

Защищено:  
Папин А.В..

Демонстрация:  
Папин А.В..

"\_\_" \_\_\_\_\_ 2022 г.

"\_\_" \_\_\_\_\_ 2022 г.

## **Отчет по лабораторной работе №5 по курсу базовые компоненты интернет-технологий (БКИТ)**

**Тема работы: "Модульное тестирование в Python"**

9

(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-54Б  
Папин Алексей

Гапанюк Ю.Е.

\_\_\_\_\_  
(подпись)

"\_\_" \_\_\_\_\_ 2022 г.

## СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы.....	2
2. Описание задания. ....	2
3. Листинг программы: .....	3
3.1. Для unittest.....	3
3.2. unique.py .....	3
3.2.1.test.py .....	4
3.3. Для Behave .....	5
3.3.1.check_unique.feature.....	5
4.1.1.Unique.py .....	6
5. Результаты работы программы:.....	7
5.1. В IDE JetBrains PyCharm.....	7
5.1.1.Unittest .....	7
5.1.2.Behave .....	7
5.2. Через cmd / powershell.....	8
5.2.1.Unittest .....	8
5.2.2.Behave .....	8

## **1. Цель лабораторной работы**

Изучение возможностей возможностей модульного тестирования в языке Python.

## **2. Описание задания.**

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

1. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
2. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
  - TDD - фреймворк (не менее 3 тестов).
  - BDD - фреймворк (не менее 3 тестов).
  - Создание Моск-объектов (необязательное дополнительное задание).

### 3. Листинг программы:

#### 3.1.Для unittest

#### 3.2.unique.py

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор
        # должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться
        # одинаковыми строки в разном регистре
        # Например: ignore_case = True, АБв и АБВ - разные строки
        # ignore_case = False, АБв и АБВ - одинаковые строки,
        # одна из которых удалится
        # По-умолчанию ignore_case = False

        self.arr = []

        # используя кортежи, получаем ключ и значения
        for key, value in kwargs.items():
            # если ключ пустой и значение ИСТИНА, то
            if key == 'ignore_case' and value == True:
                # в текущем списке все символы преобразуем в нижний регистр
                # через функции lower
                items = [i.lower() for i in items]

        for index in items:
            # Если текущее значение с списка item не совпадает / не
            # существует в созданном списке arr
            if index not in self.arr:
                # то присвоим несуществующее значение в созданном списке arr
                self.arr.append(index)
            pass

    def __next__(self):
        try:
            x = self.arr[self.begin]
            self.begin += 1
            return x
        except:
            raise StopIteration

    def __iter__(self):
        self.begin = 0
        return self
```

### 3.2.1. test.py

```
# Подключаем библиотеку unittest для тестирования
import unittest
import math

'''
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
'''

from function.unique import Unique

class test_unique(unittest.TestCase):
    # Проверка на чисел
    def test_value(self):
        # Дан список с числами
        data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            [1, 2]
        )

    # Проверка на буквы
    def test_letters(self):
        # Дан список с числами
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            ['a', 'A', 'b', 'B']
        )

    # Проверка на буквы без чувствительного регистра
    def test_letters_ignore_case(self):
        # Дан список с числами
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data, ignore_case = True).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            ['a', 'b']
        )
```

```
if __name__ == '__main__':
    unittest.main()
```

### 3.3.Для Behave

#### 3.3.1. check unique.feature

4. **Feature:** Calculating and getting unique values

```
# Уникальные значения числового типа
# If <CASE> is 1 then is True
```

**Scenario Outline:** We get unique values from the list of the contained number

```
Given I have a class of unique values
And Getting the list: <list>
When Finding unique values, case: <CASE>
Then Output unique values: <unique>
```

**Examples:**

list	unique	CASE
[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]	[1, 2]	0
[1, 3, 1, 1, 1, 3, 2, 2, 2, 2]	[1, 3, 2]	0

```
# Уникальные значения символьного типа
```

**Scenario Outline:** We get unique values from the list of the contained char

