**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 №5

“Модульное тестирование в Python”

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34Б: |  | преподаватель каф. ИУ5 |
| Стукалов Иван Дмитриевич |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

2022

Описание задания:

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

Модифицируйте код таким образом, чтобы он был пригоден для модульного

тестирования.

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).

Текст программы:

TDD-тестирование:

// файл tdd\_tests.py

import unittest  
from lab\_python\_fp.field import field  
from lab\_python\_fp.unique import Unique  
from lab\_python\_fp.sort import sortedList  
  
  
class TestingLab(unittest.TestCase):  
  
 def test\_field(self):  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'}  
 ]  
 self.assertEqual(field(goods, 'title'), [{'title': 'Ковер'}, {'title': 'Диван для отдыха'}])  
 self.assertEqual(field(goods, 'title', 'price'),  
 [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}])  
  
 def test\_unique(self):  
 count\_list = ["aaa", "b", "c", "b", "A", "b", "AAA", "Aaa", "a"]  
 unique = Unique(count\_list)  
 unique\_it = Unique.\_\_iter\_\_(unique)  
 self.assertEqual(Unique.\_\_next\_\_(unique\_it), "aaa")  
 self.assertEqual(Unique.\_\_next\_\_(unique\_it), "b")  
 self.assertEqual(Unique.\_\_next\_\_(unique\_it), "c")  
  
 def test\_sort(self):  
 list\_arr = [-3, 6, -2, 5, -4, 3, -10, -53, 8]  
 self.assertEqual(sortedList(list\_arr), [8, 6, 5, 3, -2, -3, -4, -10, -53])  
 self.assertEqual(sortedList(list\_arr, lambda x: abs(x)), [-53, -10, 8, 6, 5, -4, 3, -3, -2])

BDD-тестирование:

//файл bdd\_field.feature

*# content for bdd\_field.feature* Feature: Filtering fields  
  
 Scenario: Connecting field  
 Given I have a data  
  
 When func works with 2 args  
 And func works with 3 args  
  
 Then I have filtered fields from dict with 2 args  
 And I have filtered fields from dict with 3 args

// файл bdd\_sort.feature

*# content for sort* Feature: Sorting the array  
  
 Scenario: Connecting sorting  
  
 Given Array for sorting  
  
 When sorting works no lambda  
 And sorting works with lambda  
  
 Then I have sorted arr no lambda  
 And I have sorted arr with lambda

//файл bdd\_unique.feature

*# content for sort* Feature: Sorting the array  
  
 Scenario: Connecting sorting  
  
 Given Array for sorting  
  
 When sorting works no lambda  
 And sorting works with lambda  
  
 Then I have sorted arr no lambda  
 And I have sorted arr with lambda

// файл bdd\_test\_field.py

from pytest\_bdd import scenario, given, when, then  
from lab\_python\_fp.field import field  
  
  
@scenario('bdd\_tests/bdd\_field.feature', 'Connecting field')  
def test\_field\_1():  
 pass  
  
  
@given("I have a data", target\_fixture="data")  
def data():  
 return [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'}  
 ]  
  
  
@when("func works with 2 args", target\_fixture="result\_1")  
def func\_1(data):  
 return field(data, 'title')  
  
  
@when("func works with 3 args", target\_fixture="result\_2")  
def func\_2(data):  
 return field(data, 'title', 'price')  
  
  
@then("I have filtered fields from dict with 2 args")  
def res\_1(result\_1):  
 assert result\_1 == [{'title': 'Ковер'}, {'title': 'Диван для отдыха'}]  
  
  
@then("I have filtered fields from dict with 3 args")  
def res\_2(result\_2):  
 assert result\_2 == [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]

// файл bdd\_test\_sort.py

from pytest\_bdd import scenario, given, when, then  
from lab\_python\_fp.sort import sortedList  
  
  
@scenario('bdd\_tests/bdd\_sort.feature', 'Connecting sorting')  
def test\_sort():  
 pass  
  
  
@given('Array for sorting', target\_fixture="data")  
def get\_data():  
 return [-3, 6, -2, 5, -4, 3, -10, -53, 8]  
  
  
@when('sorting works no lambda', target\_fixture='result\_1')  
def func\_1(data):  
 data\_to\_no\_lambda = data  
 return sortedList(data\_to\_no\_lambda)  
  
  
@when('sorting works with lambda', target\_fixture='result\_2')  
def func\_2(data):  
 data\_to\_lambda = data.copy()  
 return sortedList(data\_to\_lambda, lambda x: abs(x))  
  
  
@then('I have sorted arr no lambda')  
def res\_1(result\_1):  
 assert result\_1 == [8, 6, 5, 3, -2, -3, -4, -10, -53]  
  
  
@then('I have sorted arr with lambda')  
def res\_2(result\_2):  
 assert result\_2 == [-53, -10, 8, 6, 5, -4, 3, -3, -2]

// файл bdd\_test\_unique.py

from pytest\_bdd import scenario, given, when, then  
from lab\_python\_fp.unique import Unique  
  
  
@scenario('bdd\_tests/bdd\_unique.feature', 'Connecting generator')  
def test\_unique():  
 pass  
  
  
@given("Some args", target\_fixture="data")  
def get\_data():  
 count\_list = ["aaa", "b", "c", "b", "A", "b", "AAA", "Aaa", "a"]  
 unique = Unique(count\_list)  
 unique\_it = Unique.\_\_iter\_\_(unique)  
 return unique\_it  
  
  
@when("func works first time", target\_fixture="result\_1")  
def func\_1(data):  
 return Unique.\_\_next\_\_(data)  
  
  
@when("func works second time", target\_fixture="result\_2")  
def func\_2(data):  
 return Unique.\_\_next\_\_(data)  
  
  
@when("func works third time", target\_fixture="result\_3")  
def func\_3(data):  
 return Unique.\_\_next\_\_(data)  
  
  
@then("I have unique arr first time")  
def res\_1(result\_1):  
 assert result\_1 == "aaa"  
  
  
@then("I have unique arr second time")  
def res\_2(result\_2):  
 assert result\_2 == "b"  
  
  
@then("I have unique arr third time")  
def res\_3(result\_3):  
 assert result\_3 == "c"

Результаты работы программы:





