Зображення, що містить текст

Автоматично згенерований опис

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Лабораторна робота №2**

Аналіз текстів з використанням мови Python

**Тема:** Попередня обробка тексту за допомогою NLTK

**Варіант:** 1

Виконав Перевірив:

студент групи ІП-11: Тимофєєва Ю. С

Панченко С. В.

Київ 2023

Зміст

[1 Мета лабораторної роботи 6](#__RefHeading___Toc192_2646909948)

[2 Завдання 7](#__RefHeading___Toc194_2646909948)

[3 Виконання 8](#__RefHeading___Toc196_2646909948)

[3.1 Перше завдання 8](#__RefHeading___Toc198_2646909948)

[3.2 Друге завдання 15](#__RefHeading___Toc200_2646909948)

[4 Висновок 18](#__RefHeading___Toc202_2646909948)

# Мета лабораторної роботи

Ознайомитись з представленням тексту Python в та регулярними виразами.

# Завдання

Створити програму,яка виконує завдання відповідно до варіанту, використовуючи бібліотеку NLTK:

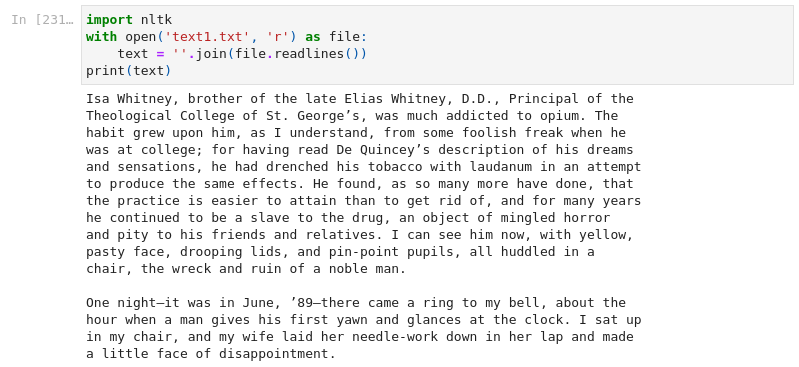
Варіант 1.

Зчитати файл text1. а) Порахувати кількість речень в тексті; б) вивести 10 слів, які зустрічаються найчастіше; в) провести лематизацію слів третього речення.

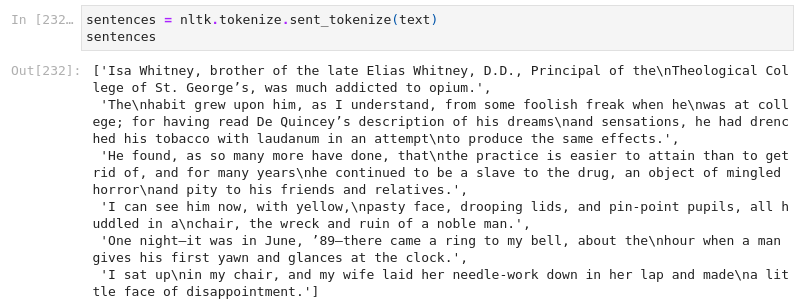
# Виконання

## Перше завдання

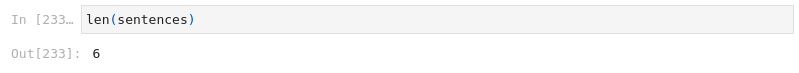
Для початку завантажимо текст, імпортуємо модуль nltk.

  
  
Рисунок 3.1 - Завантаження файлу

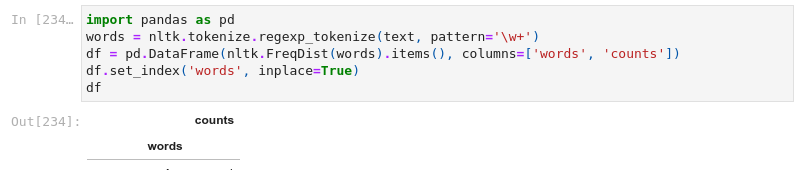
Розіб'ємо текст на речення за допомогою функції tokenize.sent\_tokenize.

