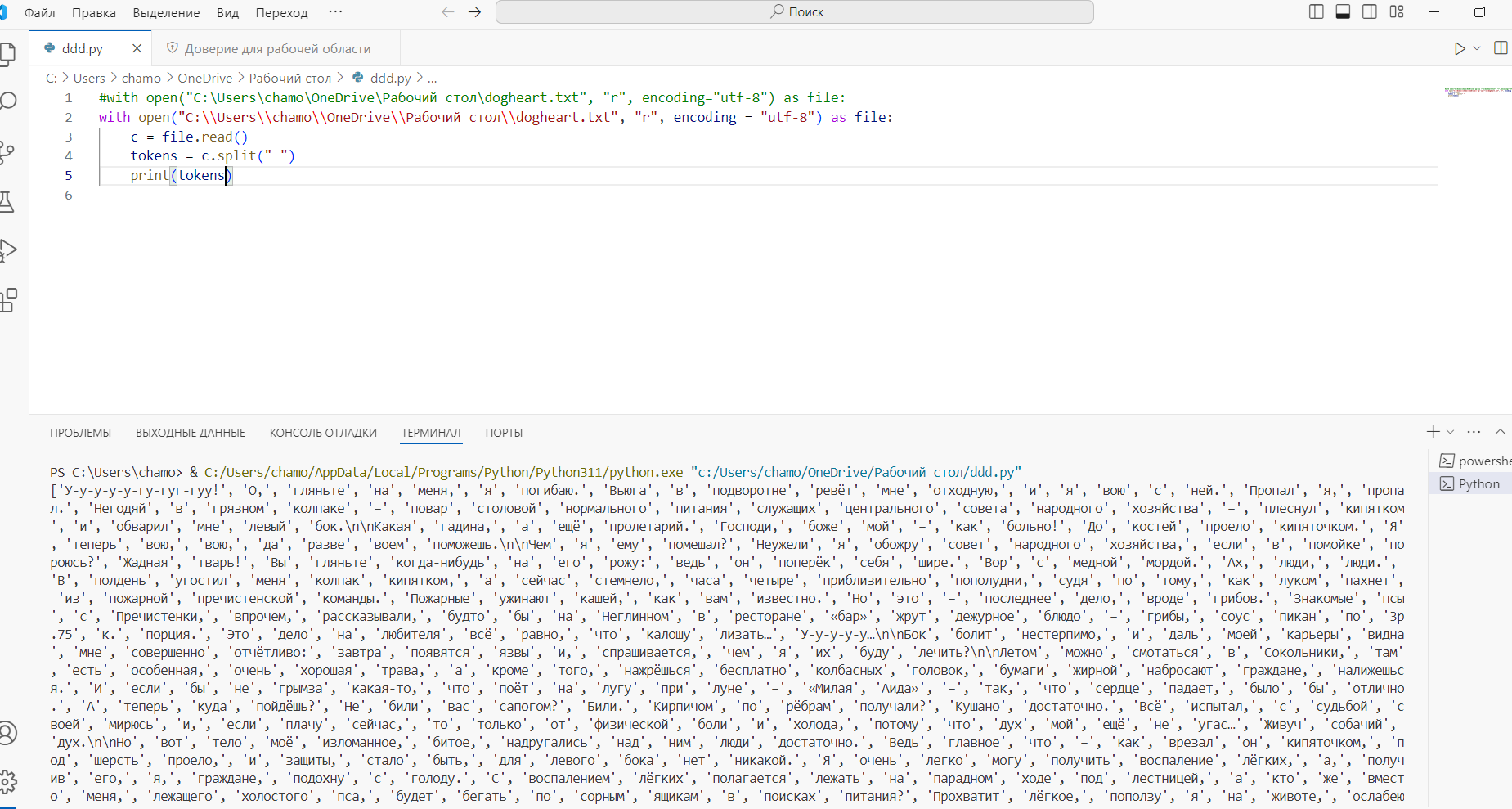
