# Config Language Translator

## Описание

Этот инструмент переводит файлы в формате **YAML** в пользовательский учебный конфигурационный язык. Программа поддерживает вложенные структуры, такие как словари и массивы, и обрабатывает булевые значения, строки, числа, массивы и словари.

## Основные возможности

- Поддержка вложенных структур данных.
- Преобразование массивов в формат `{ значение. значение. ... }`.
- Преобразование словарей в формат `dict(имя = значение, ...)`.
- Обработка строк, чисел и булевых значений.
- Легкий запуск через командную строку.

Использование Запуск программы Для запуска программы используйте следующую команду:

```
python config_language.py input.yaml output.config
```

input.yaml — путь к входному YAML-файлу. output.config — путь к выходному файлу с конфигурацией. Пример использования Входной файл: input.yaml

```
system:
  hostname: "server01"
  os: "Linux"
  users:
    - root
    - admin
  services:
    ssh:
      enabled: true
      port: 22
    web:
      enabled: false
```

Запуск программы:

```
python config_language.py input.yaml output.config
```

Результат: output.config

```
def system = dict(
    hostname = @"server01",
    os = @"Linux",
    users = { @"root". @"admin". },
    services = dict(
        ssh = dict(
            enabled = true,
            port = 22
        ),
        web = dict(
            enabled = false
        )
    )
);
```

Тестирование Запуск тестов Для автоматического создания .config файлов из примеров используйте тестовый скрипт:

```
python test_config_language.py
```

Ожидаемый результат После выполнения в папке examples будут созданы файлы .config для каждого входного YAML-файла.

Примеры файлов Пример 1: Настройки системы Входной файл: examples/example1.yaml

```
system:
  hostname: "server01"
  os: "Linux"
  users:
    - root
    - admin
  services:
    ssh:
      enabled: true
      port: 22
    web:
      enabled: false
```

Выходной файл: examples/example1.config

```
def system = dict(
    hostname = @"server01",
    os = @"Linux",
    users = { @"root". @"admin". },
    services = dict(
        ssh = dict(
```

```
            enabled = true,
            port = 22
        ),
        web = dict(
            enabled = false
        )
    )
);
```

Структура проекта

```
project/
│
├── config_language.py          # Основной код транслятора
├── test_config_language.py     # Скрипт для тестирования
├── examples/                   # Примеры входных файлов
│   ├── example1.yaml
│   ├── example1.config
│   ├── example2.yaml
│   ├── example2.config
│   ├── example3.yaml
│   └── example3.config
├── README.md                   # Документация
└── requirements.txt            # Зависимости проекта
Зависимости
Проект использует стандартную библиотеку Python, а также:
PyYAML для обработки файлов YAML.
```

Установите зависимости через pip:

```
pip install -r requirements.txt
```

```
Обработка файла examples/example1.yaml...
Файл examples/example1.config успешно создан.
Обработка файла examples/example2.yaml...
Файл examples/example2.config успешно создан.
Обработка файла examples/example3.yaml...
Файл examples/example3.config успешно создан.

Все конфигурации успешно созданы.
```