  
  
Рисунок 3.2 - Розбиття тексту на речення

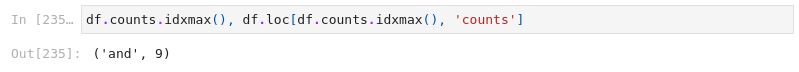
Підрахуємо їхню кількість.

  
  
Рисунок 3.3 - Кількість речень у тексті

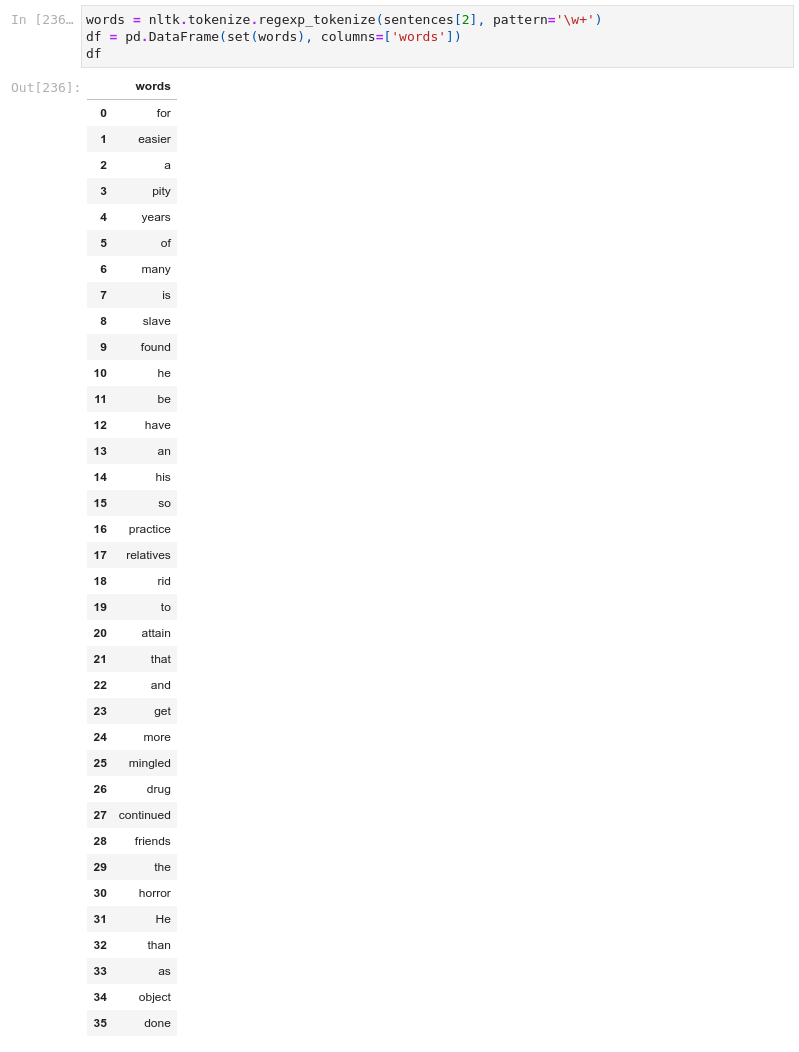
Розділимо текст на слова за допомогою функції nltk.tokenize.regexp\_tokenize. Використаємо шаблон розділення як r'\w+' для розбиття речення на незалежні алфавітні токени. Використаємо nltk.FreqDist для знаходження частоти токенів.

  
  
