|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | МИНОБРНАУКИ РОССИИ | МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение*  *высшего образования*  ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** | *овательное учреждение*  *высшего образования*  ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** | *Федеральное государственное бюджетное образовательное учреждение*  *высшего образования*  ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |



Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

ПРАКТИЧЕСКАЯ РАБОТА №3

по дисциплине

**«Тестирование и верификация программного обеспечения»**

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы ИКБО-36-22 | Утенков Ю. Ю. |
| **Проверил:** | ассистент Петрова А. А. |

Москва 2024 г.

СОДЕРЖАНИЕ

[ЦЕЛЬ РАБОТЫ 3](#_Toc178365701)

[РЕАЛИЗАЦИЯ МЕТОДОЛОГИИ TDD 3](#_Toc178365702)

[Этап №1. 3](#_Toc178365703)

[Этап №2. 5](#_Toc178365704)

[Этап №3. 5](#_Toc178365705)

[Этап №4. 7](#_Toc178365706)

[РЕАЛИЗАЦИЯ МЕТОДОЛОГИИ BDD 8](#_Toc178365707)

[Этап №1. 9](#_Toc178365708)

[Этап №2. 9](#_Toc178365709)

[Этап №3. 13](#_Toc178365710)

[Этап №4. 13](#_Toc178365711)

[ЗАКЛЮЧЕНИЕ 13](#_Toc178365712)

# ЦЕЛЬ РАБОТЫ

Ознакомится с понятиями и примерами по методологиям TDD и BDD. Вариант 22: Генератор мелодий по заданным аккордам.

# РЕАЛИЗАЦИЯ МЕТОДОЛОГИИ TDD

*TDD, или Test-Driven Development* (Разработка, управляемая тестами), - это методология разработки программного обеспечения, которая подразумевает создание тестов для функциональности ПО до того, как эта функциональность будет фактически реализована. TDD представляет собой циклический процесс, который помогает разработчикам создавать высококачественное, надежное ПО.

## **Этап №1.**

В данном этапе необходимо создать тесты для будущего приложения. На рисунках 1 – 2 представлены тесты с использованием библиотеки pytest.

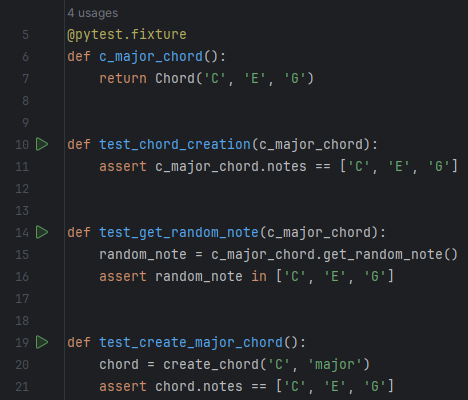


Рисунок 1 – тесты для будущего приложения, часть 1

1. Первый тест проверяет правильность создания аккорда, корректность что ноты аккорда соответствуют ожидаемым ('C', 'E', 'G'). Задаётся с помощью фикстуры с\_major\_chord.
2. Второй тест проверяет функцию получения случайной ноты из аккорда, который получаем из фикстуры с\_major\_chord, проверяется, что случайная нота находится внутри заданного аккорда.
3. Третий тест проверяет функцию создания мажорного аккорда. Вызывается функция create\_chord() с параметрами 'C' (корневая нота) и 'major' (качество), далее проверяется, что созданный аккорд имеет правильные ноты для мажор ('C', 'E', 'G').

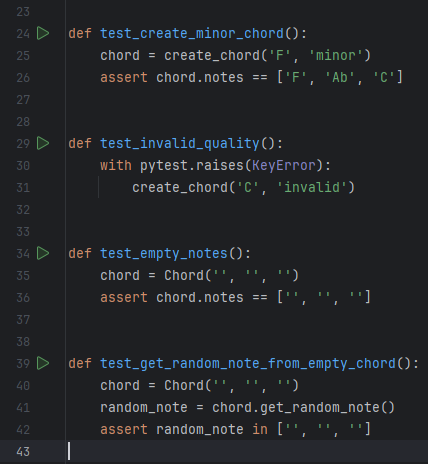


Рисунок 2 – тесты для будущего приложения, часть 2

1. Четвёртый тест аналогичен предыдущему, но проверяет создание минорного аккорда.
2. Пятый тест проверяет обработку некорректных данных. Использует контекстный менеджер pytest.raises() для проверки исключений. В случае ввода пользователем некорректной информации, выбрасывается исключение KeyError.
3. Шестой тест проверяет граничный случай с пустыми нотами. Создает аккорд с пустыми строками вместо нот. Проверяет, что класс Chord корректно инициализируется даже с пустыми данными. Убеждается, что структура аккорда сохраняется даже при отсутствии реальных нот.
4. Седьмой тест также проверяет работу с пустым аккордом. Создает аккорд с пустыми нотами, далее вызывает метод get\_random\_note() для этого аккорда, проверяет, что полученная нота является пустой строкой, после убеждается, что функция случайного выбора работает корректно даже с пустым набором нот.

## **Этап №2.**

В данном этапе запускаются тесты, но так-как классы, методы и функции ещё не определены, тесты не будут успешно пройдены (рисунок 3).

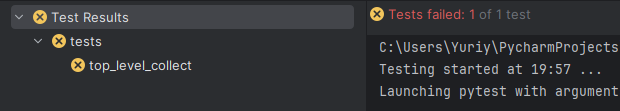


Рисунок 3 – Провал тестов

## **Этап №3.**

Далее реализуются классы, методы и функции, чтобы сделать реализованные раннее тесты успешными. Пример кода представлен на рисунках 4 – 5.

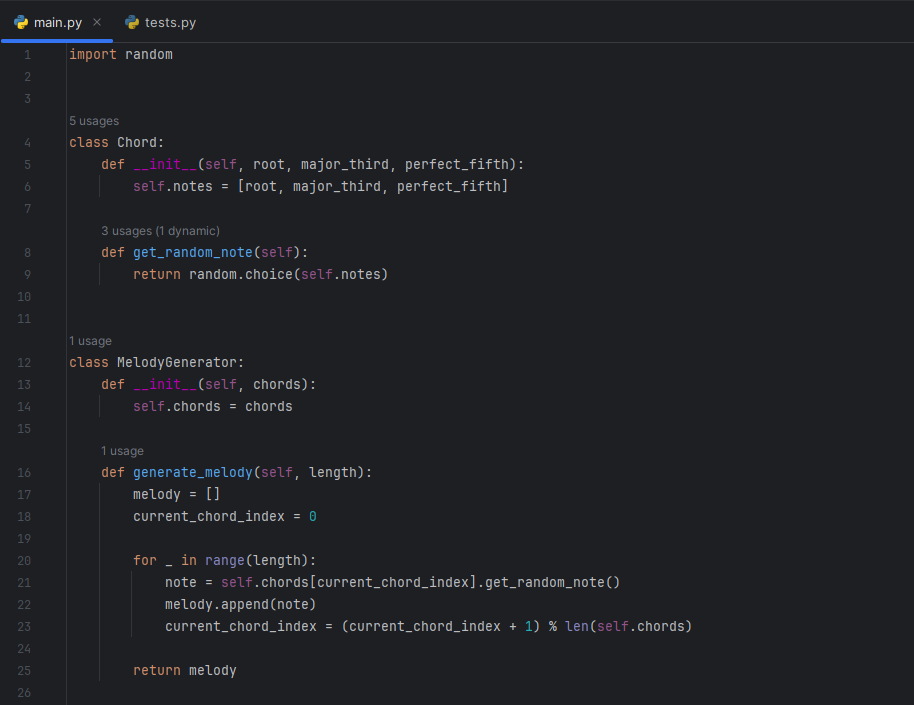


Рисунок 4 – фрагмент кода для выполнения задачи, часть 1

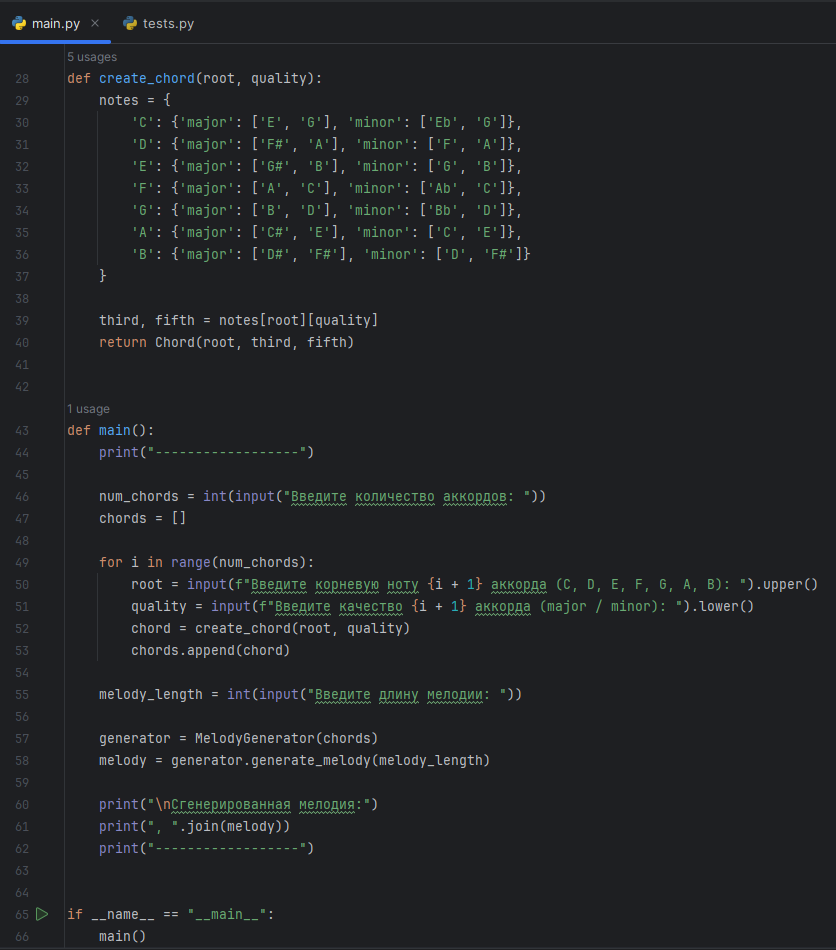


Рисунок 5 – фрагмент кода для выполнения задачи, часть 2

## **Этап №4.**

В данном этапе необходимо убедиться в работоспособности приложения, запустив по новой все раннее написанные тесты для данной программы. Результат выполнения тестов представлен на рисунке 6.

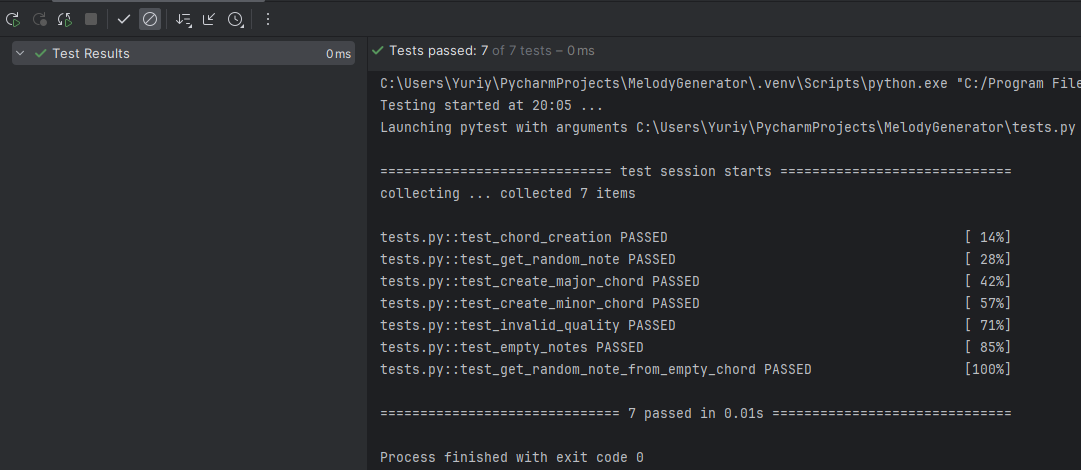


Рисунок 6 – успешное прохождение всех тестов

# РЕАЛИЗАЦИЯ МЕТОДОЛОГИИ BDD

*BDD, или Behavior Driven Development* (Разработка, ориентированная на поведение), - это методология разработки программного обеспечения, которая сосредотачивается на описании поведения программы с точки зрения её пользователей и интересующих сторон. BDD представляет собой эволюцию техники TDD (Test-Driven Development), в которой акцент делается на спецификациях поведения и участии бизнес-аналитиков и представителей заказчика в процессе разработки.

## **Этап №1.**

В данном этапе создаётся описание сценариев BDD для функциональности калькулятора. Ниже приведена таблица с описанием функционала и сценариев в файле melody.feature, специальном файле для описания сценариев и функционала (рисунок 7).

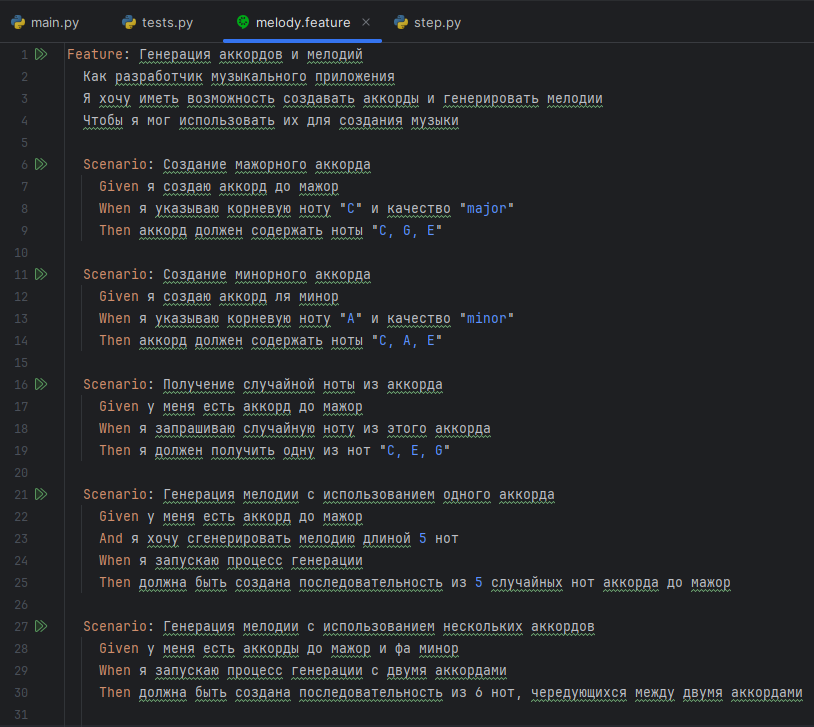


Рисунок 7 – Описание сценариев для генератора мелодий

## **Этап №2.**

На данном этапе создаются автоматизированные тесты для каждого сценария, используя фреймворк для тестирования Behave для Python, который поддерживает методологию BDD (рисунок 10). На рисунке 8 производится установка модуля Behave. На рисунках 9 – 11 приведены рисунки с программным кодом автоматизированных тестов с использованием фреймворка Behave, поддерживающим методологию BDD.

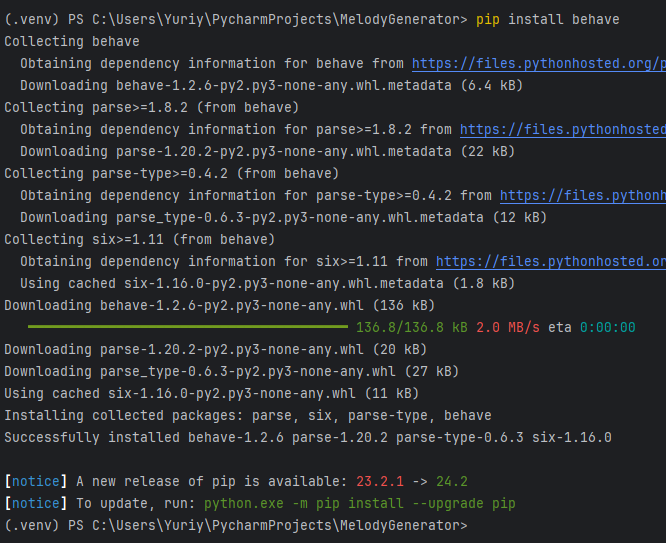


Рисунок 8 – успешная установка модуля Behave

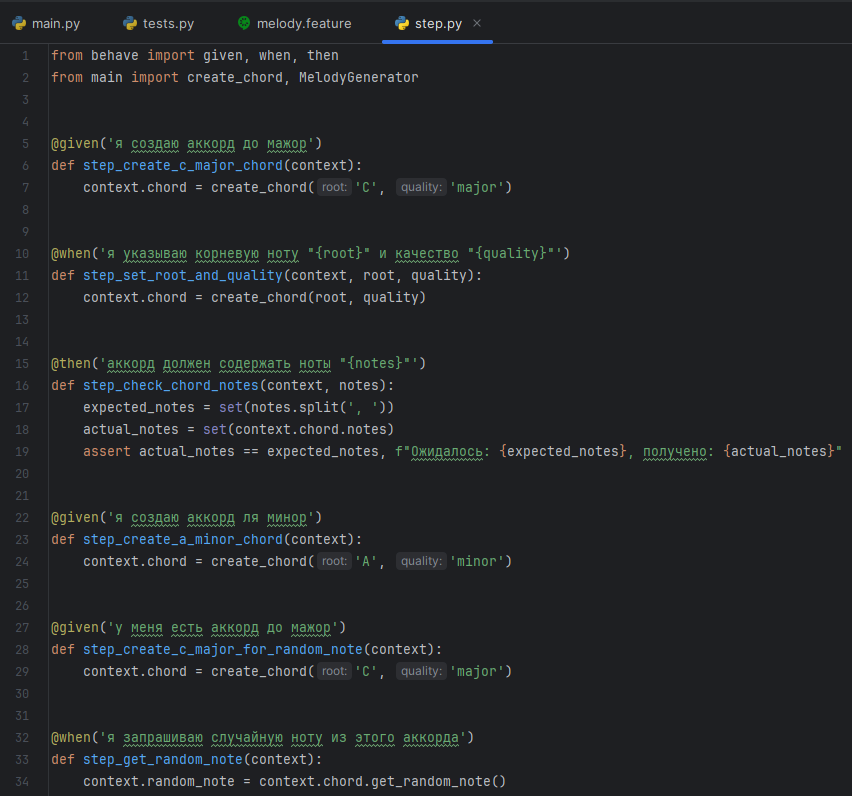


Рисунок 9 – автоматизированные тесты для каждого сценария используя фреймворк Behave, часть 1

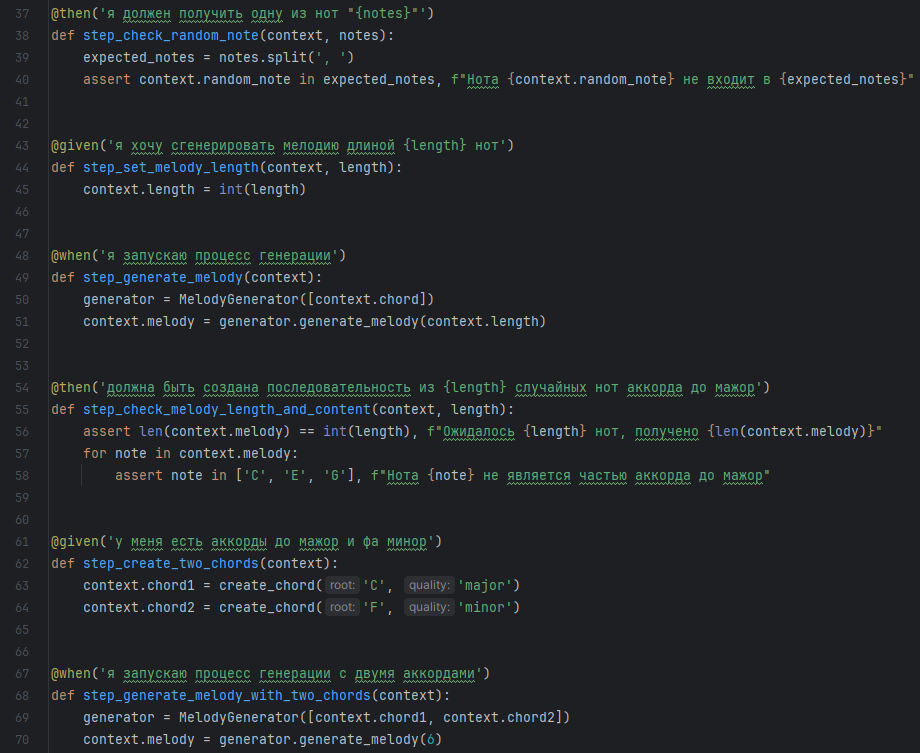


Рисунок 10 – автоматизированные тесты для каждого сценария используя фреймворк Behave, часть 2

