Кейс 1: Написание и выполнение простых тестов

Задание:

1. Создать файл `test_sample.py`.
2. Написать простую функцию, например, `def add(a, b): return a + b`.
3. Написать тесты для этой функции, проверяющие корректность ее работы.

Ожидаемый результат:

- Файл `test_sample.py` создан.
- Тесты написаны и успешно выполняются с использованием `pytest`.

Кейс 2: Использование фикстур

Задание:

1. Создать файл `test_fixtures.py`.
2. Написать фикстуру для инициализации некоторых данных (например, создание списка).
3. Написать тесты, использующие эту фикстуру.

Ожидаемый результат:

- Файл `test_fixtures.py` создан.
- Фикстура инициализирует данные.
- Тесты используют фикстуру и выполняются успешно.

Кейс 3: Параметризованные тесты

Задание:

1. Создать файл `test_parametrize.py`.
2. Написать параметризованные тесты для проверки функции `add` с различными наборами входных данных.

Ожидаемый результат:

- Файл `test_parametrize.py` создан.
- Параметризованные тесты написаны и выполняются успешно.

Кейс 4: Тестирование исключений

Задание:

1. Создать файл `test_exceptions.py`.

2. Написать функцию, которая выбрасывает исключение при определенных условиях.
3. Написать тесты, проверяющие правильность выброса исключений.

Ожидаемый результат:

- Файл `test_exceptions.py` создан.
- Тесты проверяют выброс исключений и выполняются успешно.

Кейс 5: Мокирование объектов

Задание:

1. Создать файл `test_mock.py`.
2. Использовать библиотеку `unittest.mock` для мокирования объекта.
3. Написать тесты, проверяющие взаимодействие с замокированным объектом.

Ожидаемый результат:

- Файл `test_mock.py` создан.
- Объекты замокированы, тесты проверяют взаимодействие и выполняются успешно.

Кейс 6: Организация тестов с использованием классов

Задание:

1. Создать файл `test_class.py`.
2. Организовать тесты в классы, используя `pytest` классы для группировки связанных тестов.

Ожидаемый результат:

- Файл `test_class.py` создан.
- Тесты организованы в классы и выполняются успешно.

Кейс 7: Генерация отчетов о тестировании

Задание:

1. Настроить генерацию отчетов о тестировании с помощью плагина `pytest-html`.
2. Запустить тесты и сгенерировать HTML отчет.

Ожидаемый результат:

- Отчет о тестировании сгенерирован в формате HTML.

Кейс 8: Покрытие кода тестами

Задание:

1. Использовать плагин `pytest-cov` для измерения покрытия кода тестами.
2. Настроить генерацию отчета о покрытии кода.

Ожидаемый результат:

- Покрытие кода измерено и сгенерирован отчет.

Кейс 9: Параллельное выполнение тестов

Задание:

1. Использовать плагин `pytest-xdist` для параллельного выполнения тестов.
2. Настроить и запустить тесты в параллельном режиме.

Ожидаемый результат:

- Тесты выполняются параллельно, что ускоряет процесс тестирования.

Кейс 10: Интеграция с CI/CD пайплайном

Задание:

1. Настроить выполнение тестов с использованием Pytest в CI/CD пайплайне (например, на GitHub Actions или GitLab CI).
2. Добавить шаги для установки зависимостей, запуска тестов и генерации отчетов.

Ожидаемый результат:

- Тесты выполняются автоматически в CI/CD пайплайне при каждом коммите.