**ДОКУМЕТАЦИЯ ПО ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ; ПРОЕКТ: ПРОГРАММА «КОНВЕРТЕР ВЕЛИЧИН» (Работающая через терминал)**

*Вели документацию: Калачева Виктория, Мамедов Асиман*

* В течение последних трех дней практики мы работали в команде над разработкой небольшого консольного приложения. Наша группа выбрала "Конвертер величин" на Python, поддерживающий преобразование единиц температуры, длины и веса.

**Распределение задач:**

* Планирование и архитектура – обсуждение функционала, выбор языка (Python), создание репозитория.
* Реализация основных функций генерация пароля, настройка длины, выбор типов символов (цифры, буквы, спецсимволы).
* Документация – написание README.md, описание проекта и инструкции по запуску.
* Code Review – проверка кода через Pull Requests, исправление ошибок.
* Тестирование – проверка корректности работы генератора.

**Документация к коду "Конвертер величин"**

Общее описание

Этот скрипт представляет собой консольный конвертер величин, который позволяет преобразовывать значения между различными единицами измерения в следующих категориях:

- Длина

- Масса

- Объем

- Температура

Программа имеет интерактивный интерфейс и работает до тех пор, пока пользователь не введет 'q' для выхода.

Структура кода

Основная функция

`unit\_converter()` - главная функция, содержащая всю логику работы конвертера.

Словарь преобразований

`conversions` - словарь, содержащий коэффициенты преобразования для каждой категории (кроме температуры). Коэффициенты заданы относительно базовых единиц:

- Длина: метры (м)

- Масса: килограммы (кг)

- Объем: литры (л)

Особый случай

Температура обрабатывается отдельно, так как требует специальных формул преобразования между шкалами Цельсия (C), Фаренгейта (F) и Кельвина (K).

Логика работы

1. Выводится приветственное сообщение с доступными категориями.

2. Пользователь выбирает категорию для конвертации.

3. Для температуры:

- Ввод значения и единиц измерения (исходной и целевой)

- Преобразование через промежуточное значение в градусах Цельсия

4. Для других категорий:

- Вывод доступных единиц измерения

- Ввод значения и единиц измерения

- Преобразование через базовую единицу

5. Результат выводится с точностью до 6 знаков после запятой (для температуры - до 2 знаков).

Доступные единицы измерения

Длина

- м (метры)

- км (километры)

- см (сантиметры)

- мм (миллиметры)

- миля (мили)

- ярд (ярды)

- фут (футы)

- дюйм (дюймы)

Масса

- кг (килограммы)

- г (граммы)

- мг (миллиграммы)

- т (тонны)

- фунт (фунты)

- унция (унции)

Объем

- л (литры)

- мл (миллилитры)

- м³ (кубические метры)

- галлон (галлоны)

- пинта (пинты)

- кварта (кварты)

Температура

- C (градусы Цельсия)

- F (градусы Фаренгейта)

- K (Кельвины)

Обработка ошибок

Программа обрабатывает следующие ошибки:

- Неверный ввод категории

- Неверные единицы измерения

- Некорректный числовой ввод (не-числовые значения)

Запуск программы

Программа запускается при выполнении скрипта напрямую (через `if \_\_name\_\_ == "\_\_main\_\_":`).

**Анализ приобретенных навыков**

* Командная разработка: научился распределять задачи, согласовывать изменения через Git.
* Работа с Git: активное использование веток, Pull Requests, разрешение конфликтов.
* Сode Review: понимание важности проверки кода, улучшение его качества.
* Документирование: написание README.md в Markdown.

Заключение о практике

* Совместная разработка научила эффективно распределять задачи и контролировать качество кода.
* Также работа в команде позволила лучше понять процессы разработки, важность коммуникации и контроля версий. Несмотря на небольшие трудности, проект был завершен успешно. Полученный опыт будет полезен в будущих проектах.

**Распределение работы:**

* Калачева Виктория- Юнит-тесты, документация группы 666
* Поляков Владимир– написание кода программы (при поддержке DeepSeek), тестирование программы, создание репозитория группы 666
* Мамедов Асиман – обсуждение функционала, Code Review, реализация основных функций пароля

**Проблемы и решения**

* Проблема: Конфликты слияния веток Решение: Регулярный git pull --rebase

**Ключевые выводы**

* Git-дисциплина критически важна в командной работе
* Code review помогает находить 30% + ошибок до мержа
* Документирование экономит время при подключении новых участников
* Итог: Проект завершен в срок, все запланированные функции реализованы.