Защищено:
Гапанюк Ю.Е.

"__"_____2025 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2025 г.

Отчет по лабораторной работе № 5 по курсу Парадигмы и конструкции языков программирования ГУИМЦ

Тема работы: "Модульное тестирование в Python."

9
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-54Б _____
(подпись)

Щетинин Д.С.

"__"_____2025 г.

Москва, МГТУ - 2025 г.

1. Тема и задание для выполнения лабораторной работы.

Тема работы: "Изучение модульного тестирования в Python."
Задание:

1. Выбрал фрагмент кода из лабораторных работ 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).

2. Листинг программы

Файл: run_behave.py

```
from behave.__main__ import main as behave_main
behave_main(["features/process.feature"])
```

Файл: features/steps/process_steps.py

```
from behave import *
from lab_python_fp.process_data import DataProcessor

@given('имеются данные о вакансиях')
def step_impl(context):
    context.processor = DataProcessor()
    context.data = [
        {'job-name': 'Программист Python'},
        {'job-name': 'Аналитик'},
        {'job-name': 'Программист Java'}
    ]

@when('применяется конвейер обработки')
def step_impl(context):
    context.result = context.processor.process_pipeline(context.data)

@then('получаем только вакансии программистов с Python и зарплатой')
def step_impl(context):
    assert len(context.result) == 2
    assert all('Программист' in job for job in context.result)
    assert all('с опытом Python' in job for job in context.result)
    assert all('зарплата' in job for job in context.result)
```

Файл: features/process.feature

```
Feature: Обработка данных о вакансиях
  Проверка корректности обработки вакансий

  Scenario: Фильтрация вакансий программистов
    Given имеются данные о вакансиях
    When применяется конвейер обработки
    Then получаем только вакансии программистов с Python и зарплатой
```

Файл: lab_python_fp/cm_timer.py

```
import time
from contextlib import contextmanager


class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        print(f"time: {time.time() - self.start_time}")


@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    print(f"time: {time.time() - start_time}")


if __name__ == '__main__':
    print("Тестирование cm_timer_1:")
    with cm_timer_1():
        time.sleep(1.5)

    print("\nТестирование cm_timer_2:")
    with cm_timer_2():
        time.sleep(1.5)
```

Файл: lab_python_fp/data_light.json

```
[
  {
    "job-name": "Программист Python",
    "salary": 120000
  },
  {
    "job-name": "Программист Java",
    "salary": 110000
  },
  {
    "job-name": "Аналитик данных",
    "salary": 90000
  },
  {
    "job-name": "Программист C++",
    "salary": 130000
  },
  {
    "job-name": "Веб-разработчик",
    "salary": 100000
  }
]
```

Файл: lab_python_fp/field.py

```
def field(items, *args):
    assert len(args) > 0
```

```
    for item in items:
        if len(args) == 1:
            value = item.get(args[0])
            if value is not None:
                yield value
        else:
            result = {}
            has_values = False
            for key in args:
                value = item.get(key)
                if value is not None:
                    result[key] = value
                    has_values = True
            if has_values:
                yield result



if __name__ == "__main__":
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]

    # Тест 1
    print("Test 1:")
    for item in field(goods, 'title'):
        print(item)

    # Тест 2
    print("\nTest 2:")
    for item in field(goods, 'title', 'price'):
        print(item)
```

Файл: lab_python_fp/gen_random.py

```
import random

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randint(begin, end)



if __name__ == "__main__":
    # Тест
    print("Test:")
    for num in gen_random(5, 1, 3):
        print(num, end=" ")
    print()
```

Файл: lab_python_fp/print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(func.__name__)
        if isinstance(result, list):
            for item in result:
```

```
            print(item)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)
        return result
    return wrapper

@print_result
def test_1():
    return 1

@print_result
def test_2():
    return 'iu5'

@print_result
def test_3():
    return {'a': 1, 'b': 2}

@print_result
def test_4():
    return [1, 2]

if __name__ == '__main__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

Файл: lab_python_fp/process_data.py

```
import json
import sys
from .gen_random import gen_random
from .unique import Unique
from .print_result import print_result
from .cm_timer import cm_timer_1


class DataProcessor:
    """Класс для обработки данных о вакансиях"""

    @staticmethod
    def load_data(path):
        """Загрузка данных из JSON файла"""
        with open(path, 'r', encoding='utf-8') as f:
            return json.load(f)

    @print_result
    def f1(self, data):
        """Этап 1: Уникальные названия с сортировкой"""
        job_names = [item['job-name'] for item in data]
        return sorted(Unique(job_names, ignore_case=True), key=str.lower)

    @print_result
    def f2(self, job_names):
```

```
        """Этап 2: Фильтрация программистов"""
        return [name for name in job_names if name.lower().startswith('программист')]

    @print_result
    def f3(self, job_names):
        """Этап 3: Добавление Python-опыта"""
        return [f"{name} с опытом Python" for name in job_names]

    @print_result
    def f4(self, job_names):
        """Этап 4: Добавление зарплат"""
        salaries = list(gen_random(len(job_names), 100000, 200000))
        return [f"{job}, зарплата {salary} руб." for job, salary in zip(job_names, salaries)]

    def process_pipeline(self, data):
        """Запуск полного конвейера обработки"""
        return self.f4(self.f3(self.f2(self.f1(data))))


if __name__ == '__main__':
    if len(sys.argv) < 2:
        print("Usage: python process_data.py <path_to_json>")
        sys.exit(1)

    processor = DataProcessor()
    data = processor.load_data(sys.argv[1])
    with cm_timer_1():
        result = processor.process_pipeline(data)
```

