|  |  |  |
| --- | --- | --- |
| Защищено:  Иниятуллин Р. М.    "\_\_" \_\_\_\_\_\_\_\_\_\_\_2022 г. |  | Демонстрация:  Иниятуллин Р. М.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

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

**базовые компоненты интернет-технологий (БКИТ)**

**Тема работы:**

## **" Модульное тестирование в Python"**

5

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-54Б Иниятуллин Р. М. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Гапанюк Ю.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Москва, МГТУ - 2022

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

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Цель лабораторной работы 2](#_Toc113807885)

[2. Описание задания. 2](#_Toc113807886)

[3. Листинг программы: 3](#_Toc113807887)

[4. Результаты работы программы: 5](#_Toc113807888)

# **Цель лабораторной работы**

Изучение возможностей модульного тестирования в языке Python.

# **Описание задания.**

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 программы:**

**tdd\_test.py**

# Подключаем библиотеку unitetest для тестирования  
import unittest  
  
from unique import Unique  
  
'''  
assertEqual(self, first, second)  
first - передаваемое значение  
second - полученное значение (в тело функции должен быть return, если вы там не оставили, тогда прописать здесь как None)  
если передаваемое значение совпадает с полученным значением, то тест пройден успешно  
'''  
  
  
class test\_unique(unittest.TestCase):  
 def test\_value(self):  
 data = [1, 1, 1, 2, 2, 2, 3, 3, 4]  
 arr\_unique = list(Unique(data))  
  
 self.assertEqual(arr\_unique, [1, 2, 3, 4])  
  
 def test\_val(self):  
 data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
 # Получаем уникальные значения и сохраним его в переменной  
 arr\_unique = list(Unique(data))  
 # Проверяем  
 self.assertEqual(  
 arr\_unique,  
 ['a', 'A', 'b', 'B']  
 )  
  
 def test\_val\_ignore\_case(self):  
 data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
 # Получаем уникальные значения и сохраним его в переменной  
 arr\_unique = list(Unique(data, ignore\_case=True))  
 # Проверяем  
 self.assertEqual(  
 arr\_unique,  
 ['a', 'b']  
 )  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**asd.feature**

*# Created by Dagnarys at 20.12.2022*Feature: Get only unique elements of list  
  
 Scenario Outline: Getting list of unique elements  
 Given some data **<list>** When data is getting unique and there is case **<case>** Then data is unique **<result>** Examples:  
 | **list** |**result** | **case** |  
 | **[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]** |**[1,2]** | **False**|  
  
  
 Scenario Outline: Getting list of unique letters  
 Given some data **<list>** When data is getting unique and there is case **<case>** Then data is unique **<result>** Examples:  
 | **list** | **result** | **case** |  
 | **['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']** | **['a', 'A', 'b', 'B']** | **False** |  
  
 Scenario Outline: Getting list of unique letters with lower case  
 Given some data **<list>** When data is getting unique and there is case **<case>** Then data is unique **<result>** Examples:  
 | **list** | **result** | **case** |  
 | **['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']** | **['a','b']** | **True** |

**bdd\_test.py**

from unique import Unique  
from pytest\_bdd import scenario, given, when, then  
import ast  
from behave import \*  
  
  
####################  
  
  
@given('some data {LIST}')  
def data(context, LIST):  
 context.LIST = list(ast.literal\_eval(LIST))  
 print(LIST)  
  
  
@when('data is getting unique and there is case {CASE}')  
def unique\_data(context, CASE):  
 if CASE == 'True':  
 uniq\_list = list(Unique(context.LIST, ignore\_case= True))  
 if CASE == 'False':  
 uniq\_list = list(Unique(context.LIST))  
 context.res = uniq\_list  
  
  
@then('data is unique {RESULT}')  
def res\_data(context, RESULT):  
 assert list(context.res) == ast.literal\_eval(RESULT)  
 print(context.res)

**mock.py**

from unittest import mock  
from unittest import TestCase  
  
class Test(TestCase):  
 @mock.patch('mock.sum',return\_value=8)  
 def test(self ,mock\_check\_output):  
 result = main()  
 expect\_result = 8  
 self.assertEqual(expect\_result,result)  
  
def sum(a,b):  
 return a+b  
  
def main():  
  
 s = sum(3, 5)  
 return s  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# **Результаты работы программы:**



