Лабораторная работа 3

Тема: «Разработка модульных тестов с применением библиотек python»

Цель: «Реализовать набор модульных тестов для приложения из лабораторной работы 2 (консольное приложение - игра с применением принципов объектно-ориентированного программирования)»

Содержание

Задача на самостоятельную реализацию

Реализовать модули тестирования (работа с библиотеками pytest или unittests) для классов консольной игры. Одновременный запуск всех тестов возможен с помощью pytest или библиотеки nose2.

Выполнить задание в рамках лабораторной работы и задание 2 в рамках курсового проекта.

Задача 1. Реализовать набор тестов для программы

А. Создать файлы для проверки функционала программы

Для каждого отдельного модуля создается свой класс тестов (pytest или unittests). Проверить следующие требования:

1. Класс игрового поля
    a. Создание поля заданного размера;
    b. Ошибки/ввод неверных данных при создании поля;
    c. Контроль максимального числа объектов на поле;
    d. Удаление и добавление элементов поля, в т.ч. при достижении максимальных и минимальных параметров.
2. Класс юнитов (элементов поля)
    a. Проверка общего интерфейса юнитов – позитивный исход, ошибочный исход;
    b. Проверка работы с атрибутами юнитов (здоровье, броня и т.д.);
    c. Проверка перемещения по карте.
3. Класс базы
    a. Проверка размещения на карте;
    b. Проверка генерации юнитов;
    c. Проверка учёта юнитов, реакций на уничтожение, создание;
    d. Проверка функций управления базой.
4. Проверка ландшафта
    a. Ландшафт должен влиять на юнитов.
5. Проверка нейтральных объектов
    a. Взаимодействие юнитов с нейтральными объектами;
    b. Интерфейса нейтральных объектов.

Реализуемые типы тестов

a. Позитивные тесты для функций;
b. Проверка ошибочных данных;
c. Проверка выпадения exception;
d. Тесты с данными (для консоли, например).

Б. Запустить все тесты библиотекой pytest или nose2. При необходимости отладить решение

Задача 2. Реализовать набор тестов для курсового проекта

Разработать и реализовать набор тестовых классов по теме курсового проекта.

Теория и материалы

При разработке модульных тестов следует придерживаться следующих рекомендаций:

- каждый тест должен проверять небольшой срез функциональности (1 метод 1 кейс, в идеале);
- тесты должны быть автономными и изолированными (как и методы для тестов);
- тестировать следует как ожидаемые, так и ошибочные ситуации (ожидаемой ситуацией может быть exception, который обработан в методе).

Как правило, тестируются следующие элементы программы (рис. 1) в области видимости теста (тестовая фикстура для pytest): функции, классы, модули, сессии подключения.

Рис. 1. Список составляющих теста на примере фикстуры pytest

Тестовые фикстуры инициализируют тестовые функции. Они обеспечивают надежность тестов, согласованность и повторяемость их результатов. При инициализации можно настраивать сервисы, состояния, переменные окружения.

Фикстуры (fixtures) – это функции, выполняемые pytest до (а иногда и после) фактических тестовых функций. Код в фикстуре может делать все, что вам необходимо. Часто фикстуры используются для получения набора данных для тестирования. Для библиотеки unittests фикстуры создаются вручную.

Существуют также термины стабы (stubs) и моки (mocks).

Стаб (stab) – объект, имитирующий заданное состояние. С помощью этого объекта проверяется состояние тестируемого класса или результат выполненного метода.

Мок (mock) – объект, у которого есть ожидания, или конкретная фиктивная реализацию интерфейса для тестирования взаимодействия, относительно которого высказывается утверждение. В процедурном программировании аналогичная конструкция называется заглушкой (dummy). С помощью моков, например, тестируется взаимодействие с базой данных.

По теме можно посмотреть следующие материалы:

1. Лекция Unit-тестирование алгоритма сортировки пузырьком и других функций. URL: https://www.youtube.com/watch?v=JBeZ80IvdFI
2. Дополнительно. Работа с mocks в unit-тестировании. URL: https://www.youtube.com/watch?v=kd2TzV8Ut-A
3. Курс на Stepik. Основы ООП. Введение в паттерны проектирования. URL: https://stepik.org/course/100558/syllabus
4. Статья по unittests. URL: https://pythonworld.ru/moduli/modul-unittest.html
5. Фикстуры pytest: явные, модальные, расширяемые. URL: https://pytest-docs-ru.readthedocs.io/ru/latest/fixture.html
6. Документация unittests. URL: https://docs.python.org/3/library/unittest.html
7. Документация nose2 (требуется установка). URL: https://docs.nose2.io/en/latest/index.html
8. Документация pytest (требуется установка). URL: https://docs.pytest.org/en/7.1.x/
9. Библиотека hypothesis (требуется установка). URL: https://hypothesis.readthedocs.io/en/latest/

Пример использования на рис. 2 и по url: https://habr.com/ru/post/354144/?

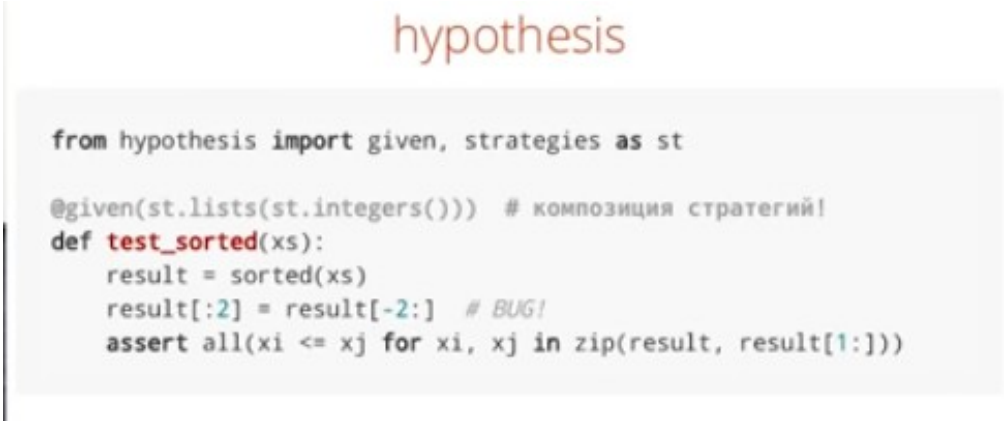

Рис. 2. Пример тестирования с библиотекой hypothesis

Технические требования

А. Реализация на Python 3.*, использовать библиотеку unittests. Можно заменить unittests на pytest и использовать тестовые фикстуры.

Б. Для запуска всех тестов использовать библиотеку pytest или nose2. Учитывать требования к названию тестов.

В. Обратить внимание на структуру проекта.

Г. Можно использовать docker для автоматизации тестирования.

Требования к отчёту по работе

1. Титульный лист;
2. Описание постановки задачи;
3. Схема программы;
4. Описания тестовых классов;
5. Скриншоты работы программы – запуск, тестирование, тестирование с pytest или nose2.