```
Given I have a class of unique values
And Getting the list: <list>
When Finding unique values, case: <CASE>
Then Output unique values: <unique>
```

**Examples:**

CASE	list	unique
0	['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']	['a', 'A', 'b', 'B']
0	['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B', 'c', 'A']	['a', 'C', 'b', 'B', 'c', 'A']

```
# Уникальные значения символьного типа без чувствительного регистра
```

**Scenario Outline:** We get unique values from the list of the contained char ignore\_case

```
Given I have a class of unique values
And Getting the list: <list>
When Finding unique values, case: <CASE>
Then Output unique values: <unique>
```

**Examples:**

list	unique	CASE
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']	['a', 'b']	1

```

|
|   | ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B'] | ['a', 'c', 'b'] | 1
|
|
|
|   # Уникальные значения смешанного типа
|   Scenario Outline: We get unique values from the list of the contained
|   all type
|       Given I have a class of unique values
|       And Getting the list: <list>
|       When Finding unique values, case: <CASE>
|       Then Output unique values: <unique>
|
|   Examples:
|       | list | unique
|   CASE |
|       | ['a', 'A', 'b', 'B', '1', '1', '2', '2'] | ['a', 'A', 'b', 'B',
|   '1', '2'] | 0
|       | ['a', 'A', 'b', 'B', '1', '1', '2', '2'] | ['a', 'b', '1', '2']
|   1

```

#### 4.1.1. Unique.py

```

from behave import Given, When, Then
from function.unique import Unique
import ast

@Given('I have a class of unique values')
def step_impl(context):
    pass

@Given("Getting the list: {LIST}")
def given_increment(context, LIST):
    context.LIST = list(ast.literal_eval(LIST))
    print(f'Список: {LIST}')

@When("Finding unique values, case: {CASE}")
def given_increment(context, CASE):
    check = bool(int(CASE))
    if (check == True):
        unique_list = Unique(context.LIST, ignore_case=check)
    else:
        unique_list = Unique(context.LIST)

    context.results = unique_list
    # print(f'Уникальные значения: {unique_list}')