Рисунок 3.4 - Розбиття тексту на слова

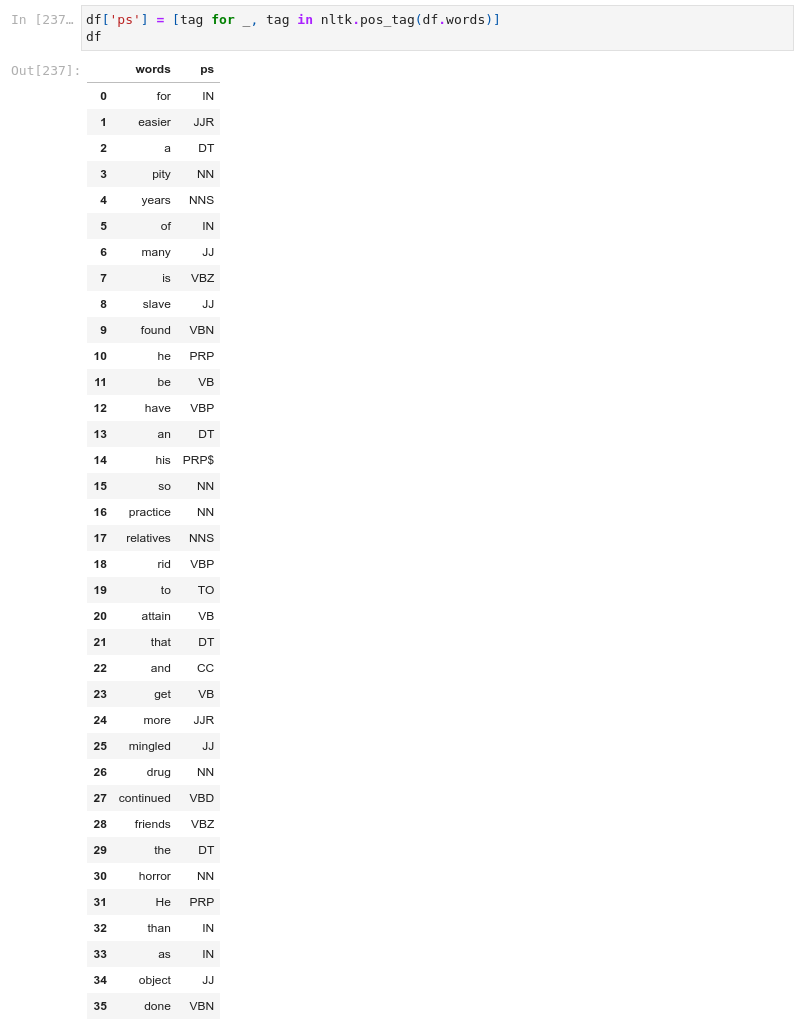
За допомогою idxmax виведемо найбільш уживане слово та кількість разів його використання.

  
  
Рисунок 3.5 - Найбільш уживане слово

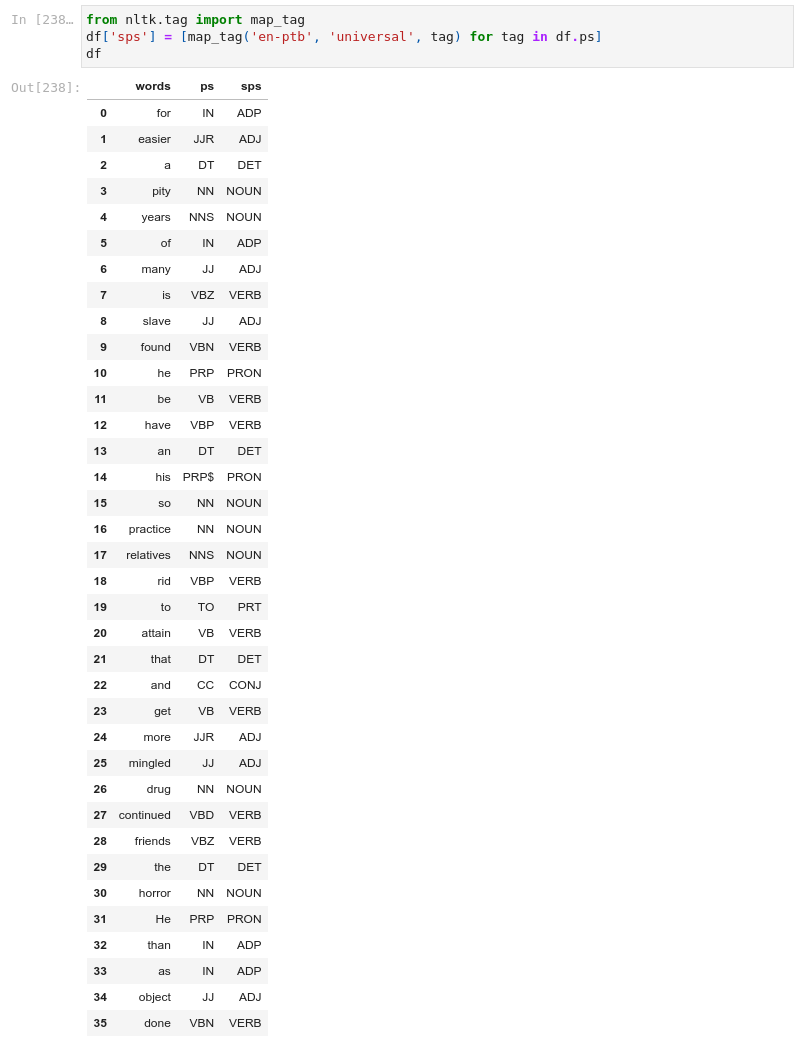
Виділимо слова третього речення, приберемо повтори.