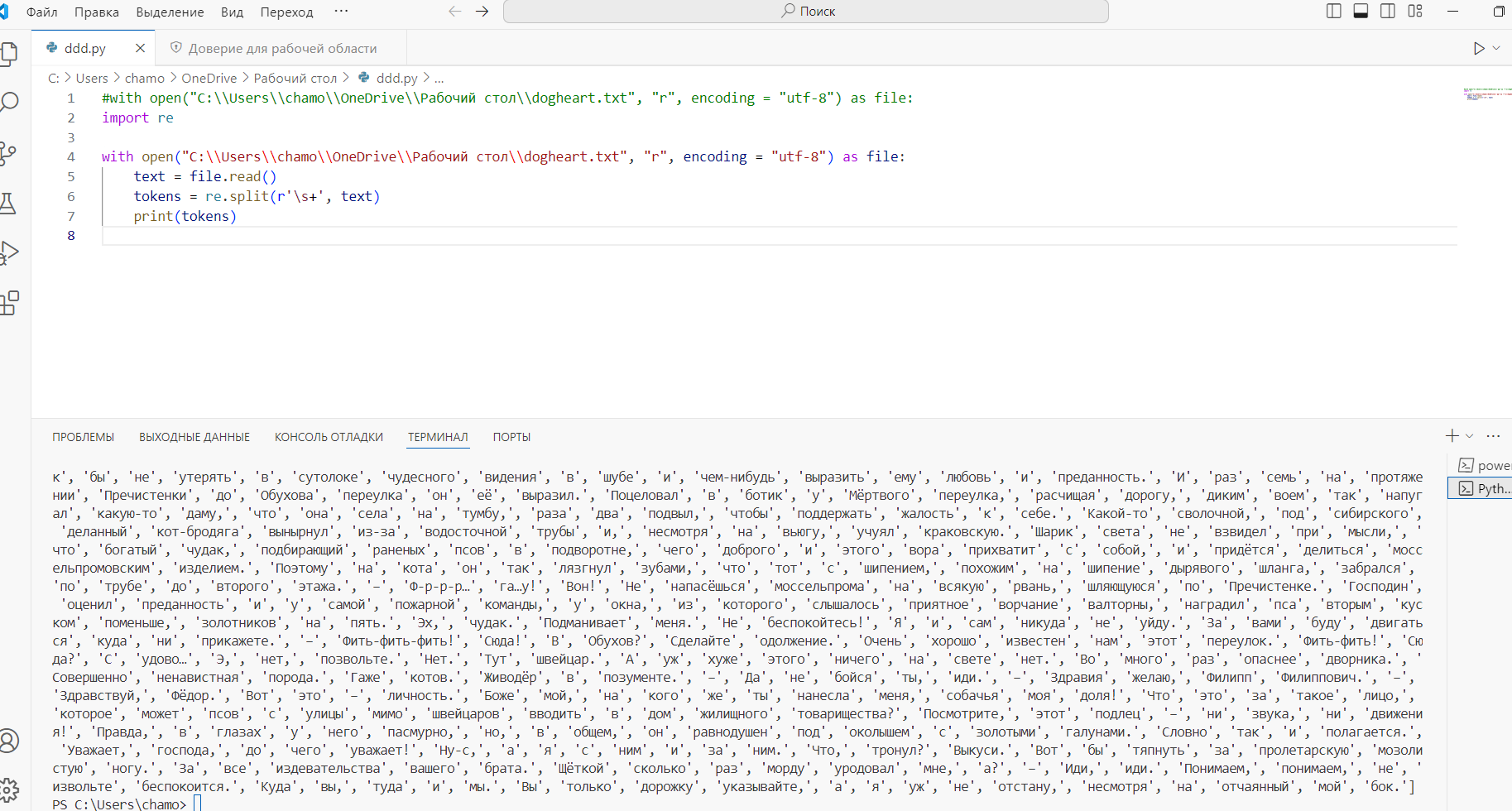
**Уровень 1.**