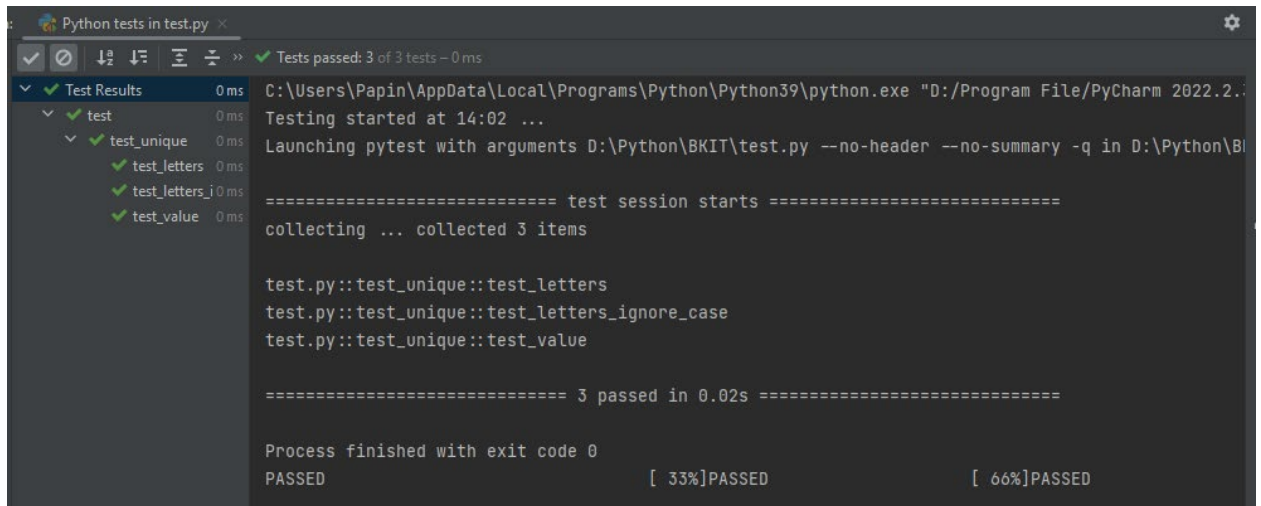
@Then("Output unique values: {UNIQUE}")
def then_results(context, UNIQUE):
    assert context.results.arr == ast.literal_eval(UNIQUE)
    print(f'Уникальные значения: {context.results.arr}')

```

## 5. Результаты работы программы:

### 5.1.B IDE JetBrains PyCharm

#### 5.1.1. Unittest



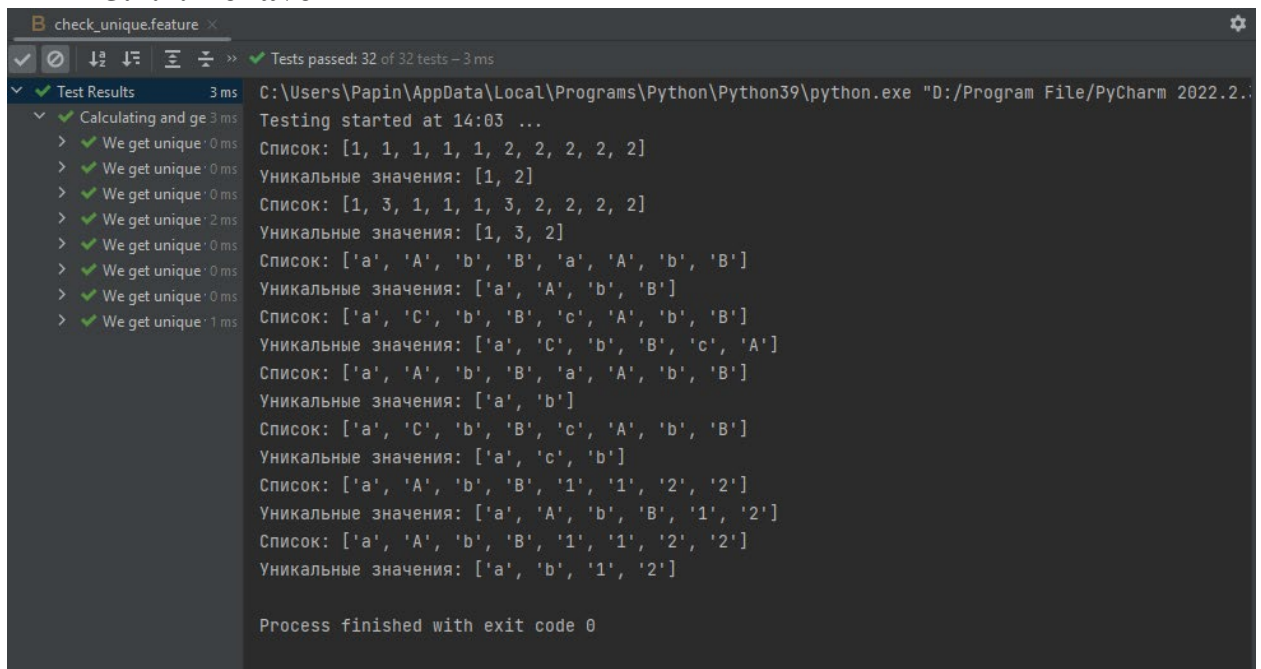
```
Python tests in test.py x
✓ Tests passed: 3 of 3 tests - 0 ms
Test Results 0 ms
  ✓ test 0 ms
    ✓ test_unique 0 ms
    ✓ test_letters 0 ms
    ✓ test_letters_i 0 ms
    ✓ test_value 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.1/PyCharm 2022.2.1.exe"
Testing started at 14:02 ...
Launching pytest with arguments D:\Python\BKIT\test.py --no-header --no-summary -q in D:\Python\BKIT
===== test session starts =====
collecting ... collected 3 items

test.py::test_unique::test_letters
test.py::test_unique::test_letters_ignore_case
test.py::test_unique::test_value

