**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 |
| Угрюмов Михаил |  | Гапанюк Ю. Е. |
|  |  |  |

2022 г.

**Задание:**

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).

**Текст программы.**

**sort.py**

def my\_sort\_with\_lambda(data):  
 return sorted(data, key = lambda x: abs(x))  
  
def my\_sort\_without\_lambda(data):  
 return sorted(data, key = abs)

**field.py**

def field(items, \*args):  
 assert len(args) > 0  
  
 if len(args) == 1:  
 answer = []  
 for i in range(len(items)):  
 answer.append(items[i].get(args[0]))  
 else:  
 answer = [{} for i in range(len(items))]  
 for i in range(len(items)):  
 for key in args:  
 curr = items[i].get(key)  
 if curr is not None:  
 answer[i][key] = curr  
  
 return answer

**unique.py**

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.cnt = -1  
 if len(kwargs) == 0:  
 self.ignore\_case = False  
 else:  
 value = kwargs['ignore\_case']  
 self.ignore\_case = value  
 self.items = []  
 for elem in items:  
 if isinstance(elem, str):  
 if self.ignore\_case == True:  
 elem\_ = elem.lower()  
 if elem\_ not in self.items:  
 self.items.append(elem\_)  
 else:  
 if elem not in self.items:  
 self.items.append(elem)  
 else:  
 if elem not in self.items:  
 self.items.append(elem)  
  
 def \_\_next\_\_(self):  
 if self.cnt < len(self.items) - 1:  
 self.cnt += 1  
 return self.items[self.cnt]  
 else:  
 raise StopIteration  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_repr\_\_(self):  
 return str(self.items)

**TDD\_sort.py**

from functions.sort import my\_sort\_with\_lambda, my\_sort\_without\_lambda  
import unittest as ut  
  
class TestSort(ut.TestCase):  
 def setUp(self):  
 self.data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]  
 self.answer = [0, 1, -1, 4, -4, -30, 30, 100, -100, 123]  
  
 def test\_1(self):  
 self.assertEqual(my\_sort\_with\_lambda(self.data), self.answer)  
  
 def test\_2(self):  
 self.assertEqual(my\_sort\_without\_lambda(self.data), self.answer)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ut.main()

**TDD\_field.py**

from functions.field import field  
import unittest as ut  
  
class TestField(ut.TestCase):  
 def setUp(self):  
 self.goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 self.answer\_1 = ['Ковер', 'Диван для отдыха']  
 self.answer\_2 = [  
 {'title': 'Ковер', 'price': 2000},  
 {'title': 'Диван для отдыха', 'price': 5300}  
 ]  
  
 def test\_1(self):  
 self.assertEqual(field(self.goods, 'title'), self.answer\_1)  
  
 def test\_2(self):  
 self.assertEqual(field(self.goods, 'title', 'price'), self.answer\_2)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ut.main()

**TDD\_unique.py**

from functions.unique import Unique  
import unittest as ut  
  
class TestUnique(ut.TestCase):  
 def setUp(self):  
 self.symbols = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
 self.numbers = [1, 2, 2, 1, 1, 3, 2, 3, 5, 4, 1]  
 self.answer\_1 = [1, 2, 3, 4, 5]  
 self.answer\_2 = ['a', 'A', 'b', 'B']  
 self.answer\_3 = ['a', 'b']  
  
 def test\_1(self):  
 self.assertEqual(set(Unique(self.numbers)), set(self.answer\_1))  
  
 def test\_2(self):  
 self.assertEqual(set(Unique(self.symbols)), set(self.answer\_2))  
  
 def test\_3(self):  
 self.assertEqual(set(Unique(self.symbols, ignore\_case=True)), set(self.answer\_3))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ut.main()

**my\_sort\_with\_lambda.feature**

Feature: sorting by ABS with lambda  
Scenario: sorting by ABS with lambda  
Given data  
When sorting data: my\_sort\_with\_lambda  
Then data was sorted

**my\_sort\_without\_lambda.feature**

Feature: sorting by ABS without lambda  
Scenario: sorting by ABS without lambda  
Given data  
When sorting data: my\_sort\_without\_lambda  
Then data was sorted

**BDD\_sort.py**

from functions.sort import my\_sort\_with\_lambda, my\_sort\_without\_lambda  
from pytest\_bdd import scenario, given, when, then  
  
@scenario('features\\my\_sort\_with\_lambda.feature','sorting by ABS with lambda')  
def testing\_my\_sort\_with\_lambda():  
 pass  
  
@scenario('features\\my\_sort\_without\_lambda.feature','sorting by ABS without lambda')  
def testing\_my\_sort\_without\_lambda():  
 pass  
  
@given('data', target\_fixture='data')  
def data():  
 return [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]  
  
@when('sorting data: my\_sort\_with\_lambda', target\_fixture='answer')  
def sort\_with\_lambda(data):  
 return my\_sort\_with\_lambda(data)  
  
@when('sorting data: my\_sort\_without\_lambda', target\_fixture='answer')  
def sort\_without\_lambda(data):  
 return my\_sort\_without\_lambda(data)  
  