**- Токенизация.**

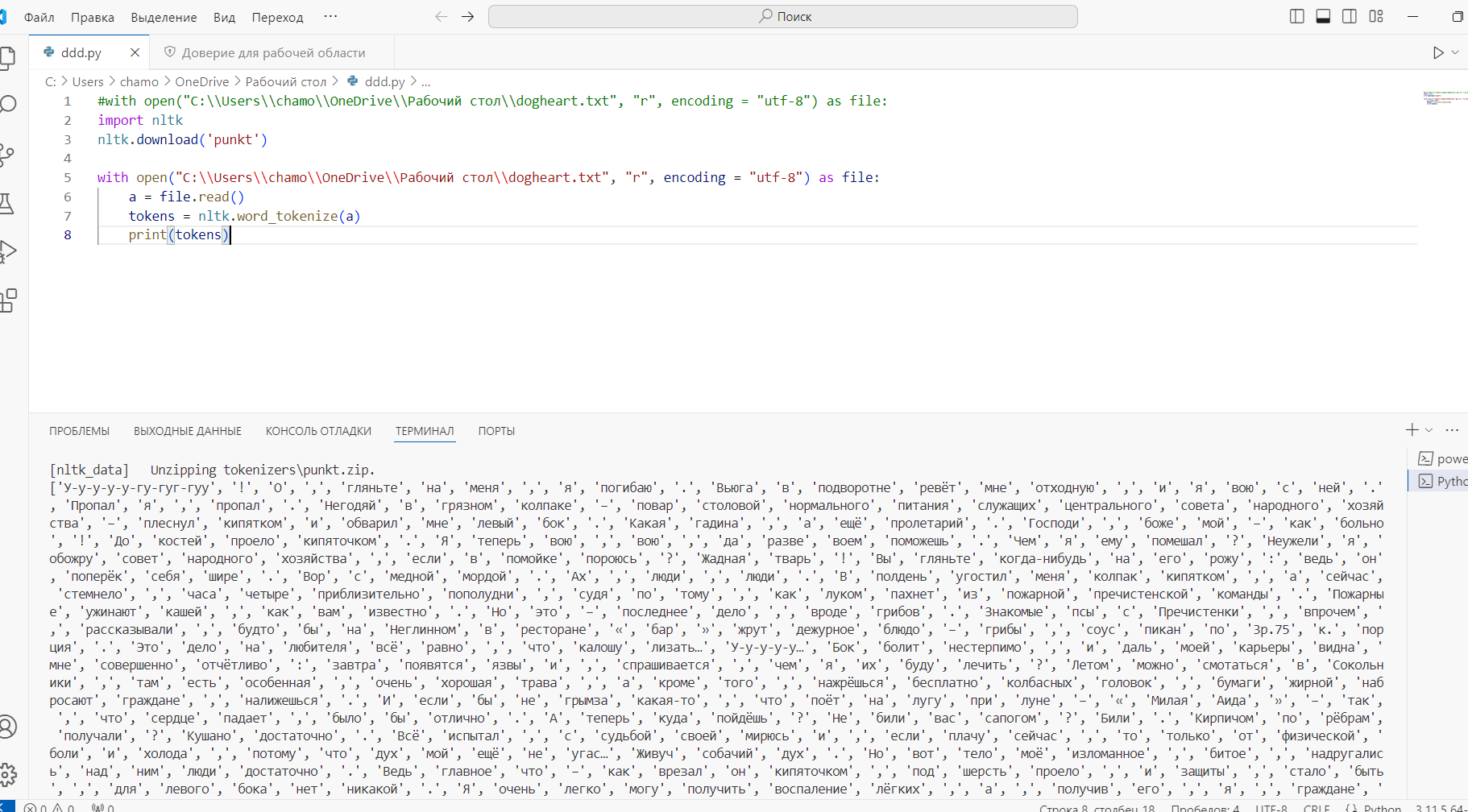
Метод split. содержимое файла разбивается на токены (слова) с использованием пробела в качестве разделителя с помощью метода split(" "). Результат разбиения сохраняется в переменной tokens.



Регулярное выражение. [A-Z]{2,}(?![a-z]): Этот шаблон соответствует последовательности из двух или более заглавных букв, за которыми не следует строчная буква. [A-Z][a-z]+(?=[A-Z]): Этот шаблон соответствует словам, начинающимся с заглавной буквы, за которой следует одна или более строчных букв, а затем снова заглавная буква. ['\w\-]+: Этот шаблон соответствует любому слову или числу, включая символы.

