



**Министерство науки и высшего образования
Российской Федерации Федеральное государственное
бюджетное образовательное учреждение высшего
образования**

**«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика, искусственный интеллект и
системы управления»**

**Кафедра «Системы обработки информации и
управления»**

**Лабораторная работа №5 по курсу
«Базовые компоненты интернет-технологий»**

Выполнил:
студент группы ИУ5-33Б
Иванченко Максим

Проверил:
Доцент кафедры ИУ5
Гапанюк Юрий Евгеньевич

2022 г.

Постановка задачи

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:
 - a. TDD - фреймворк (не менее 3 тестов).
 - b. BDD - фреймворк (не менее 3 тестов).
 - c. Создание Mock-объектов (необязательное дополнительное задание).

Текст программы

TDD-тестирование

*Функция field(items, *args) из лабораторной работы №3-4*

```
def field(items, *args):
    assert len(args) > 0
    for dictionary in items:
        res_dict = {}
        for key in args:
            if dictionary.get(key) is not None:
                res_dict[key] = dictionary[key]
        if len(res_dict) == 0:
            pass
        elif len(res_dict) == 1:
            key = [key for key in res_dict.keys()][0]
            yield "" + str(res_dict[key]) + ""
        else:
            yield res_dict
```

Тестирование unittest

```
import unittest
from main import field
```

```
class MyTestCase(unittest.TestCase):
    def test_single_arg(self):
        """
        Check strings return for single arg
        """
        goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title':
'Диван для отдыха', 'price': 5300,
                                                                    'color':
'black'}]
        self.assertEqual([elem for elem in field(goods, 'title')], ["Ковер",
"Диван для отдыха"])

    def test_some_args(self):
        """
        Check dictionary return for some args
        """
        goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title':
'Диван для отдыха', 'price': 5300,
                                                                    'color':
'black'}]
        self.assertEqual([elem for elem in field(goods, 'title', 'price',
'color')], [{'title': 'Ковер',
'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price':
5300, 'color': 'black'}])

    def test_with_None_fields(self):
        """
        Check absence of None field
        """
        goods = [
            {'title': 'Ковер', 'price': None, 'color': 'green'},
            {'title': 'Диван для отдыха', 'price': 5300, 'color': None}
        ]
        self.assertEqual([elem for elem in field(goods, 'title', 'price',
'color')], [{'title': 'Ковер',
'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300}])
```

```

def test_all_None(self):
    """
    Check return nothing when all fields are None
    """
    goods = [{'title': None, 'price': None}, {'title': None, 'price': None}]
    self.assertEqual([elem for elem in field(goods, 'title', 'price')], [])

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

```

BDD-тестирование

Итератор Unique(object) из лабораторной работы №3-4

```

class Unique(object):
    def __init__(self, items, **kwargs):
        self.used = set()
        self.items = set(items)
        if len(kwargs) != 0:
            for name, value in kwargs.items():
                if name == 'ignore_case':
                    self.ignore_case = value
            else:
                self.ignore_case = False
        if self.ignore_case is True:
            to_change = set()
            for elem in self.items:
                if isinstance(elem, str):
                    if elem.lower() != elem:
                        to_change.add(elem)
            for elem in to_change:
                self.items.remove(elem)
                self.items.add(elem.lower())

    def __next__(self):
        if len(self.used) == len(self.items):
            raise StopIteration
        for elem in self.items:
            if elem not in self.used:
                self.used.add(elem)
                return elem

    def __iter__(self):
        return self

```

Тестирование behave

Файл test_bdd.feature (описание сценариев)

Feature: Unique() iterator work

Scenario: List of Integers

Given we have list of integers [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

When we run Unique() iterator

Then we get values without repetition [1, 2]

Scenario: Generator of Integers

Given we have generator [1, 1, 3, 1, 1, 2, 2, 3, 2, 1]

When we run Unique() iterator

Then we get values without repetition [1, 2, 3]

Scenario: List of Strings

Given we have list of strings [a, A, b, B, a, A, b, B]

When we run Unique() iterator
Then we get symbols without repetition [a, b, A, B]
Scenario: List of Strings with ignore_case
Given we have list of strings [a, A, b, B, a, A, b, B]
When we run Unique() iterator with ignore_case
Then we get symbols without repetition [a, b]