  
  
Рисунок 3.6 - Слова третього речення

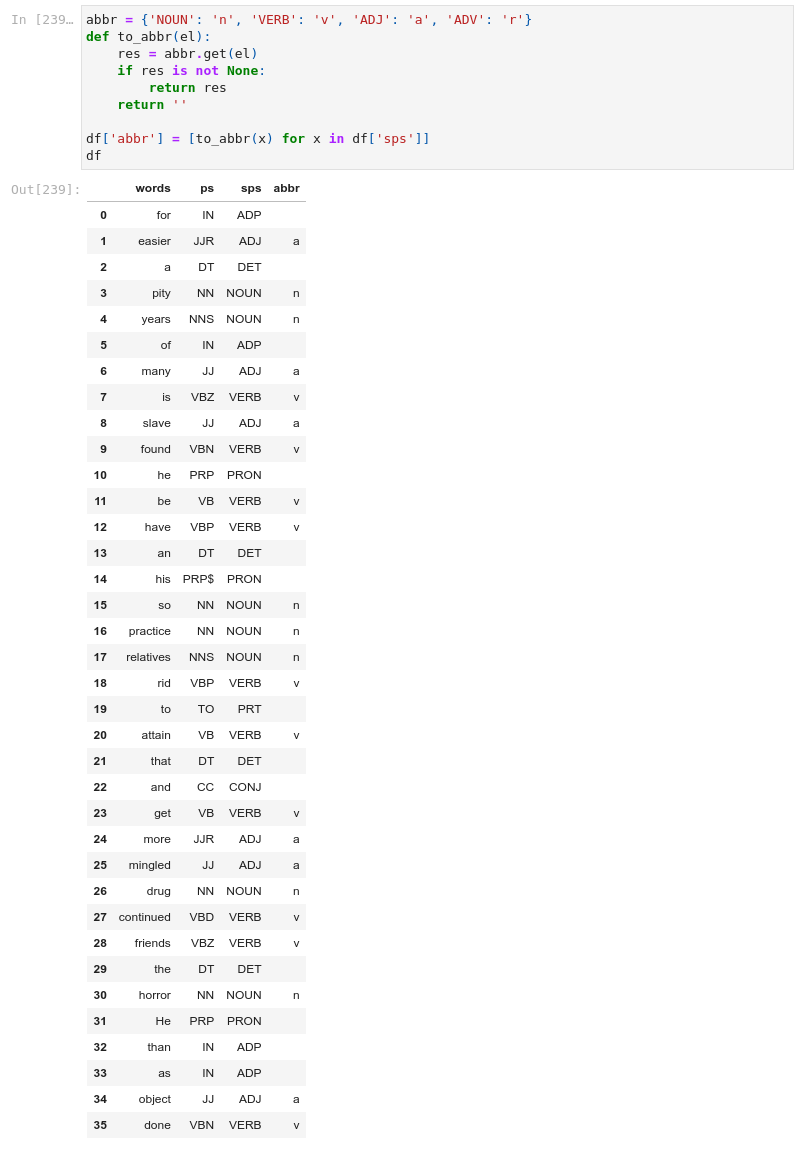
Визначимо частину мови для кожного слова. Використаємо функцію nltk.pos\_tag.

