Краткий отчет по практическому заданию

Студент: Шилов Павел Васильевич

Проект: Консольный планировщик задач на Python

Цель: Разработать консольное приложение, позволяющее управлять личными задачами:

- **Добавление задач с названием, приоритетом и статусом.**
- **Просмотр всех задач.**
- **Удаление задачи по ID.**
- **Изменение статуса задачи (выполнено / не выполнено).**
- **Фильтрация задач по статусу.**
- **Сортировка задач по приоритету.**

Что реализовано:

1. **Архитектура проекта:**
    - **Класс Task: определяет структуру отдельной задачи, включая поля ID, название, приоритет (1-5) и статус (по умолчанию "не выполнено"). Также содержит метод __str__ для форматированного вывода.**
    - **Класс TaskScheduler: инкапсулирует логику управления списком задач. Хранит задачи в списке и управляет следующим доступным ID.**
    - **Функция main_menu и основной блок исполнения (if __name__ == "__main__":): реализуют пользовательский интерфейс через консольное меню, обработку ввода пользователя и вызов соответствующих методов SchedulerManager.**
2. **Основной функционал:**
    - **Добавление задачи: Пользователь вводит название и приоритет (с проверкой корректности приоритета от 1 до 5). ID генерируется автоматически (автоинкремент), статус по умолчанию устанавливается как "не выполнено".**
    - **Просмотр всех задач: Отображение полного списка текущих задач с их ID, названием, приоритетом и статусом.**
    - **Удаление задачи по ID: Реализован поиск задачи по её уникальному идентификатору и последующее удаление из списка.**

- **Смена статуса задачи: Возможность изменить статус задачи (на "выполнено" или "не выполнено") по её ID.**
- **Фильтрация по статусу: Пользователь может выбрать отображение только выполненных или только невыполненных задач.**
- **Сортировка задач по приоритету: Задачи отображаются отсортированными по приоритету в порядке убывания (от высшего к низшему).**
- **Обработка отсутствия задач: При попытке выполнить операции (просмотр, удаление, изменение статуса, фильтрация, сортировка) над пустым списком задач выводятся соответствующие информационные сообщения.**
- **Информационные сообщения: После выполнения операций (добавление, удаление, изменение статуса) выводятся подтверждающие сообщения.**

Дополнительно:

- **Автоматический инкремент next_id для присвоения уникальных ID новым задачам.**
- **После операций сортировки и фильтрации список задач немедленно выводится в консоль в соответствующем виде.**
- **Для удобства пользователя после большинства операций программа ожидает нажатия Enter перед возвратом в главное меню.**

Сложности и решения:

Проблема	Решение
Обеспечение интуитивно понятного и последовательного вывода информации для пользователя после каждой операции, особенно при сортировке или фильтрации.	**Каждая функция, которая изменяет порядок или состав отображаемых задач (например, sort_tasks_by_priority или filter_tasks_by_status), явно вызывает функцию отображения show_tasks с актуальным набором данных.**
Валидация пользовательского ввода, в частности, приоритета задачи.	**При вводе приоритета осуществляется проверка, что значение является целым числом в диапазоне от 1 до 5. В случае некорректного ввода, пользователю выводится сообщение об ошибке с требованием повторного ввода.**

Технологии:

- **Язык: Python 3.9**
- **Среда: Любой текстовый редактор/IDE с интерпретатором Python**
- **Хранение данных: В оперативной памяти (используется стандартный Python list в классе TaskScheduler).**

Результат: Создано полнофункциональное консольное приложение "Планировщик задач", которое полностью соответствует изначальному техническому заданию. Приложение имеет четкую объектно-ориентированную структуру (классы Task и TaskScheduler), что обеспечивает хорошую читаемость кода и возможности для его дальнейшего расширения.