Ярошевич В.А.

090302-ИСТа-о22

ФИТУ 3 курс

**Лабораторная работа №5**

**РАБОТА С ФАЙЛАМИ. РАЗРАБОТКА СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА. ВЫВОД ФОРМАТИРОВАННЫХ ДАННЫХ В ФОРМАТЕ JSON**

**Цель работы:** изучить работу с файлами с помощью функций из стандартной библиотеки; рассмотреть понятие синтаксического анализа текста и ознакомиться с его применением на языке Python; изучить возможности взаимодействия Python с форматом хранения данных JSON.

**Задание №9:** написать функцию, которая принимает путь к файлу и параметр x, который может являться строкой или списком, и возвращает частоту повторений параметра x в строке. В случае, когда параметром x является список, следует вернуть словарь, в котором в качестве ключей будут искомые строки, а их значениями частота повторений.

**Краткая теория**

**Работа с файлами**

*Python* поддерживает работу с множеством различных типов файлов. Условно их можно разделить на текстовые и бинарные. Текстовые файлы – это те файлы, которые хранят данные в текстовом виде, а бинарные – это, к примеру видео, изображение и прочее.

Python позволяет работать с файлами двумя способами:

− непосредственного открытия и закрытия файла с помощью встроенных функций **open(path, mode) и close();**

− с применением конструкции **with**.

Чтение файла и запись в файл осуществляются с помощью методов **read()** и **write()** соответственно. *Python* позволяет работать с файлом в определенных режимах, передаваемых функции *open* в качестве строкового аргумента для параметра *mode*:

1) «**r**» (Read) – файл открывается для чтения. Если файл не найден, то генерируется исключение *FileNotFoundError*;

2) «**w**» (Write) – файл открывается для записи. В любом случае создается новый. Следовательно, старые данные будут удалены;

3) «**a**» (Append) – файл открывается для дополнительной записи. Если файл отсутствует, то создастся новый. При существовании такого файла, данные запишутся в конец;

4) «**b**» (Binary) – используется для работы с бинарными файлами и применяется вместе с другими режимами («*wb*», «*rb*», «*ab*»);

5) «**+**» – файл открывается одновременно на чтение и запись. Применяется вместе с другими режимами («r+», «a+», «wb+» и т.д.).

Стоит выделить комбинации «r+» и «w+» при таких режимах, оба файла будут открыты на одновременное чтение и запись, но в первом случае при отсутствии файла возникнет исключение, а во втором создастся новый файл.

Как было отмечено ранее, для работы с файлами можно использовать встроенные функции open() и close() и конструкцию *with*. В первом случае стоит учитывать, что во время работы с файлами могут быть сгенерированы различные исключения, что может привести к не выполнению функции close() и соответственно не закрытию файла. Для предотвращения данной ситуации применяется конструкция try, где в блоке *finally* обязательно вызовется функция close(). Однако, чтобы избежать данных дополнительных действий существует конструкция with, которая внутри себя уже реализует обработку исключений и обязательного закрытия файла.

**Модуль Json**

− **dump(obj, file)** – метод предназначен для процесса сериализации (преобразование) объекта obj в формат *Json* и записи его в файл file;

− **dumps(obj)** – метод предназначен для процесса сериализации (преобразование) объекта obj в формат *Json* в виде строки;

− **load(file)** – метод предназначен для процесса десериализации (обратного преобразования) файла file, содержащего *Json* в объект Python;

− **loads(str)** – метод предназначен для процесса десериализации *Json* строки str в объект Python.

Стоит отметить, что методы, описанные выше, содержат большее количество параметров. С ними можно ознакомиться в официальной документации Python 3. На основе разработанного ранее синтаксического анализатора, рассмотрим сериализацию и десериализацию данных.

**Ход работы**

Ознакомившись с материалом, можно приступать к выполнению задания. Данную лабораторную следует разбить на следующие пункты:

1) Импортировать модуль

2) Написать функцию *days\_difference*

3) Абсолютное значение

1. Мы импортируем класс *datetime* из модуля *datetime*.

2. В этой функции мы вычисляем разницу между двумя датами, используя вычитание. Это дает объект *timedelta*, из которого мы можем получить количество дней с помощью атрибута days*.*

3. Мы используем функцию *abs(),* чтобы гарантировать, что возвращаемое значение всегда будет положительным, вне зависимости от порядка переданных дат.

