Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологиий»

Лабораторная работа №5.

Выполнил:
студент группы ИУ5-34Б:
Белозеров Дмитрий Сергеевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич
Подпись и дата:

2022

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).
    - Создание Mock-объектов (необязательное дополнительное задание).

Решение с использованием TDD фреймворка.

unique_TDD.py

```
import unittest


class Uniquetest(unittest.TestCase):
    def test_with_default_ignore_case(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        res = ['a', 'A', 'b', 'B']
        t = list(Unique(data))
        self.assertEqual(t, res)

    def test_with_true_ignore_case(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        res = ['a', 'b']
        t = list(Unique(data, ignore_case=True))
        self.assertEqual(t, res)

    def test_numbers_with_default_ignore_case(self):
        data = [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2]
        res = [1, 2]
        t = list(Unique(data))
        self.assertEqual(t, res)


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


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

Результат

```
============================= test session starts ==============================
collecting ... collected 3 items

unique_TDD.py::Uniquetest::test_numbers_with_default_ignore_case PASSED  [ 33%]
unique_TDD.py::Uniquetest::test_with_default_ignore_case PASSED          [ 66%]
unique_TDD.py::Uniquetest::test_with_true_ignore_case PASSED             [100%]

============================== 3 passed in 0.03s ===============================

Process finished with exit code 0
```

sort_TDD.py

```
import unittest


class Sorttest(unittest.TestCase):
    def setUp(self):
        self.data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
        self.res = [123, 100, -100, -30, 4, -4, 1, -1, 0]

    def test_result(self):
        self.assertEqual(first_result(self.data), self.res)

    def test_result_with_lambda(self):
        self.assertEqual(second_result(self.data), self.res)


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


def second_result(data):
    return sorted(data, key = lambda x: -abs(x))


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

Результат

```
============================= test session starts ==============================
collecting ... collected 2 items

sort_TDD.py::Sorttest::test_result PASSED                                [ 50%]
sort_TDD.py::Sorttest::test_result_with_lambda PASSED                    [100%]

============================== 2 passed in 0.04s ===============================

Process finished with exit code 0
```

field_TDD.py

```
import unittest


class Testfield(unittest.TestCase):
    def setUp(self):
        self.goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
                      {'title': 'Диван для отдыха', 'price': 5300, 'color':
'black'}]

    def test_with_one_arg(self):
        res = ["'Ковер'", "'Диван для отдыха'"]
        count = 0
        for i in field(self.goods, 'title'):
            self.assertEqual(i, res[count])
            count += 1

    def test_with_two_args(self):
        res = [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для
отдыха', 'price': 5300}]
        count = 0
        for i in field(self.goods, 'title', 'price'):
            self.assertEqual(i, res[count])
            count += 1


def field(goods, *args):
    assert len(args) > 0
    line = {}
    for i in goods:
        for v,b in i.items():
            if v in args:
                safe_v = v
                line[v] = b
        if len(line) == 0:
            yield None
        if len(line) == 1:
            yield "'" + line[safe_v] + "'"
        else:
            yield line


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

Результат

```
============================= test session starts ==============================
collecting ... collected 2 items

field_TDD.py::Testfield::test_with_one_arg PASSED                        [ 50%]
field_TDD.py::Testfield::test_with_two_args PASSED                       [100%]

============================== 2 passed in 0.03s ===============================

Process finished with exit code 0
```

Решение с использованием BDD фреймворка.

unique_BDD.py

```python
from pytest_bdd import scenario, given, when, then


@scenario('features/unique_BDD_1.feature', "Getting list of unique elements")
def test_unique_1():
    pass

@scenario('features/unique_BDD_2.feature', "Getting list of unique elements")
def test_unique_2():
    pass

@scenario('features/unique_BDD_3.feature', "Getting list of unique elements")
def test_unique_3():
    pass


@given('Some string data', target_fixture= 'data_1')
def data():
    return ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']


@given('Some numeric data', target_fixture= 'data_2')
def data():
    return [1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2]


@when('String data is getting unique without ignore_case',
target_fixture='unique_data_1')
def unique_data(data_1):
    return unique_func(data_1)


@when('String data is getting unique with ignore_case',
target_fixture='unique_data_2')
def unique_data(data_1):
    return unique_func(data_1, 1)


@when('Numeric data is getting unique without ignore_case',
```

```
target_fixture='unique_data_3')
def unique_data(data_2):
    return unique_func(data_2)


@then('String data is unique without ignore_case')
def res_data(unique_data_1):
    assert unique_data_1 == ['a', 'A', 'b', 'B']


@then('String data is unique with ignore_case')
def res_data(unique_data_2):
    assert unique_data_2 == ['a', 'b']


@then('Numeric data is unique')
def res_data(unique_data_3):
    assert unique_data_3 == [1, 2]


def unique_func(data, flag=0):
    if flag == 1: return list(Unique(data, ignore_case = True))
    if flag == 0: return list(Unique(data))