Файл test_steps.py

```
from behave import given, when, then, step
from main2 import Unique

@given('we have list of integers [{arr}]')
def step(context, arr):
    context.data = [int(elem) for elem in arr.split(', ')]

@given('we have generator [{gen}]')
def step(context, gen):
    context.data = (int(elem) for elem in gen.split(', '))

@given('we have list of strings [{arr}]')
def step(context, arr):
    context.data = arr.split(', ')

@when('we run Unique() iterator')
def step(context):
    context.res = list(Unique(context.data))

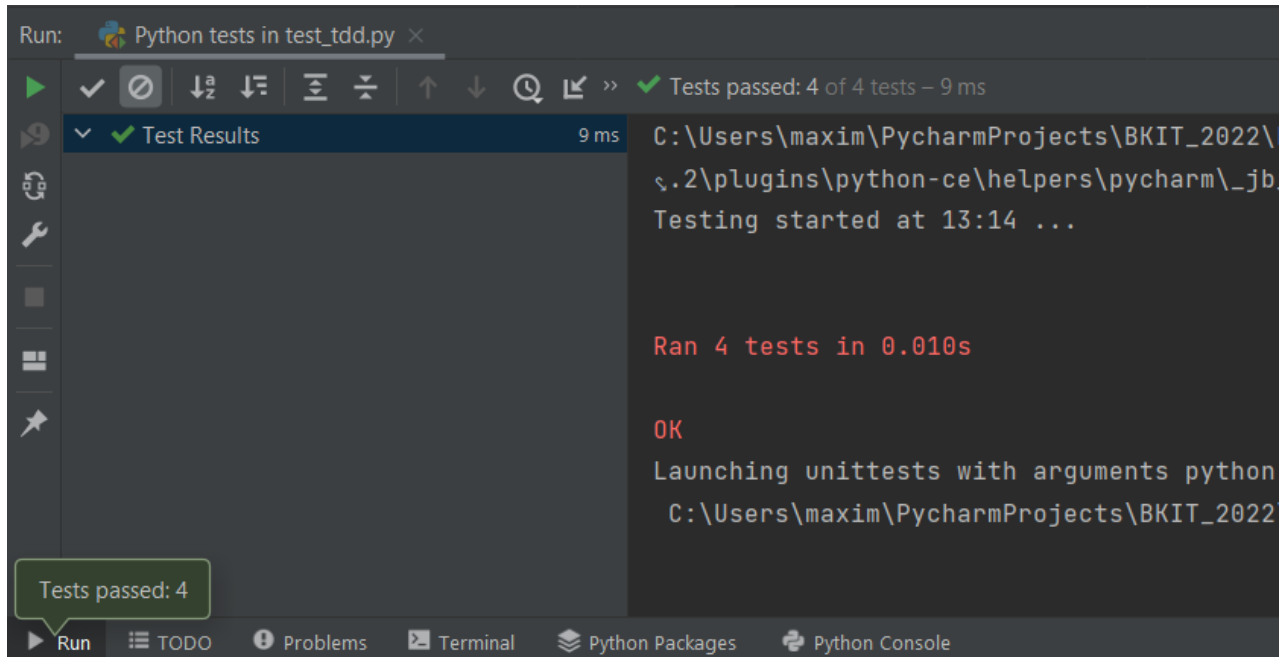
@when('we run Unique() iterator with ignore_case')
def step(context):
    context.res = list(Unique(context.data, ignore_case=True))

@then('we get values without repetition [{arr}]')
def step(context, arr):
    assert context.res == [int(elem) for elem in arr.split(', ')]

@then('we get symbols without repetition [{arr}]')
def step(context, arr):
    assert sorted(context.res) == sorted(arr.split(', '))
```