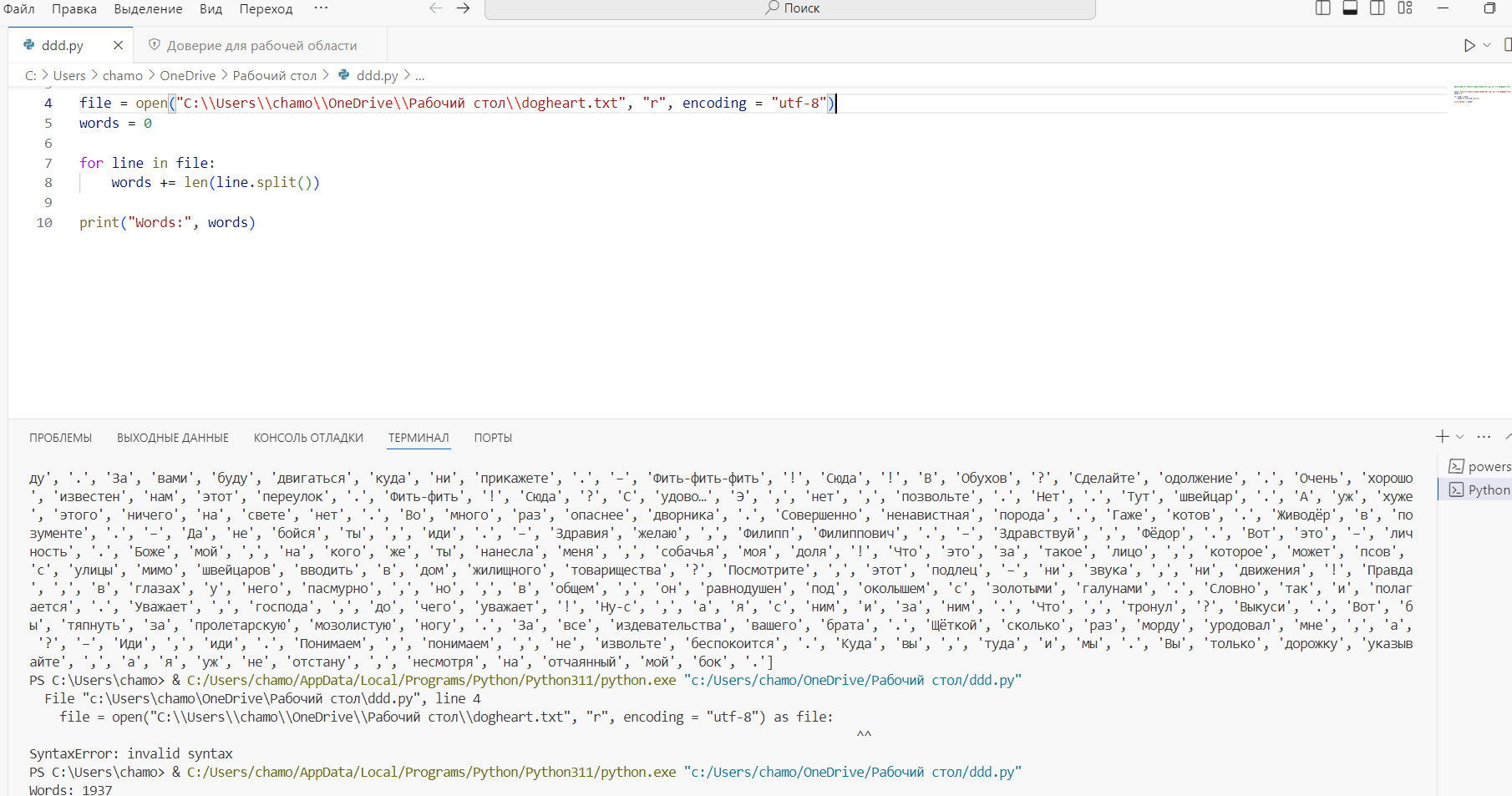


С помощью библиотеки. nltk.download('punkt'): Эта строка загружает токенизатор Punkt из nltk. tokens = nltk.word\_tokenize(a): Содержимое файла токенизируется с использованием функции word\_tokenize из библиотеки nltk.

