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

Выполнил:	Проверил:
студент группы ИУ5-32Б:	преподаватель каф. ИУ5
Ховен Ольги	Гапанюк Ю.Е.
Александровны	
Подпись и дата:	Подпись и дата:

Москва, 2022 г.

test_tdd_field.py

```
import pytest as pytest

def field(items, *args):
    result = {}
    assert len(args) > 0
    for d in items:
        for i,j in d.items():
            if i in args:
                result[i] = j
        if len(result) == 1:
            s = result.popitem()
            s = "'" + str(s[1]) + "'"
            yield s
        else :
            yield result

@pytest.fixture
def goods():
    goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
             {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
    return goods

def test_1(goods):
    t = list(field(goods, 'title'))
    res = ["'Ковер'", "'Диван для отдыха'"]
    assert res == t

def test_2(goods):
    t = field(goods, 'price', 'title')
    assert next(t) == {'title': 'Ковер', 'price': 2000} and next(t) == {'title': 'Диван для отдыха', 'price': 5300}
```

test_tdd_sort.py

```
import pytest

def my_sort1(data):
    return sorted(data, key = abs, reverse = True)

def my_sort2(data):
    return sorted(data, key = lambda n: -abs(n))
```

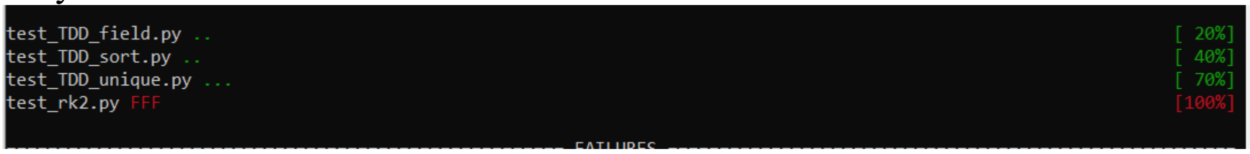
```
def test_my_sort1():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    res = [123, 100, -100, -30, 4, -4, 1, -1, 0]
    assert res == my_sort1(data)

def test_my_sort2():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    res = [123, 100, -100, -30, 4, -4, 1, -1, 0]
    assert res == my_sort2(data)
```

test_tdd_unique.py

```
import pytest
import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.seen = []
        for i in items:
            if  len(kwargs) > 0 and kwargs["ignore_case"]:
                flag = True
                for j in self.seen:
                    if j.lower() == i.lower():
                        flag = False
                if flag:
                    (self.seen).append(i)
            else:
                if i in self.seen:
                    continue
                self.seen.append(i)

    def __next__(self):
        if len(self.seen) == 0:
            raise StopIteration
        item = self.seen[0]
        del self.seen[0]
        return item

    def __iter__(self):
        return self

@pytest.fixture
def data():
    d = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    return d
```

```
def test_unique_ignore_case_True(data):
    t = list(Unique(data, ignore_case = True))
    res = ['a', 'b']
    assert res == t

def test_unique_ignore_case_False(data):
    t = list(Unique(data))
    res = ['a', 'A', 'b', 'B']
    assert res == t

def test_unique_numbers():
    data = [1,1,1,1,1,1,2,2,2,2,2,2]
    t = list(Unique(data))
    res = [1, 2]
    assert res == t
```

Результаты TDD тестов



```
test_TDD_field.py ..                                                                     [ 20%]
test_TDD_sort.py ..                                                                      [ 40%]
test_TDD_unique.py ...                                                                   [ 70%]
test_rk2.py FFF                                                                          [100%]

------------------------------------------------------- FAILURES -------------------------------------------------------
```

Field_1.py

```
def field(items, *args):
    result = {}
    assert len(args) > 0
    for d in items:
        for i,j in d.items():
            if i in args:
                result[i] = j
        if len(result) == 1:
            s = result.popitem()
            s = "'" + str(s[1]) + "'"
            yield s
        else :
            yield result
```

sort_1.py

```
def sort_lambda(data):
    return sorted(data, key = lambda x: abs(x))
```

```
def sort_w_l(data):
    return sorted(data, key = abs)
```