Результаты выполнения

TDD-тестирование



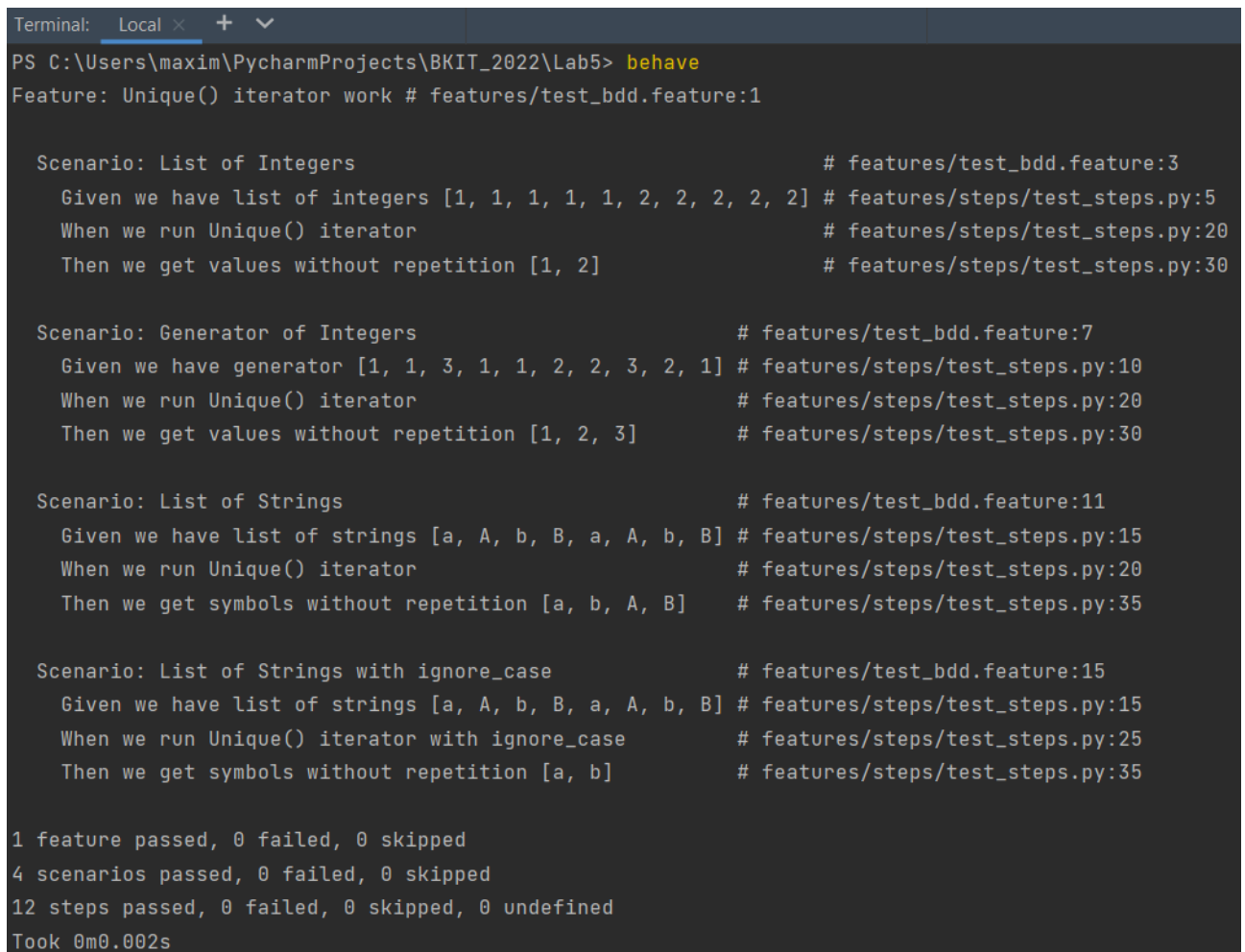
The screenshot shows the PyCharm Run window for a Python test. The top bar indicates 'Tests passed: 4 of 4 tests - 9 ms'. The main panel shows the test results, including the path to the test file and the command used to launch unittests. A green callout box at the bottom left states 'Tests passed: 4'.

```
Run: Python tests in test_tdd.py x
Tests passed: 4 of 4 tests - 9 ms
Test Results 9 ms
C:\Users\maxim\PycharmProjects\BKIT_2022\
s.2\plugins\python-ce\helpers\pycharm\_jb
Testing started at 13:14 ...

Ran 4 tests in 0.010s

OK
Launching unittests with arguments python
C:\Users\maxim\PycharmProjects\BKIT_2022
Tests passed: 4
Run TODO Problems Terminal Python Packages Python Console
```

BDD-тестирование



The screenshot shows a terminal window with the output of a behave test run. It details four scenarios: 'List of Integers', 'Generator of Integers', 'List of Strings', and 'List of Strings with ignore_case'. Each scenario includes given, when, and then steps with corresponding file paths. The final summary shows that all tests passed.

```
Terminal: Local x + v
PS C:\Users\maxim\PycharmProjects\BKIT_2022\Lab5> behave
Feature: Unique() iterator work # features/test_bdd.feature:1

Scenario: List of Integers # features/test_bdd.feature:3
  Given we have list of integers [1, 1, 1, 1, 1, 2, 2, 2, 2, 2] # features/steps/test_steps.py:5
  When we run Unique() iterator # features/steps/test_steps.py:20
  Then we get values without repetition [1, 2] # features/steps/test_steps.py:30

Scenario: Generator of Integers # features/test_bdd.feature:7
  Given we have generator [1, 1, 3, 1, 1, 2, 2, 3, 2, 1] # features/steps/test_steps.py:10
  When we run Unique() iterator # features/steps/test_steps.py:20
  Then we get values without repetition [1, 2, 3] # features/steps/test_steps.py:30

Scenario: List of Strings # features/test_bdd.feature:11
  Given we have list of strings [a, A, b, B, a, A, b, B] # features/steps/test_steps.py:15
  When we run Unique() iterator # features/steps/test_steps.py:20
  Then we get symbols without repetition [a, b, A, B] # features/steps/test_steps.py:35

Scenario: List of Strings with ignore_case # features/test_bdd.feature:15
  Given we have list of strings [a, A, b, B, a, A, b, B] # features/steps/test_steps.py:15
  When we run Unique() iterator with ignore_case # features/steps/test_steps.py:25
  Then we get symbols without repetition [a, b] # features/steps/test_steps.py:35

1 feature passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
12 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.002s
```