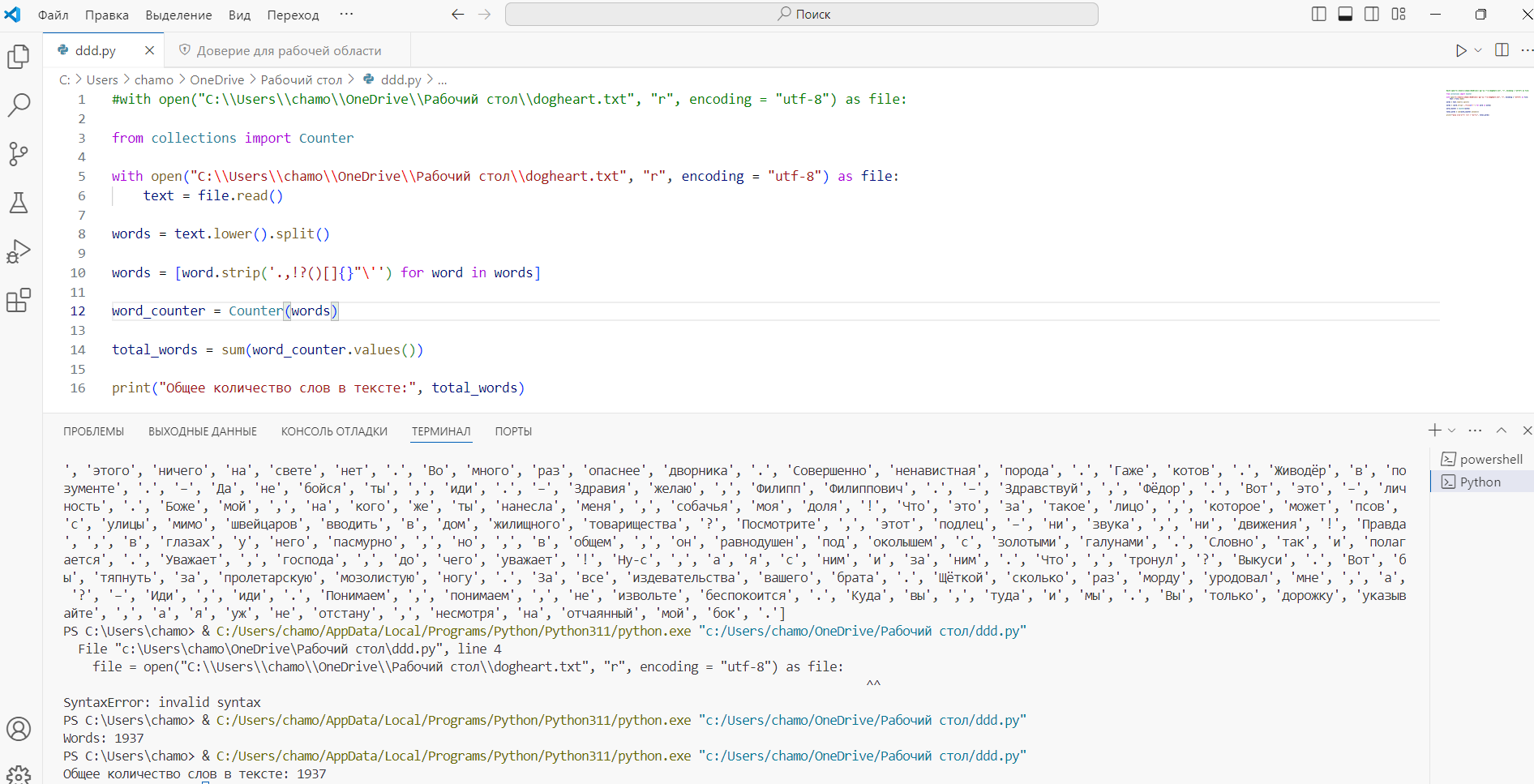


**Подсчёт слов.**

С помощью Len. words = 0: Создается переменная words, которая будет использоваться для подсчета количества слов в тексте. for line in file: words += len(line.split()): Этот цикл проходит по каждой строке в файле. Для каждой строки выполняется следующее действие: строка разбивается на отдельные слова с помощью метода split(), затем количество полученных слов добавляется к переменной words.