Листинг кода:

def count\_occurrences(file\_path, x):  
 try:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 content = file.read()  
  
 if isinstance(x, str):  
 return content.count(x)  
 elif isinstance(x, list):  
 frequency\_dict = {item: content.count(item) for item in x}  
 return frequency\_dict  
 else:  
 raise ValueError("Параметр x должен быть строкой или списком строк.")  
  
 except FileNotFoundError:  
 return f"Файл по пути '{file\_path}' не найден."  
 except Exception as e:  
 return str(e)  
  
file\_path = 'C:/Users/Public/BlueStacks/fff.txt'  
  
result\_string = count\_occurrences(file\_path, 'искомаястрока')  
print(result\_string)  
  
result\_list = count\_occurrences(file\_path, ['Бордюр', 'Бордюр', 'Лестница', 'Лестница'])  
print(result\_list)

**Результаты выполнения кода:**

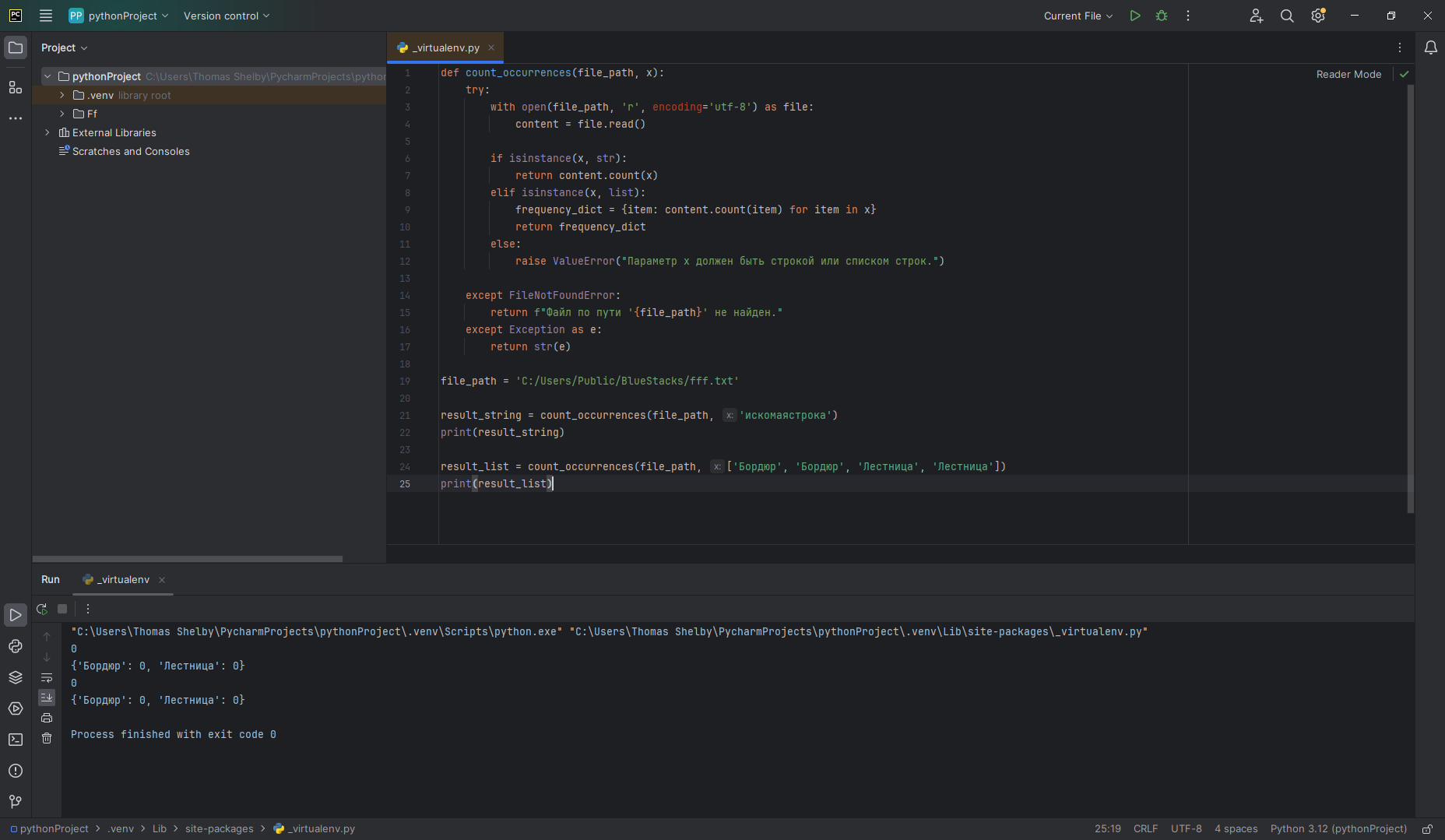


Рис. 1 - результаты работы кода

**Вывод**: в результате работы мы изучили работу с файлами с помощью функций из стандартной библиотеки, рассмотрели понятие синтаксического анализа текста и ознакомились с его применением на языке Python, а также изучили возможности взаимодействия Python с форматом хранения данных JSON.