  
  
Рисунок 3.7 - Визначення частини мови

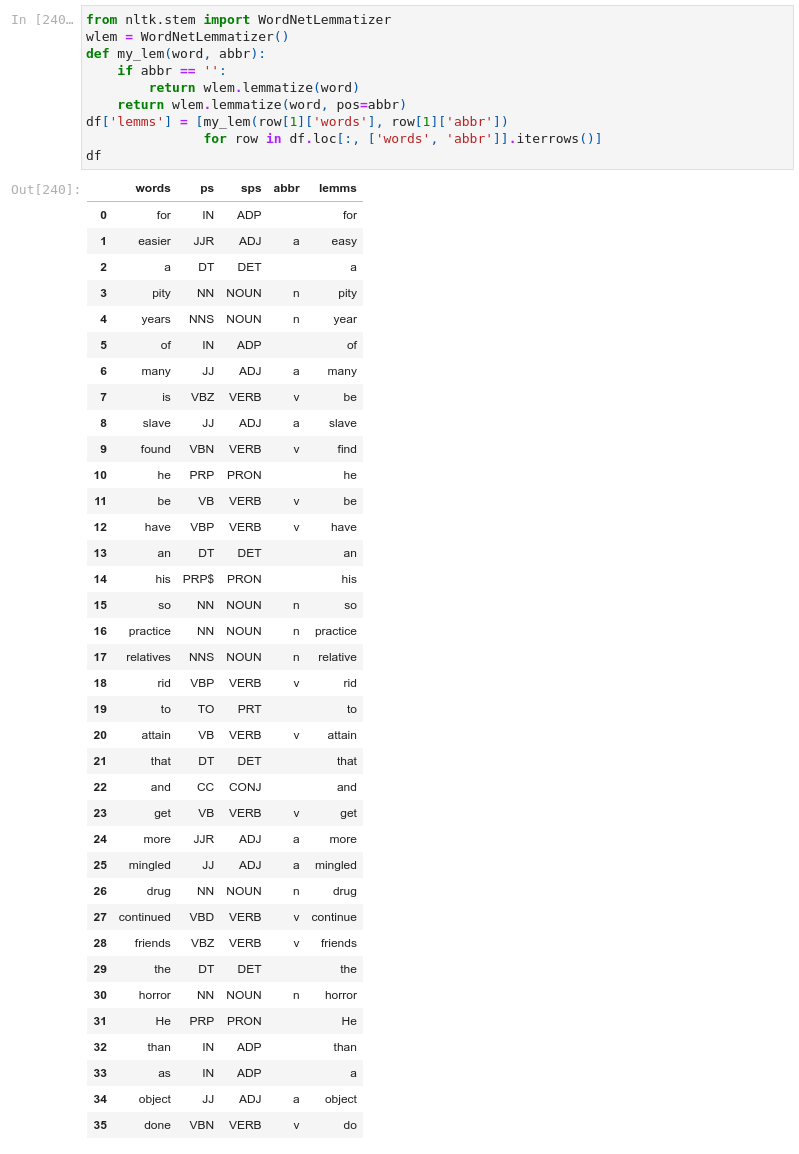
Узагальнимо частини мови до звичайних: noun, adective, verb тощо.

  
  
Рисунок 3.8 - Узагальнення чатин мов

Перетворимо узагальнені частини мови на абревіатури для лематизації.

  
  
Рисунок 3.9 - Приведення загальних частин мов до абревіатур

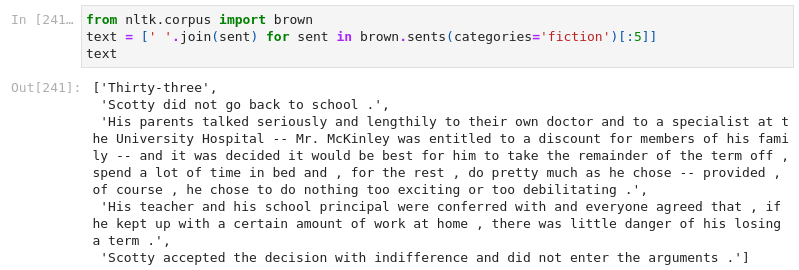
Проведемо лематизацію кожного слова за допомогою методу lemmatize об'єкта класу nltk.stem.WordNetLemmatizer.

