Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе № 5
«Модульное тестирование в Python»
Вариант № 13

Выполнил:
студент группы РТ5-31Б
Романов Ю.И.
Подпись и дата:

Проверил:
к.т.н., доц. каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

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

field.py:

```
def field(items, *args):
    assert len(args) > 0, "Необходимо передать хотя бы одно поле"

    if len(args) == 1:
        key = args[0]
        for item in items:
            if key in item and item[key] is not None:
                yield item[key]

    else:
        for item in items:
            result = {}
            for key in args:
                if key in item and item[key] is not None:
                    result[key] = item[key]

            if result:
                yield result
```

test_field.py:

```
import unittest
from field import field

class TestFieldGenerator(unittest.TestCase):
    '''TDD тесты для генератора field'''
```

```
    def test_for_one_field(self):
        '''Тест 1: Возвращаем значения для одного поля'''
        goods = [{'title': 'Ковер', 'price': 2000},
                 {'title': 'Диван', 'price': 3000},
                 {'title': None, 'price': 1000}]
        result = list(field(goods, 'title'))
        expected = ['Ковер', 'Диван']
        self.assertEqual(result, expected)

    def test_for_two_fields(self):
        '''Тест 2: Возвращаем значения для двух полей'''
        goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
                 {'title': 'Диван', 'color': 'black'},
                 {'price': 1000, 'color': 'white'}]
        result = list(field(goods, 'title', 'price'))
        expected = [{'title': 'Ковер', 'price': 2000},
                    {'title': 'Диван'},
                    {'price': 1000}]
        self.assertEqual(result, expected)

    def test_for_none_fields(self):
        '''Тест 3: None значения правильно фильтруются'''
        goods = [{'title': None, 'price': 100},
                 {'title': 'Шкаф', 'price': None},
                 {'title': None, 'price': None}]
        result = list(field(goods, 'title', 'price'))
        expected = [{'price': 100},
                    {'title': 'Шкаф'}]
        self.assertEqual(result, expected)

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

field.feature:

```
# language: ru
Функционал: Генератор field

  Сценарий: Извлечение поля title
    Допустим есть товары "Ковер" и "Диван для отдыха"
    Когда извлекаю поле title
    Тогда получаю ["Ковер", "Диван для отдыха"]

  Сценарий: Извлечение полей title и price
```

```
    Допустим есть товары с ценами
    Когда извлекаю поля title и price
    Тогда получаю словари с этими полями

  Сценарий: Пропуск None значений
    Допустим есть товары с пропущенными данными
    Когда извлекаю поля title и price
    Тогда None значения пропускаются
```

field_steps.py:

```
from behave import given, when, then
from field import field

@given('есть товары "Ковер" и "Диван для отдыха"')
def step_given_simple_goods(context):
    context.items = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},
                     {'title': 'Диван для отдыха', 'color': 'black'},
                     {'title': None, 'price': 1000}]

@given('есть товары с ценами')
def step_given_goods_with_prices(context):
    context.items = [{'title': 'Ковер', 'price': 2000},
                     {'title': 'Диван', 'price': 3000},
                     {'title': 'Стул', 'price': None}]

@given('есть товары с пропущенными данными')
def step_given_goods_with_missing(context):
    context.items = [{'title': None, 'price': 100},
                     {'title': 'Шкаф', 'price': None},
                     {'title': None, 'price': None}]

@when('извлекаю поле title')
def step_when_extract_title(context):
    context.result = list(field(context.items, 'title'))

@when('извлекаю поля title и price')
def step_when_extract_title_price(context):
    context.result = list(field(context.items, 'title', 'price'))

@then('получаю ["{value1}", "{value2}"]')
def step_then_get_list(context, value1, value2):
    assert context.result == [value1, value2]

@then('получаю словари с этими полями')
```

```python
def step_then_get_dicts_simple(context):
    assert len(context.result) > 0
    assert all(isinstance(item, dict) for item in context.result)

@then('None значения пропускаются')
def step_then_none_filtered_simple(context):
    for item in context.result:
        for value in item.values():
            assert value is not None
```

Экранные формы с примерами выполнения программы

TDD-тесты:

```
yuriy@MacBook-Air-Urij code % source "/Users/yuriy/Бауманка/2 курс/BMSTU
(venv) yuriy@MacBook-Air-Urij code % python3 -m unittest test_field.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

BDD-тесты:

```
yuriy@MacBook-Air-Urij code % source "/Users/yuriy/Бауманка/2 курс/BMSTU_COURSE_PCPL/la
(venv) yuriy@MacBook-Air-Urij code % behave
USING RUNNER: behave.runner:Runner
Функционал: Генератор field # features/field.feature:2

  Сценарий: Извлечение поля title                    # features/field.feature:4
    Допустим есть товары "Ковер" и "Диван для отдыха" # steps/field_steps.py:4 0.000s
    Когда извлекаю поле title                        # steps/field_steps.py:22 0.000s
    Тогда получаю ["Ковер", "Диван для отдыха"]      # steps/field_steps.py:30 0.000s

  Сценарий: Извлечение полей title и price  # features/field.feature:9
    Допустим есть товары с ценами           # steps/field_steps.py:10 0.000s
    Когда извлекаю поля title и price       # steps/field_steps.py:26 0.000s
    Тогда получаю словари с этими полями    # steps/field_steps.py:34 0.000s

  Сценарий: Пропуск None значений               # features/field.feature:14
    Допустим есть товары с пропущенными данными # steps/field_steps.py:16 0.000s
    Когда извлекаю поля title и price           # steps/field_steps.py:26 0.000s
    Тогда None значения пропускаются            # steps/field_steps.py:39 0.000s

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped
Took 0min 0.001s
```