Задание №3

Разработать инструмент командной строки для учебного конфигурационного языка, синтаксис которого приведен далее. Этот инструмент преобразует текст из входного формата в выходной. Синтаксические ошибки выявляются с выдачей сообщений.
Входной текст на учебном конфигурационном языке принимается из файла, путь к которому задан ключом командной строки. Выходной текст на языке yaml попадает в файл, путь к которому задан ключом командной строки.

Требования

1. **Пример входного файла**

Многострочные комментарии:
/+
Это многострочный
комментарий
+/
Массивы:
#(значение, значение, значение, ...)
Имена:
[_a-z]+
Значения:
• Числа.
• Массивы.
Объявление константы на этапе трансляции:
159
set имя = значение
Вычисление константного выражения на этапе трансляции (инфиксная форма), пример:
.[имя + 1].
Результатом вычисления константного выражения является значение.
Для константных вычислений определены операции и функции:

Сложение.

Вычитание.

Умножение.

max().

3. **Тестирование:**
    - Все функции визуализатора зависимостей должны быть покрыты тестами.
    - Для каждой из функций необходимо написать по два теста (позитивный и негативный).

Решение

Проект состоит из следующих компонентов:

4. **Основной Скрипт (`main.py`):**
    - Обработка входных данных (`transform_input_text`): - Функция отвечает за обработку исходного текста, в котором могут присутствовать однострочные и многострочные комментарии, а также константы и словари. Входной текст изменяется следующим образом:
      Однострочные комментарии (начинающиеся с ::) заменяются на формат комментариев в Python (#).
      Многострочные комментарии (заключённые в --[[...]]) также преобразуются в комментарии Python.
      Объявления констант (например, def PI := 3.14;) заменяются на стандартное присваивание (например, PI = 3.14).
      Словари, заданные через dict(...), преобразуются в специальный формат для дальнейшей обработки.
    - **Обработка входных и выходных файлов** (`process_files`):
      Скрипт ожидает два аргумента командной строки: путь к входному файлу и путь к выходному файлу.
      Он читает данные из входного файла, передаёт их в функцию transform_input_text для обработки, а затем сохраняет результат в выходной файл.
5. **Тестовый Набор (`test.py`):**
    - **Тестирование обработки однострочных комментариев:** Проверяется, что комментарии, начинающиеся с ::, корректно преобразуются в стандартный формат комментариев Python.
    - **Тестирование обработки многострочных комментариев:** Проверяется, что многострочные комментарии, начинающиеся с -- [[, правильно преобразуются в формат Python.
    - **Тестирование преобразования констант:** Проверяется, что константы, объявленные через def имя := значение;, заменяются на формат Python с присваиванием (например, PI = 3.14).
    - **Тестирование преобразования словарей:** Проверяется корректная обработка словарей, представленных через dict(...), и их преобразование в формат [dict].
    - **Тестирование игнорирования пустых строк:** Проверяется, что пустые строки игнорируются при обработке входных данных.
    - **Тестирование комбинированных случаев:** Проверяется, что все изменения (комментарии, константы и словари) корректно обрабатываются вместе в одном файле.

Тестирование

Для тестирования инструмента визуализации были созданы следующие компоненты:

6. **Тестовый Набор:**

- Используется модуль `unittest` для проведения тестов.
- Для каждой из функций написаны по два теста

Результаты Тестирования

Все проведенные тесты прошли успешно, подтверждая корректную реализацию команд.