"Модульное тестирование в Python"

6

 (количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель |  |  |
| студент группы РТ5-31Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Билалов А.К. |
|  |  | “16” декабря 2022 г. |
|  |  |  |
| Проверил |  |  |
| Доцент кафедры ИУ5 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Гапанюк Ю.Е. |
|  |  | “\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва 2022

# Оглавление

[Оглавление 2](#_Toc113796368)

[Описание задания 3](#_Toc113796369)

[Текст программы 4](#_Toc113796370)

[Результаты выполнения программы 6](#_Toc113796371)

# Описание задания

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

Задание:

Выбрать любой фрагмент кода из предыдущих лабораторных работ (выбрал 3-4).

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

Разработать модульные тесты с помощью TDD и BDD фреймворков.

# Текст программы

features/my.feature:

Feature: Showing off behave

Scenario: Run a simple test

Given a list of numbers

When we call Unique

Then we get a list of unique numbers

Given a list of strings

When we call Unique with ignore\_case=True

Then we get a list of unique strings

Given a list of dictionaries

When we call field with "age"

Then we get a list of ages

features/steps/bddtests.py:

from behave import given, when, then, step

import lab\_python\_fp.unique as unique

import lab\_python\_fp.field as field

@given('a list of numbers')

def step\_impl(context):

    context.list\_of\_numbers = [1, 2, 3, 1, 2, 3]

@when('we call Unique')

def step\_impl(context):

    context.unique = unique.Unique(context.list\_of\_numbers)

@then('we get a list of unique numbers')

def step\_impl(context):

    assert context.unique.items == [1, 2, 3]

@given('a list of strings')

def step\_impl(context):

    context.list\_of\_strings = ["a", "A", "B", "b"]

@when('we call Unique with ignore\_case=True')

def step\_impl(context):

    context.unique = unique.Unique(context.list\_of\_strings, ignore\_case=True)

@then('we get a list of unique strings')

def step\_impl(context):

    assert context.unique.items == ["a", "B"]

@given('a list of dictionaries')

def step\_impl(context):

    context.workers = [

        {'name': 'Иван', 'surname': 'Иванов', 'age': 25, 'position': 'водитель'},

        {'name': 'Петр', 'surname': 'Петров', 'age': 35, 'position': 'инженер'},

        {'name': 'Алексей', 'surname': 'Алексеев', 'age': 18, 'position': 'стажер'},

    ]

@when('we call field with "age"')

def step\_impl(context):

    context.result = field.field(context.workers, 'age')

@then('we get a list of ages')

def step\_impl(context):

    assert context.result == [25, 35, 18]

tests/tddtests.py:

import lab\_python\_fp.unique as unique

import lab\_python\_fp.field as field

def test\_unique():

    assert unique.Unique([1, 2, 3, 1, 2, 3]).items == [1, 2, 3]

    assert unique.Unique(["a", "A", "B", "b"], ignore\_case=True).items == ["a", "B"]

def test\_field():

    workers = [

        {'name': 'Иван', 'surname': 'Иванов', 'age': 25, 'position': 'водитель'},

        {'name': 'Петр', 'surname': 'Петров', 'age': 35, 'position': 'инженер'},

        {'name': 'Алексей', 'surname': 'Алексеев', 'age': 18, 'position': 'стажер'},

    ]

    assert field.field(workers, 'age') == [25, 35, 18]

    assert field.field(workers, 'name') == ['Иван', 'Петр', 'Алексей']

    assert field.field(workers, 'name', 'position') == [{'name': 'Иван', 'position': 'водитель'}, {'name': 'Петр', 'position': 'инженер'}, {'name': 'Алексей', 'position': 'стажер'}]

# Результаты выполнения программы