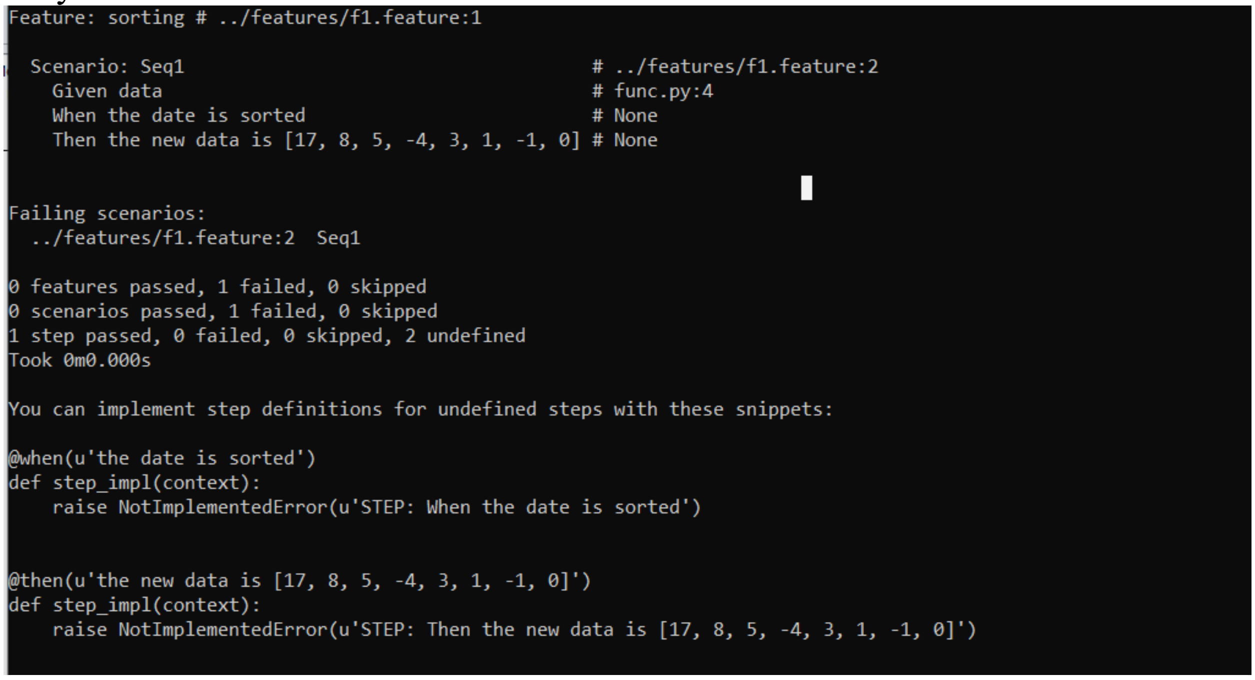
func.py

```
from sort_1 import sort_lambda, sort_w_l
from behave import *

@given('data')
def step_impl(context):
    pass
@when('the data is sorted')
def step_impl(context):
    context.data=sort_w_l(context.data)
@then('the new data is [17, 5, -4, 4, 3, 1, -1, 0]')
def step_impl(context):
    assert context.data==[17, 5, -4, 4, 3, 1, -1, 0]
```

unique_1.py

```
import gen_random
class Unique(object):
    def __init__(self, items, **kwargs):
        self.count = 0
        if len(kwargs) == 0:
            self.ignore_case = False
        else:
            value = kwargs['ignore_case']
            self.ignore_case = value
        self.items = []
        for num in items:
            if isinstance(num, str):
                if self.ignore_case == True:
                    num_ = num.lower()
                    if num_ not in self.items:
                        self.items.append(num_)
                else:
                    if num not in self.items:
                        self.items.append(num)
            else:
                if num not in self.items:
                    self.items.append(num)

    def __next__(self):
        if self.count < len(self.items) - 1:
            self.count += 1
            return self.items[self.count]
        else:
            raise StopIteration

    def __iter__(self):
        return self

    def __repr__(self):
        return str(self.items)
```

Результат BDD тестов:



```
Feature: sorting # ../features/f1.feature:1

  Scenario: Seq1                                      # ../features/f1.feature:2
    Given data                                        # func.py:4
    When the date is sorted                           # None
    Then the new data is [17, 8, 5, -4, 3, 1, -1, 0] # None


Failing scenarios:
  ../features/f1.feature:2  Seq1

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 0 failed, 0 skipped, 2 undefined
Took 0m0.000s

You can implement step definitions for undefined steps with these snippets:

@when(u'the date is sorted')
def step_impl(context):
    raise NotImplementedError(u'STEP: When the date is sorted')


@then(u'the new data is [17, 8, 5, -4, 3, 1, -1, 0]')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then the new data is [17, 8, 5, -4, 3, 1, -1, 0]')
```