Файл: lab_python_fp/sort.py

```
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

if __name__ == '__main__':
    # Без lambda
    result = sorted(data, key=abs, reverse=True)
    print("Без использования lambda-функции:", result)

    # С lambda
    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print("С использованием lambda-функции:", result_with_lambda)
```

Файл: lab_python_fp/unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        self.items = iter(items)
        self.seen = set()

    def __next__(self):
        while True:
            item = next(self.items)
            key = item.lower() if self.ignore_case and isinstance(item, str) else item
            if key not in self.seen:
                self.seen.add(key)
                return item

    def __iter__(self):
        return self
```

```
if __name__ == "__main__":
    # Тест 1
    print("Test 1 (numbers):")
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for item in Unique(data):
        print(item, end=" ")
    print()

    # Тест 2
    print("\nTest 2 (strings):")
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for item in Unique(data, ignore_case=True):
        print(item, end=" ")
    print()
```

Файл: tests/test_mocks.py

```
from unittest import TestCase, mock
from lab_python_fp.process_data import DataProcessor


class TestWithMocks(TestCase):
    @mock.patch('lab_python_fp.process_data.gen_random')
    def test_f4_with_mocked_salaries(self, mock_gen):
        mock_gen.return_value = [150000, 180000]
        processor = DataProcessor()
        result = processor.f4(['Job1', 'Job2'])

        self.assertEqual(result, [
            'Job1, зарплата 150000 руб.',
            'Job2, зарплата 180000 руб.'
        ])
        mock_gen.assert_called_once_with(2, 100000, 200000)
```

Файл: tests/test_process.py

```
import unittest
from unittest.mock import patch
from lab_python_fp.process_data import DataProcessor

class TestDataProcessor(unittest.TestCase):
    def setUp(self):
        self.processor = DataProcessor()
        self.test_data = [
            {'job-name': 'Программист Python'},
            {'job-name': 'Аналитик'},
            {'job-name': 'Программист Java'}
        ]

    def test_f1_sorts_and_returns_unique_names(self):
        result = self.processor.f1(self.test_data)
        self.assertEqual(result, ['Аналитик', 'Программист Java', 'Программист Python'])

    def test_f2_filters_only_programmers(self):
        names = ['Программист Python', 'Аналитик', 'Программист Java']
        result = self.processor.f2(names)
        self.assertEqual(result, ['Программист Python', 'Программист Java'])
```

```
    @patch('lab_python_fp.process_data.gen_random')
    def test_f4_with_mocked_salaries(self, mock_gen):
        mock_gen.return_value = [150000, 180000]
        result = self.processor.f4(['Job1', 'Job2'])
        self.assertEqual(result, [
            'Job1, зарплата 150000 руб.',
            'Job2, зарплата 180000 руб.'
        ])
        mock_gen.assert_called_once_with(2, 100000, 200000)

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

3. Результаты работы программы

```
C:\Program Files\Python313\python.exe" "C:/Program Files/JetBrains/PyCharm 2025.2.1.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --path C:\Users\Денис\Desktop\Python-4lab\tests\test_process.py
Testing started at 7:52 ...
Launching unittests with arguments python -m unittest C:\Users\Денис\Desktop\Python-4lab\tests\test_process.py in C:\Users\Денис\Desktop\Python-4lab

f1
Аналитик
Программист Java
Программист Python
f2
Программист Python
Программист Java
f4
Job1, зарплата 150000 руб.
Job2, зарплата 180000 руб.


Ran 3 tests in 0.018s

OK
```

C:\Program Files\Python313\python.exe" "C:/Program Files/JetBrains/PyCharm 2025.2.1.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --target tests.test_mocks.TestWithMocks
Testing started at 7:52 ...
Launching unittests with arguments python -m unittest tests.test_mocks.TestWithMocks in C:\Users\Денис\Desktop\Python-4lab

f4
Job1, зарплата 150000 руб.
Job2, зарплата 180000 руб.

Ran 1 test in 0.008s

OK

C:\Program Files\Python313\python.exe" "C:/Program Files/JetBrains/PyCharm 2025.2.1.1/plugins/python-ce/helpers/pycharm/_jb_unittest_runner.py" --path C:\Users\Денис\Desktop\Python-4lab\tests\test_mocks.py
Testing started at 7:53 ...
Launching unittests with arguments python -m unittest C:\Users\Денис\Desktop\Python-4lab\tests\test_mocks.py in C:\Users\Денис\Desktop\Python-4lab

f4
Job1, зарплата 150000 руб.
Job2, зарплата 180000 руб.

Ran 1 test in 0.009s

OK