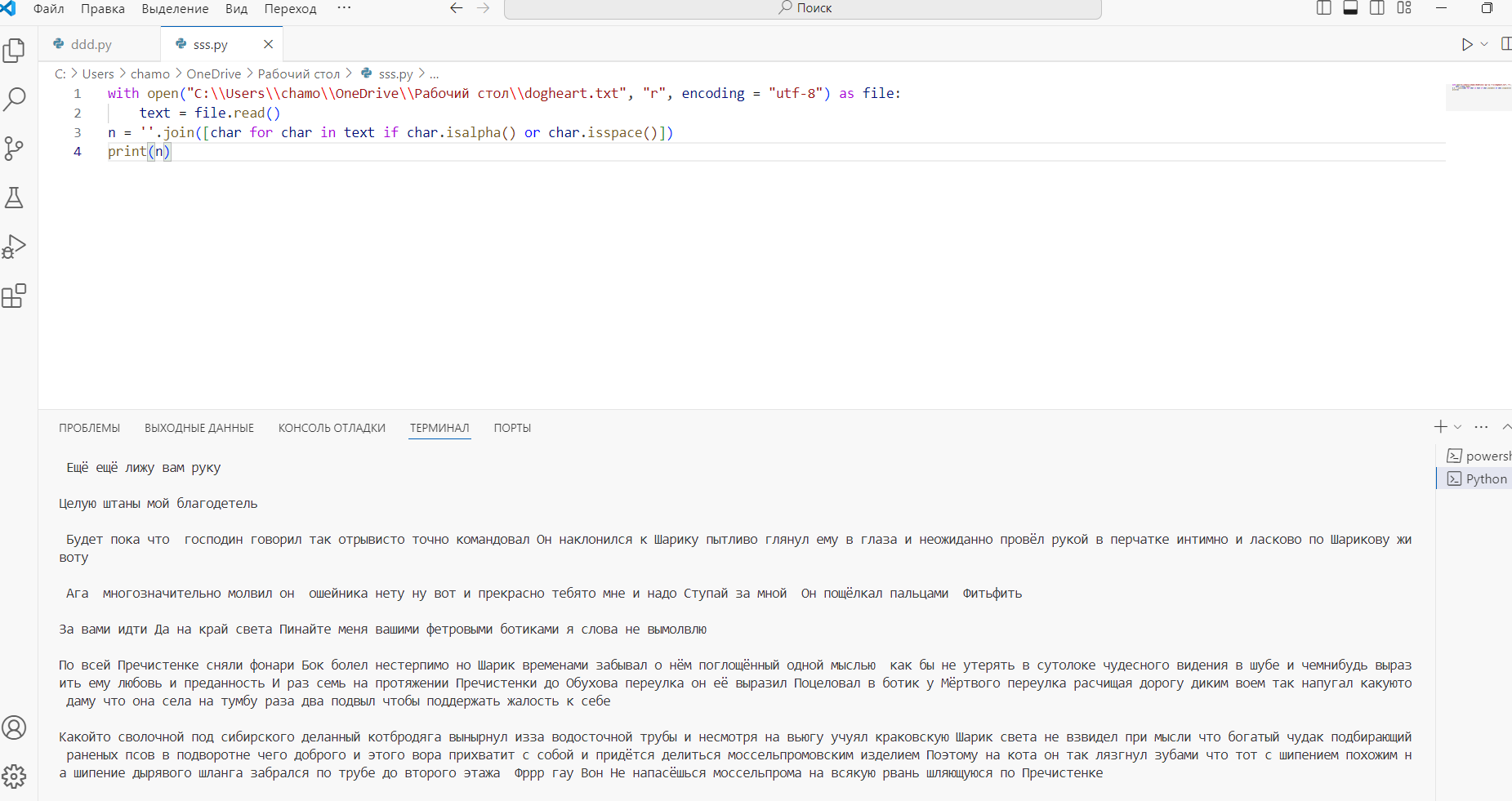


С помощью Counter. words = text.lower().split(): Текст преобразуется в нижний регистр, а затем разбивается на отдельные слова, которые сохраняются в списке words. words = [word.strip('.,!?()[]{}"\'') for word in words]: Каждое слово в списке words очищается от знаков препинания и других символов, указанных в методе strip(). word\_counter = Counter(words): Создается объект Counter, который подсчитывает количество вхождений каждого слова в списке words. total\_words = sum(word\_counter.values()): Суммируются значения счетчика, чтобы получить общее количество слов в тексте без учета повторений.

