ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РЕСПУБЛИКИ ХАКАСИЯ

«ХАКАССКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 09.02.07

«Информационные системы и программирование»

Практическое занятие №5

Практическая работа №2

Руководитель:

\_\_\_\_\_\_\_\_ А.А. Отыргашева

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, дата)

Выполнил:

Студент группы ИС(ПРО)-31

\_\_\_\_\_\_\_\_ Д. А. Коробкова

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Абакан 2025 г.

**Текст задания**

Вариант 1: Пакет для обработки текста.

1. Создайте пакет text\_processing, который включает два модуля:

- text\_analyzer.py: содержит класс TextAnalyzer, который принимает текст и предоставляет методы для анализа (подсчет слов, поиск самого длинного слова, замена слов по шаблону);

- text\_transformer.py: содержит функции для преобразования текста (перевод в верхний/нижний регистр, удаление стоп-слов, применение map для преобразования слов).

2. В основном скрипте создайте объект TextAnalyzer, продемонстрируйте его методы и используйте функции из text\_transformer.py с применением lambda.

**Программный код модуля text\_analyzer.py**

# Импорт модуля re, предоставляющий операции для работы с текстовыми шаблонами

import re

# Создание класса TextAnalyzer

class TextAnalyzer:

# Создание конструктора класса

def \_\_init\_\_(self, text):

# Присваивание значения параметра текст атрибуту self.text

self.text = text

# Создание функции для подсчета количества слов

def count\_words(self):

# Определение переменной words для поиска всех слов

words = re.findall(r'\b\w+\b', self.text.lower())

# Возвращение количества слов в списке words

return len(words)

# Создание функции для поиска самого длинного слова в тексте

def find\_longest\_word(self):

# Определение переменной words для поиска всех слов

words = re.findall(r'\b\w+\b', self.text.lower())

# Возвращает слово максимальной длины

return max(words, key=len)

# Создание функции для замены всех вхождений шаблона в тексте на указанную замену.

def replace\_words(self, pattern, replacement):

# Возвращение текст, в котором все вхождения pattern заменены на replacement

return re.sub(pattern, replacement, self.text)

**Программный код модуля text\_transformer.py**

# Импорт модуля re, предоставляющий операции для работы с текстовыми шаблонами

import re

# Создание функции, которая возвращает текст в верхнем регистре

def to\_upper(text):

# Возвращение текста в верхнем регистре

return text.upper()

# Создание функции, которая возвращает текст в нижнем регистре

def to\_lower(text):

# Возвращение текста в нижнем регистре

return text.lower()

# Создание функции для удаления из текста стоп слов

def remove\_stopwords(text, stopwords):

# Определение переменной words для поиска всех слов

words = re.findall(r'\b\w+\b', text.lower())

# Создание список filtered\_words, содержащего только те слова из списка words, которые не содержатся в списке stopwords

filtered\_words = [word for word in words if word not in stopwords]

# Возвращение строки, содержащую объединение слов из filtered\_words обратно в строку

return ' '.join(filtered\_words)

# Создание функции для использования map

def apply\_map(text, func):

# Определение переменной words для поиска всех слов

words = re.findall(r'\b\w+\b', text.lower())

# Создание переменной, которая применяет функцию func к каждому слову в списке words с использованием функции map

transformed\_words = map(func, words)

# Возвращение строки, содержащую объединение слов из transformed\_words обратно в строку

return ' '.join(transformed\_words)

**Программный код main.py**

# Импорт классов и функций из модулей text\_analyzer и text\_transformer

from textt\_processing.text\_analyzer import TextAnalyzer

from textt\_processing.text\_transformer import to\_upper, to\_lower, remove\_stopwords, apply\_map

# Конструкция, чтобы следующий блок кода выполнялся только при запуске скрипта напрямую

if \_\_name\_\_ == '\_\_main\_\_':

# Создание переменной text и присваивание ей тестового текста

text = "Это вот пример текста вот для тестирование. Тестирование, вот тестирование, вот 123."

# Создание объекта класса TextAnalyzer

analyzer = TextAnalyzer(text)

# Вывод на печать исходного текста

print("Исходный текст:", text)

# Вызов функции, которая подсчитывает и выводит количество слов в предложении

print("Количество слов:", analyzer.count\_words())

# Вызов функции, которая ищет самое длинное слово и выводит его

print("Самое длинное слово:", analyzer.find\_longest\_word())

# Вызов функции, которая заменяет все вхождения слова 'тестирование' на слово 'пример'

print("Замена слова 'тестирование' на слово 'пример':", analyzer.replace\_words(r'\bтестирование\b', 'пример'))

# Вывод пустой строки

print()

# Вывод строки и вызов функции, которая переводит все слова в тексте в верхний регистр

print("Текст в верхнем регистре:", to\_upper(text))

# Вывод строки и вызов функции, которая переводит все слова в тексте в нижний регистр

print("Текст в нижнем регистре:", to\_lower(text))

# Создание списка stopwords, в котором находятся стоп слова

stopwords = ["вот", "для"]

# Вывод строки и вызов функции, которая удаляет слова из списка stopwords из текста

print("Текст с удаленными стоп-словами:", remove\_stopwords(text, stopwords))

# Создание функции lambda, которая добавляет к каждому слова "\_мяу"

add\_myau = lambda word: word + "\_мяу"

# Вызов функции apply\_map для применения функции lambda

print("Текст с 'мяу', добавленным с помощью lambda:", apply\_map(text, add\_myau))

**Скриншот структуры файлов**

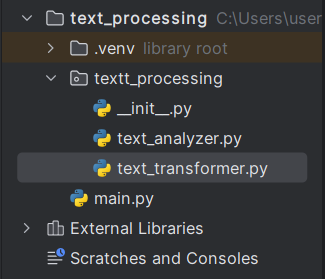


Рисунок 1 – Структура фалов