===== 3 passed in 0.02s =====

Process finished with exit code 0
PASSED [ 33%]PASSED [ 66%]PASSED
```

#### 5.1.2. Behave



```
B check_unique.feature x
✓ Tests passed: 32 of 32 tests - 3 ms
Test Results 3 ms
  ✓ Calculating and getting unique values 3 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 2 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 0 ms
    > ✓ We get unique: 1 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.1/PyCharm 2022.2.1.exe"
Testing started at 14:03 ...
Список: [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
Уникальные значения: [1, 2]
Список: [1, 3, 1, 1, 1, 3, 2, 2, 2, 2]
Уникальные значения: [1, 3, 2]
Список: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
Уникальные значения: ['a', 'A', 'b', 'B']
Список: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']
Уникальные значения: ['a', 'C', 'b', 'B', 'c', 'A']
Список: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
Уникальные значения: ['a', 'b']
Список: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']
Уникальные значения: ['a', 'c', 'b']
Список: ['a', 'A', 'b', 'B', '1', '1', '2', '2']
Уникальные значения: ['a', 'A', 'b', 'B', '1', '2']
Список: ['a', 'A', 'b', 'B', '1', '1', '2', '2']
Уникальные значения: ['a', 'b', '1', '2']

Process finished with exit code 0
```



## 5.2. Чепез cmd / powershell

### 5.2.1. Unittest

```
Windows PowerShell
PS D:\Python\BKIT> python test.py
...
-----
Ran 3 tests in 0.001s

OK
PS D:\Python\BKIT>
```

### 5.2.2. Behave

```
Windows PowerShell
PS D:\Python\BKIT> behave feature/check_unique.feature
Feature: Calculating and getting unique values # feature/check_unique.feature:1

Scenario Outline: We get unique values from the list of the contained number -- @1.1 # feature/check_unique.feature:13
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: [1, 1, 1, 1, 2, 2, 2, 2] # feature/steps/Unique.py:11
  When Finding unique values, case: 0 # feature/steps/Unique.py:17
  Then Output unique values: [1, 2] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained number -- @1.2 # feature/check_unique.feature:14
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: [1, 3, 1, 1, 3, 2, 2, 2, 2] # feature/steps/Unique.py:11
  When Finding unique values, case: 0 # feature/steps/Unique.py:17
  Then Output unique values: [1, 3, 2] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained char -- @1.1 # feature/check_unique.feature:26
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B'] # feature/steps/Unique.py:11
  When Finding unique values, case: 0 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'A', 'b', 'B'] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained char -- @1.2 # feature/check_unique.feature:27
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B'] # feature/steps/Unique.py:11
  When Finding unique values, case: 0 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'C', 'b', 'B', 'c', 'A'] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained char ignore_case -- @1.1 # feature/check_unique.feature:39
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B'] # feature/steps/Unique.py:11
  When Finding unique values, case: 1 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'b'] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained char ignore_case -- @1.2 # feature/check_unique.feature:40
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B'] # feature/steps/Unique.py:11
  When Finding unique values, case: 1 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'c', 'b'] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained all type -- @1.1 # feature/check_unique.feature:52
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'A', 'b', 'B', '1', '1', '2', '2'] # feature/steps/Unique.py:11
  When Finding unique values, case: 0 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'A', 'b', 'B', '1', '2'] # feature/steps/Unique.py:29

Scenario Outline: We get unique values from the list of the contained all type -- @1.2 # feature/check_unique.feature:53
  Given I have a class of unique values # feature/steps/Unique.py:6
  And Getting the list: ['a', 'A', 'b', 'B', '1', '1', '2', '2'] # feature/steps/Unique.py:11
  When Finding unique values, case: 1 # feature/steps/Unique.py:17
  Then Output unique values: ['a', 'b', '1', '2'] # feature/steps/Unique.py:29

1 feature passed, 0 failed, 0 skipped
8 scenarios passed, 0 failed, 0 skipped
32 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.005s
PS D:\Python\BKIT>
```