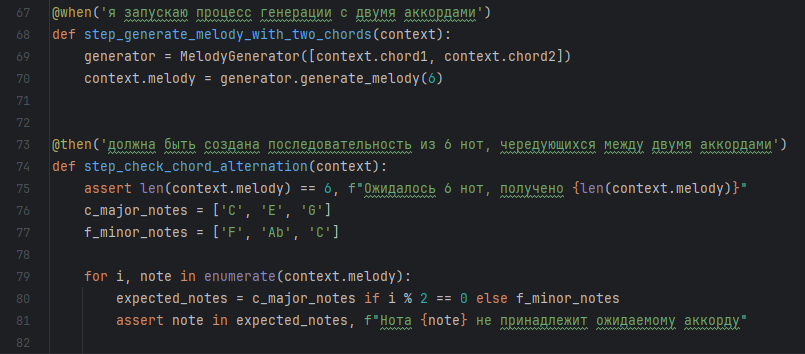


Рисунок 11 – автоматизированные тесты для каждого сценария используя фреймворк Behave, часть 3

## **Этап №3.**

Вся логика работы генератора мелодии была описана в разделе по методологии TDD.

## **Этап №4.**

В данном этапе необходимо убедится в корректности описания файла feature, а также файла step.py. Результат корректности прохождения всех тестов по методологии BDD представлен на рисунке 12.

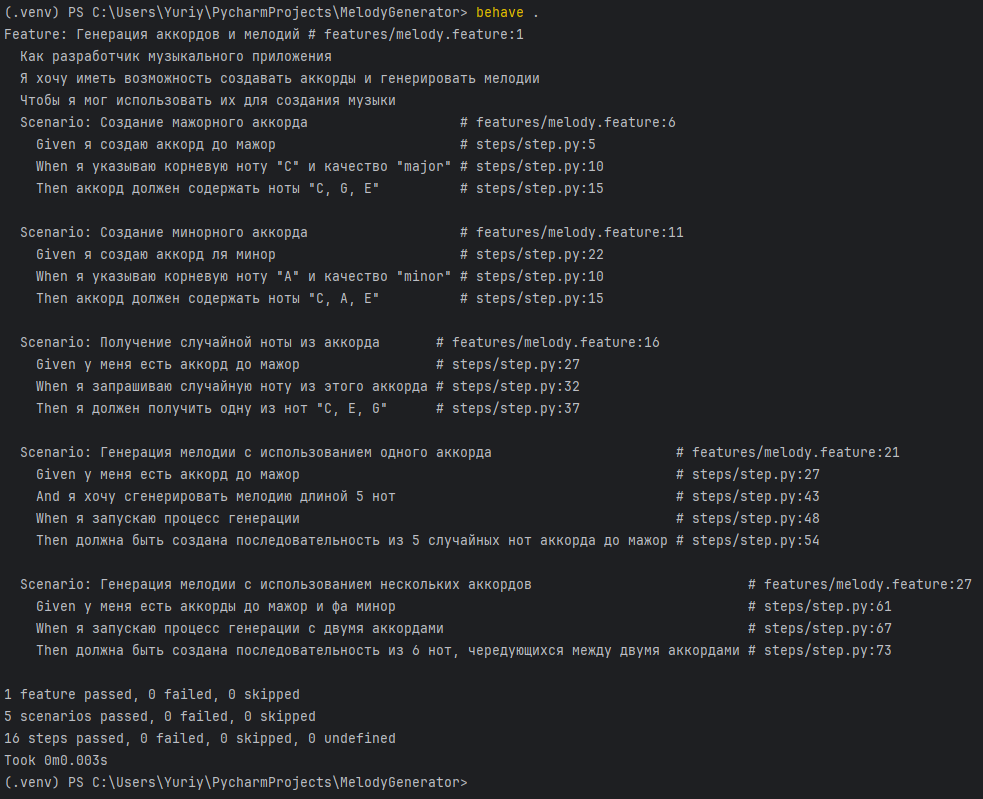


Рисунок 12 – успешное прохождение всех автоматизированных тестов для каждого сценария используя фреймворк Behave

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были изучены новые методологии тестирования: TDD и BDD. Был задействован сторонний фреймворк Behave для создания автоматизированных тестов по методологии BDD.