@then('data was sorted')  
def check\_answer(answer):  
 assert answer == [0, 1, -1, 4, -4, -30, 30, 100, -100, 123]

**field\_1.feature**

Feature: get goods` fields\_1  
Scenario: get goods` fields\_1  
Given goods\_1, \*args\_1  
When processing\_1  
Then answer\_1 got

**field\_2.feature**

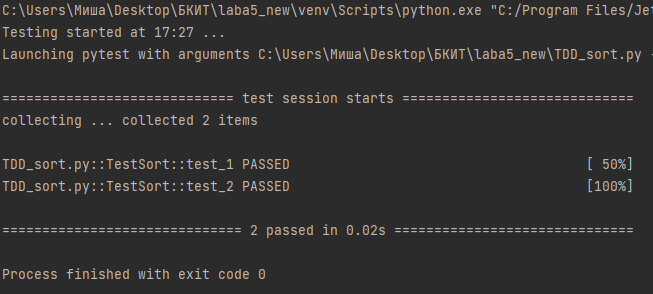
Feature: get goods` fields\_2  
Scenario: get goods` fields\_2  
Given goods\_2, \*args\_2  
When processing\_2  
Then answer\_2 got

**BDD\_field.py**

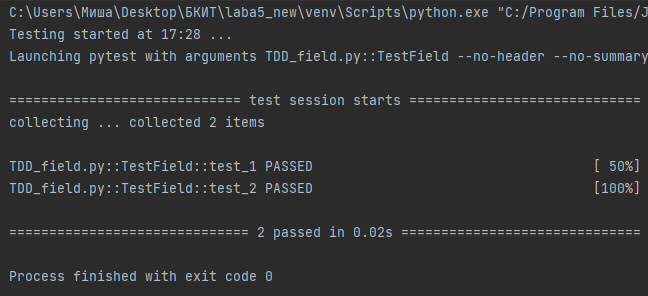
from functions.field import field  
from pytest\_bdd import scenario, given, when, then  
  
@scenario('features\\field\_1.feature','get goods` fields\_1')  
def test\_1():  
 pass  
  
@scenario('features\\field\_2.feature','get goods` fields\_2')  
def test\_2():  
 pass  
  
@given('goods\_1, \*args\_1', target\_fixture='data\_1')  
def goods():  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 args = ['title']  
 data\_1 = [goods, args]  
 return data\_1  
  
@given('goods\_2, \*args\_2', target\_fixture='data\_2')  
def goods():  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
 args = ['title', 'price']  
 data\_2 = [goods, args]  
 return data\_2  
  
@when('processing\_1', target\_fixture='answer\_1')  
def processing(data\_1):  
 return field(data\_1[0], \*data\_1[1])  
  
@when('processing\_2', target\_fixture='answer\_2')  
def processing(data\_2):  
 return field(data\_2[0], \*data\_2[1])  
  
@then('answer\_1 got')  
def check\_answer(answer\_1):  
 assert answer\_1 == ['Ковер', 'Диван для отдыха']  
  
@then('answer\_2 got')  
def check\_answer(answer\_2):  
 assert answer\_2 == [  
 {'title': 'Ковер', 'price': 2000},  
 {'title': 'Диван для отдыха', 'price': 5300}  
 ]

**Результаты выполнения программы:**

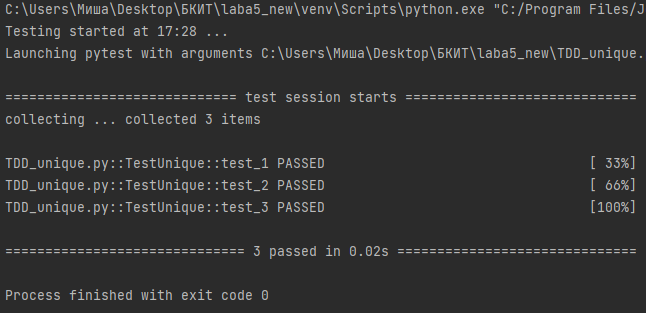
**TDD\_sort.py**



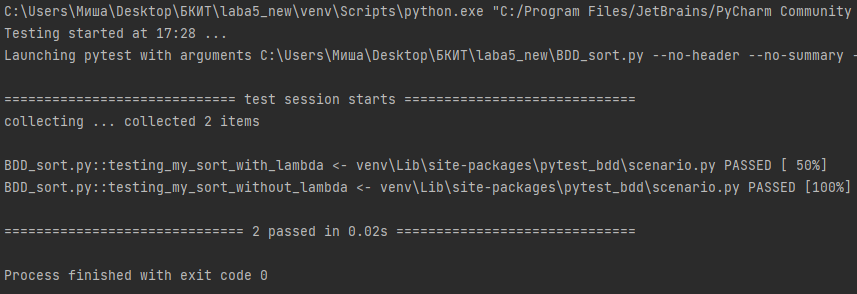
**TDD\_field.py**



**TDD\_unique.py**



**BDD\_sort.py**



**BDD\_field.py**

