МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное учреждение высшего образования
«Чувашский государственный университет И.Н. Ульянова»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники

Кросс-платформенные средства разработки программного обеспечения
Лабораторная работа 5
«Создание и использование функций»

Выполнил:

Студент группы ИВТ-41-20
Галкин Д.С.

Проверил:

Ковалев С.В.

Чебоксары, 2023

Цель работы:

1. Получение навыков создания и использования функций в Python
2. Ознакомление с методами аннотирования типов данных для параметров функций и возвращаемого функцией значения (type hinting)
3. Получение практических навыков установки и использования сторонних модулей Python

Задание для выполнения

В рамках лабораторной работы необходимо модифицировать программу из предыдущей лабораторной работы, которая предназначалась для помощи в изучении иностранного языка. Программа должна быть разбита на функции с целью улучшения читаемости кода. Кроме того, программа должна использовать модуль nltk (https://www.nltk.org/) для того, чтобы предоставить в качестве примера не весь абзац, а отдельное предложение

К программе предъявляются следующие требования:

1. Поиск примеров использования осуществляется во внешнем txt-файле, который должен быть представлен реальной книгой на любом иностранном языке
2. Необходимо реализовать функцию для считывания содержимого файла и разбиения его на отдельные предложения. В качестве параметра функции выступает имя файла, результатом работы является список строк, где каждая строка содержит отдельные предложение
3. Разбиение исходного текста на предложения осуществляется с помощью модуля nltk
4. При необходимости, для улучшения логики работы программы и её читаемости, допускается создание дополнительных функций, например, функции, для проверки того, что какая-либо строка содержит все необходимые слова
5. Параметры всех функций, а также тип возвращаемого значения для каждой функции, должны быть аннотированы
6. Параметры поиска задаются во внешнем файле «request.json», который имеет туже структуру, что и в предыдущей лабораторной работе
7. Результат работы необходимо сохранить в файле «response.json». Данный файл должен быть json- представлением, структуру которого необходимо проработать самостоятельно

Полный текст программы:

```
import json
from typing import List
import nltk
from nltk.tokenize import sent_tokenize

nltk.download('punkt')


class Document:
    def __init__(self, file_name: str, target_words: List, example_minimum_length:
int, example_maximum_length: int,
                 number_of_examples: int):
        self.file_name = file_name
        self.target_words = target_words
        self.example_minimum_length = example_minimum_length
        self.example_maximum_length = example_maximum_length
        self.number_of_examples = number_of_examples


# Функция, которая считывает параметры поиска из файла "request.json"
def read_json(filename: str) -> dict:
    with open(filename, "r") as request_file:
        return json.load(request_file)


# Функция, которая конфигурирует параметры
def config_document(request_data: dict) -> Document:
    return Document(**request_data)


# Функция, которая читает текст с файла книги
def read_text(filename: str) -> str:
    with open(filename, "r", encoding="utf-8") as book:
        return book.read()


# Функция, которая записывает реузльтат в response.json
def write_json(filename: str, data: dict):
    with open(filename, "w", encoding="utf-8") as response:
        json.dump(data, response, ensure_ascii=False, indent=4)

    print("Примеры использовани найдены и сохранены в файле 'response.json!")


# Функция, которая разбивает текст на предложения с помощью nltk
def split_text_into_sentences(text: str) -> List[str]:
    return sent_tokenize(text)


# Функция, которая ищет слова представленные с json
def find_words_sentences(sentences: List[str], param: Document, filename: str):
    examples = []
```

```
    for word in document.target_words:
        for sentence in sentences:
            if (param.example_minimum_length <= len(sentence) <=
param.example_maximum_length) and (
                    word in sentence):
                examples.append(sentence)

    response_data = {
        "examples": examples
    }

    write_json(filename, response_data)


r_filename = "request.json"
w_filename = "response.json"

document = config_document(read_json(r_filename))
book = read_text(document.file_name)
sentences = split_text_into_sentences(book)
find_words_sentences(sentences, document, w_filename)
```