Московский государственный технический университет
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина «Парадигмы и конструкции языков программирования»

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

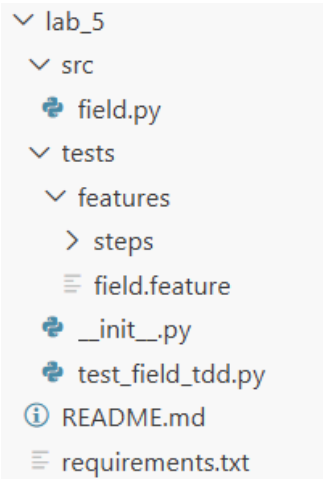
Выполнил:
студент группы ИУ5-36Б
Бойко Артем Андреевич
Проверил:
Нардид Анатолий Николаевич

Москва, 2025

Цель лабораторной работы: изучение возможностей модульного тестирования в языке Python.

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).



field.py

```python
def field(items, *args):
    """
    Генератор для извлечения полей из словарей.

    Args:
        items: Список словарей
        *args: Ключи для извлечения

    Yields:
        Если передан один аргумент - значения поля
        Если несколько аргументов - словари с указанными полями
    """
    assert len(args) > 0, "Должен быть передан хотя бы один аргумент"

    if len(args) == 1:
        # Если передан один аргумент, возвращаем только значения
        key = args[0]
        for item in items:
            if isinstance(item, dict) and key in item and item[key] is not None:
                yield item[key]
    else:
        # Если передано несколько аргументов, возвращаем словари
        for item in items:
            if not isinstance(item, dict):
                continue

            result = {}
            has_valid_fields = False
            for key in args:
                if key in item and item[key] is not None:
                    result[key] = item[key]
                    has_valid_fields = True

            if has_valid_fields:
                yield result
```

```python
def process_data(data, *fields):
    """
    Функция для обработки данных с использованием field генератора.
    Возвращает список результатов для удобства тестирования.
    """
    return list(field(data, *fields))
```

field.feature

features/field.feature

encoding: utf-8

Feature: Field Generator

  As a developer

  I want to use field generator to extract data from dictionaries

  So that I can process structured data efficiently

  Scenario: Извлечение одного поля из валидных данных

    Given список словарей с товарами

    When я извлекаю поле "title"

    Then я получаю значения "Ковер, Диван для отдыха, Стул"

    And результат содержит 3 элементов

  Scenario: Извлечение нескольких полей из валидных данных

    Given список словарей с товарами

    When я извлекаю поля "title" и "price"

    Then я получаю список словарей

    And каждый словарь содержит хотя бы одно из полей "title, price"

    And результат содержит 4 элементов

  Scenario: Обработка None значений корректно

Given список содержит словари с None значениями

When я извлекаю поле "title"

Then я получаю значения "Товар1, Товар3"

And результат содержит 2 элементов

Scenario: Обработка несуществующего поля

Given список словарей с товарами

When я извлекаю поле "non_existent"

Then результат пуст

Scenario: Использование вспомогательной функции process_data

Given список словарей с товарами

When я обрабатываю данные с полями "color"

Then я получаю значения "green, black, blue, white"

And результат содержит 4 элементов

test_field_tdd.py

```
import unittest
import sys
import os

# Правильный путь к src
sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..',
'src')))

try:
    from src.field import field, process_data
except ImportError:
    # Альтернативный вариант - импорт напрямую если файл в той же директории
    import sys
    sys.path.append('.')
    from src.field import field, process_data


class TestFieldGeneratorTDD(unittest.TestCase):
    """TDD тесты для генератора field"""
```

```python
def setUp(self):
    """Подготовка тестовых данных"""
    self.test_data = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
        {'title': None, 'price': 5000},
        {'color': 'blue'},
        {'title': 'Стул', 'price': None, 'color': 'white'},
        'invalid_item',  # Не словарь для тестирования обработки ошибок
    ]

def test_single_field_extraction(self):
    """Тест извлечения одного поля"""
    result = list(field(self.test_data, 'title'))
    expected = ['Ковер', 'Диван для отдыха', 'Стул']
    self.assertEqual(result, expected)

def test_multiple_fields_extraction(self):
    """Тест извлечения нескольких полей"""
    result = list(field(self.test_data, 'title', 'price'))
    expected = [
        {'title': 'Ковер', 'price': 2000},
        {'title': 'Диван для отдыха', 'price': 5300},
        {'price': 5000},  # title is None, so only price is included
        {'title': 'Стул'}  # price is None, so only title is included
    ]
    self.assertEqual(result, expected)

def test_none_values_handling(self):
    """Тест обработки None значений"""
    result = list(field(self.test_data, 'price'))
    expected = [2000, 5300, 5000]
    self.assertEqual(result, expected)

def test_empty_result(self):
    """Тест случая когда нет подходящих данных"""
    result = list(field(self.test_data, 'non_existent_field'))
    self.assertEqual(result, [])

def test_process_data_function(self):
    """Тест вспомогательной функции process_data"""
    result = process_data(self.test_data, 'color')
    expected = ['green', 'black', 'blue', 'white']
    self.assertEqual(result, expected)

def test_invalid_input_handling(self):
    """Тест обработки некорректного ввода"""
    # Тест с пустым списком
    result = list(field([], 'title'))
    self.assertEqual(result, [])
```

```
        # Тест с не-словарями в списке
        mixed_data = [{'a': 1}, 'string', 123, {'b': 2}]
        result = list(field(mixed_data, 'a'))
        self.assertEqual(result, [1])


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

README.md