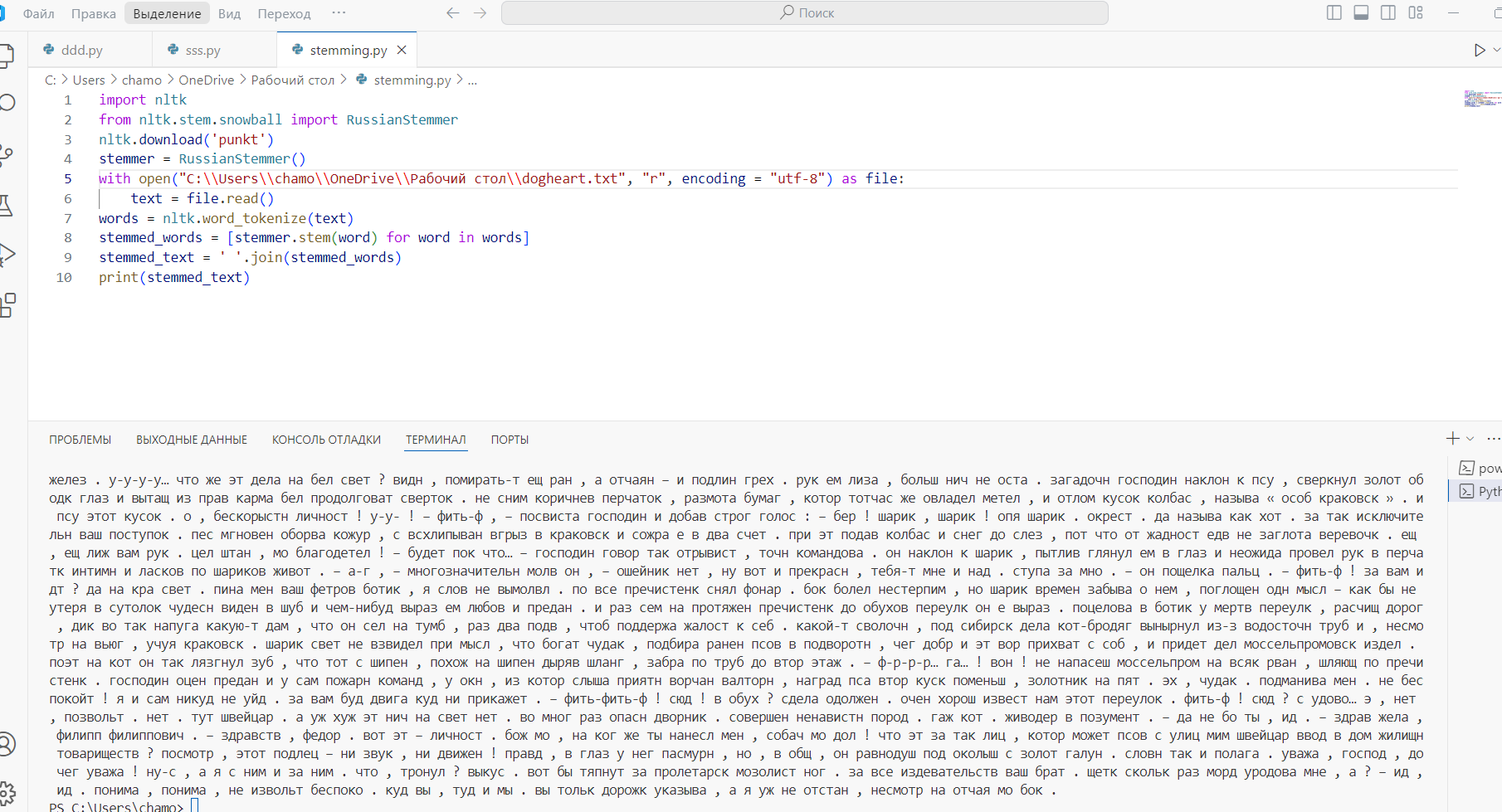


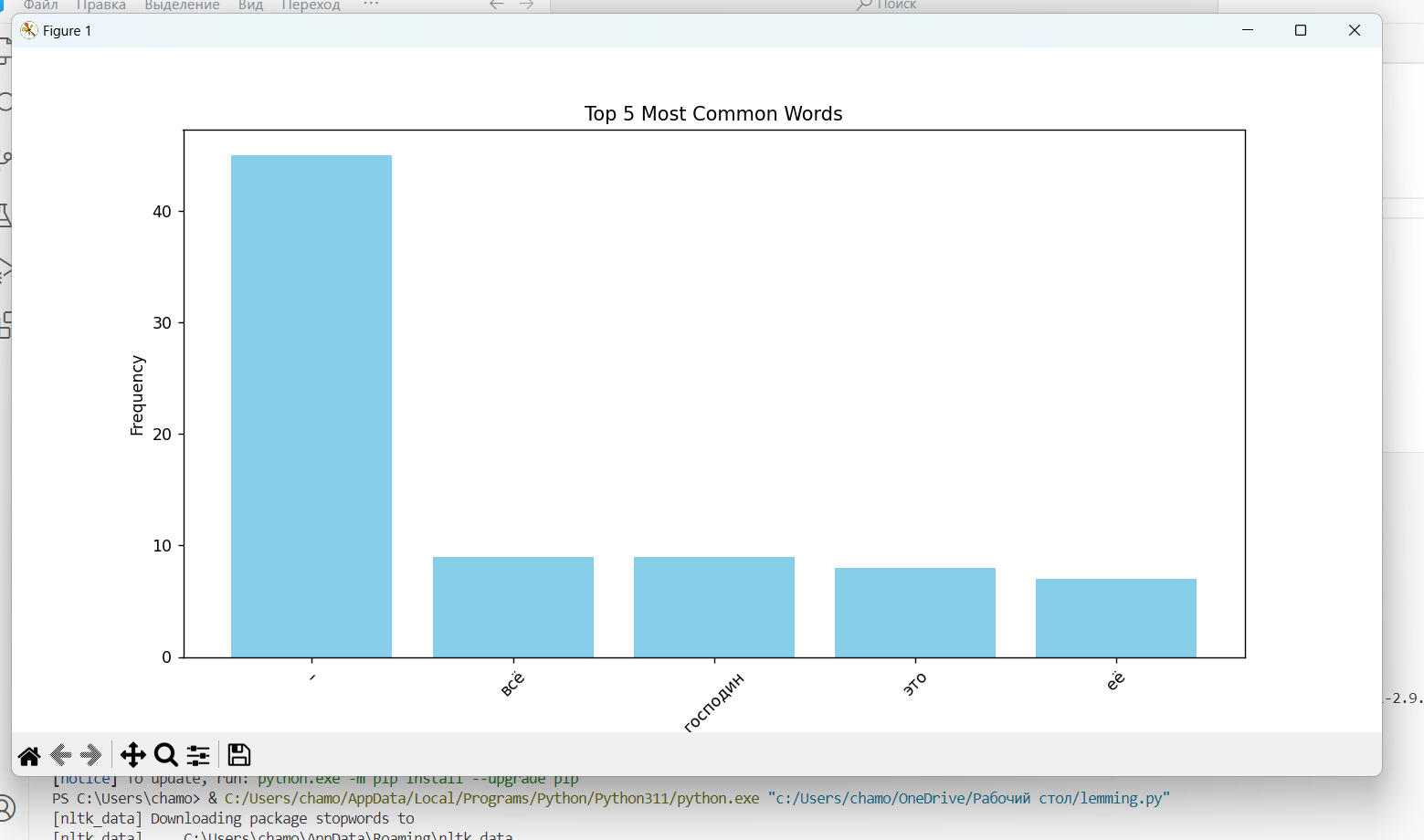
Количество слов используя разные методы не изменилось.

**- Очистка текста от пунктуации.** n = ''.join([char for char in text if char.isalpha() or char.isspace()]): В этой строке происходит фильтрация символов в переменной text. Для каждого символа в тексте проверяется, является ли он буквой (char.isalpha()) или пробелом (char.isspace()). Если символ является буквой или пробелом, он добавляется к строке n.



**- Стемминг.**  words = nltk.word\_tokenize(text): Текст разбивается на отдельные слова. stemmed\_words = [stemmer.stem(word) for word in words]: Каждое слово в тексте стемматизируется с помощью RussianStemmer, т.е. приводится к его основе. stemmed\_text = ' '.join(stemmed\_words): Простемматизированные слова объединяются обратно в строку с пробелами между словами.





def preprocess\_text(text): ...: Определяет функцию preprocess\_text, которая выполняет предварительную обработку текста. text = ''.join([char for char in text if char not in string.punctuation]): Удаляет пунктуацию из текста. words = text.lower().split(): Преобразует текст в нижний регистр и разбивает его на слова. words = [word for word in words if word not in stop\_words]: Удаляет стоп-слова из текста. processed\_text = preprocess\_text(text): Применяет функцию предварительной обработки к тексту и сохраняет результат в переменной processed\_text. most\_common\_words = word\_freq.most\_common(n): Выбирает наиболее часто встречающиеся слова из словаря частот слов.