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
```

unique_BDD_1.feature

```
Feature: Get only unique elements of list
  Scenario: Getting list of unique elements
    Given Some string data
    When String data is getting unique without ignore_case
    Then String data is unique without ignore_case
```

unique_BDD_2.feature

```
Feature: Get only unique elements of list
  Scenario: Getting list of unique elements
    Given Some string data
    When String data is getting unique with ignore_case
    Then String data is unique with ignore_case
```

unique_BDD_3.feature

```
Feature: Get only unique elements of list
  Scenario: Getting list of unique elements
    Given Some numeric data
    When Numeric data is getting unique without ignore_case
    Then Numeric data is unique
```

Результат

```
============================= test session starts ==============================
collecting ... collected 3 items

unique_BDD.py::test_unique_1 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [ 33%]
unique_BDD.py::test_unique_2 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [ 66%]
unique_BDD.py::test_unique_3 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 3 passed in 0.05s ===============================

Process finished with exit code 0
```

sort_BDD.py

```
from pytest_bdd import scenario, given, when, then


@scenario('features/sort_BDD_func.feature','Data array to be sorted by
function')
def testing_my_sort1():
    pass


@scenario('features/sort_BDD_lambda.feature','Data array to be sorted by
lambda')
def testing_my_sort2():
    pass

@given('Some data', target_fixture = "data")
def given_data():
    return [4, -30, 100, -100, 123, 1, 0, -1, -4]


@when('Data array get sorted with first_result', target_fixture = "result")
def result_with_first_result(data):
    return first_result(data)


@when('Data array get sorted with second_result', target_fixture = "result")
def result_with_first_result(data):
    return second_result(data)
```

```python
@then('Data array is sorted')
def sorted_data(result):
    assert result == [123, 100, -100, -30, 4, -4, 1, -1, 0]


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


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

sort_BDD_lambda.feature

```
# Created by beloz at 21.11.2022
Feature: Descending sort by lambda

  Scenario: Data array to be sorted by lambda
    Given Some data
    When Data array get sorted with second_result
    Then Data array is sorted
```

sort_BDD_func.feature

```
# Created by beloz at 21.11.2022
Feature: Descending sort by function

  Scenario: Data array to be sorted by function
    Given Some data
    When Data array get sorted with first_result
    Then Data array is sorted
```

Результат

```
============================= test session starts =============================
collecting ... collected 2 items

sort_BDD.py::testing_my_sort1 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [ 50%]
sort_BDD.py::testing_my_sort2 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 2 passed in 0.04s ==============================

Process finished with exit code 0
```

field_BDD.py

```python
from pytest_bdd import scenario, given, when, then


@scenario('features/field_BDD_1.feature','Getting dictionary keys')
def testing_field_1():
    pass
```

```
@scenario('features/field_BDD_2.feature','Getting dictionary keys')
def testing_field_2():
    pass


@given('Some dict', target_fixture = "data")
def given_dict():
    return [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
                    {'title': 'Диван для отдыха', 'price': 5300, 'color':
'black'}]


@when('Getting dictionary keys from some dict with 3 args', target_fixture =
"result_1")
def result_with_first_result(data):
    line = []
    for i in field(data, 'title', 'price'):
        line.append(dict(i))
    return line


@when('Getting dictionary keys from some dict with 2 args', target_fixture =
"result_2")
def result_with_first_result(data):
    line = []
    for i in field(data, 'title'):
        line.append(i)
    return line


@then('Dictionary keys from some dict with 3 args')
def res_dict_1(result_1):
    assert result_1 == [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван
для отдыха', 'price': 5300}]


@then('Dictionary keys from some dict with 2 args')
def res_dict_1(result_2):
    assert result_2 == ["'Ковер'", "'Диван для отдыха'"]


def field(goods, *args):
    assert len(args) > 0
    line = {}
    for i in goods:
        for v,b in i.items():
            if v in args:
                safe_v = v
                line[v] = b
        if len(line) == 0:
            yield None
        if len(line) == 1:
            yield "'" + line[safe_v] + "'"
        else:
            yield line
```

field_BDD_1.feature

```
Feature: Dictionary keys

  Scenario: Getting dictionary keys
    Given Some dict
```

```
    When Getting dictionary keys from some dict with 3 args
    Then Dictionary keys from some dict with 3 args
```

field_BDD_2.feature

```
Feature: Dictionary keys

  Scenario: Getting dictionary keys
    Given Some dict
    When Getting dictionary keys from some dict with 2 args
    Then Dictionary keys from some dict with 2 args
```

Результат

```
============================= test session starts =============================
collecting ... collected 2 items

field_BDD.py::testing_field_1 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [ 50%]
field_BDD.py::testing_field_2 <- ..\venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 2 passed in 0.05s ==============================

Process finished with exit code 0
```