```
# Лабораторная работа №5 - Модульное тестирование

## Установка зависимостей
```bash
pip install -r requirements.txt
```

requirements.txt

```
behave==1.2.6
unittest2==1.1.0
coverage==6.4.0
pytest==7.2.0
```
```

Результаты выполнения программы

Запуск TDD-тестов

```
● @artem-hedgehog →/workspaces/ProgLangParadigms_BoikoA_2025/lab_5 (main) $ python -m unittest tests.test_field_tdd
......
----------------------------------------------------------------------
Ran 6 tests in 0.000s

OK
```

Запуск BDD-тестов

```
● @artem-hedgehog →/workspaces/ProgLangParadigms_BoikoA_2025/lab_5 (main) $ behave tests/features/
USING RUNNER: behave.runner:Runner
Feature: Field Generator # tests/features/field.feature:4
  As a developer
  I want to use field generator to extract data from dictionaries
  So that I can process structured data efficiently
  Scenario: Извлечение одного поля из валидных данных       # tests/features/field.feature:9
    Given список словарей с товарами                        # tests/features/steps/test_field_bdd.py:24 0.000s
    When я извлекаю поле "title"                            # tests/features/steps/test_field_bdd.py:46 0.000s
    Then я получаю значения "Ковер, Диван для отдыха, Стул" # tests/features/steps/test_field_bdd.py:65 0.000s
    And результат содержит 3 элементов                      # tests/features/steps/test_field_bdd.py:90 0.000s

  Scenario: Извлечение нескольких полей из валидных данных          # tests/features/field.feature:15
    Given список словарей с товарами                                # tests/features/steps/test_field_bdd.py:24 0.000s
    When я извлекаю поля "title" и "price"                          # tests/features/steps/test_field_bdd.py:52 0.000s
    Then я получаю список словарей                                  # tests/features/steps/test_field_bdd.py:82 0.000s
    And каждый словарь содержит хотя бы одно из полей "title, price" # tests/features/steps/test_field_bdd.py:102 0.000s
    And результат содержит 4 элементов                              # tests/features/steps/test_field_bdd.py:90 0.000s

  Scenario: Обработка None значений корректно       # tests/features/field.feature:22
    Given список содержит словари с None значениями # tests/features/steps/test_field_bdd.py:36 0.000s
    When я извлекаю поле "title"                    # tests/features/steps/test_field_bdd.py:46 0.000s
    Then я получаю значения "Товар1, Товар3"        # tests/features/steps/test_field_bdd.py:65 0.000s
    And результат содержит 2 элементов              # tests/features/steps/test_field_bdd.py:90 0.000s

  Scenario: Обработка несуществующего поля  # tests/features/field.feature:28
    Given список словарей с товарами        # tests/features/steps/test_field_bdd.py:24 0.000s
    When я извлекаю поле "non_existent"     # tests/features/steps/test_field_bdd.py:46 0.000s
    Then результат пуст                     # tests/features/steps/test_field_bdd.py:96 0.000s

  Scenario: Использование вспомогательной функции process_data  # tests/features/field.feature:33
    Given список словарей с товарами                            # tests/features/steps/test_field_bdd.py:24 0.000s
    When я обрабатываю данные с полями "color"                  # tests/features/steps/test_field_bdd.py:58 0.000s
    Then я получаю значения "green, black, blue, white"         # tests/features/steps/test_field_bdd.py:65 0.000s
    And результат содержит 4 элементов                          # tests/features/steps/test_field_bdd.py:90 0.000s

1 feature passed, 0 failed, 0 skipped
5 scenarios passed, 0 failed, 0 skipped

20 steps passed, 0 failed, 0 skipped
Took 0min 0.002s
```