  
  
Рисунок 3.10 - Лематизація слів

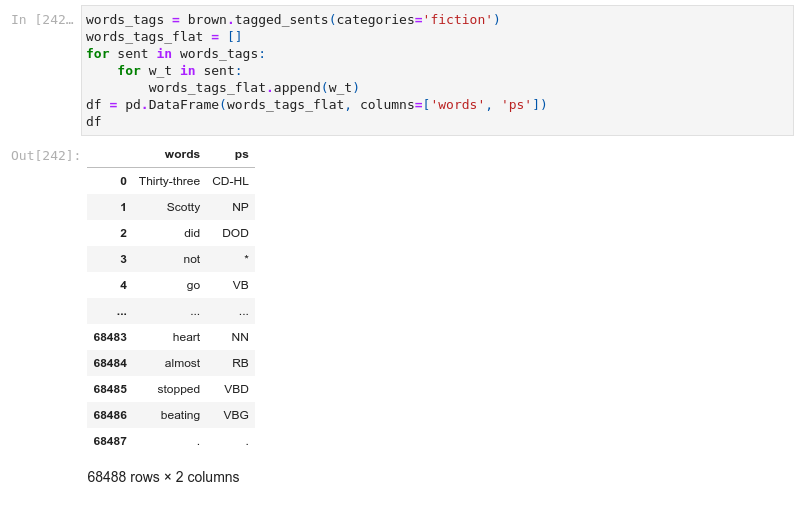
Бачимо, що під час лематизації та тегування сталася одна помилка: "friends" - це "noun", або іменник, тому мало б бути "friend". Однак, "mingled" правильно ідентифікувало, адже це дієприкметник, тобто спрощено "ajective".

## Друге завдання

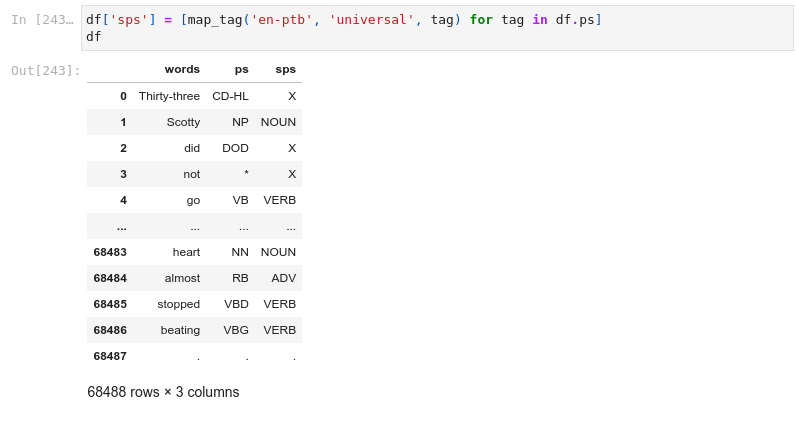
Використаємо корпус brown. Виведемо перші 5 речень.

  
  
Рисунок 3.11 - Виведення перших п'яти речень

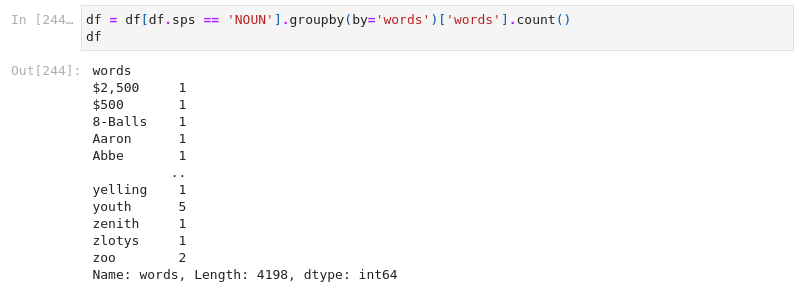
Виведемо протеговані слова з категорії "fiction".

  
  
Рисунок 3.12 - Протеговані слова

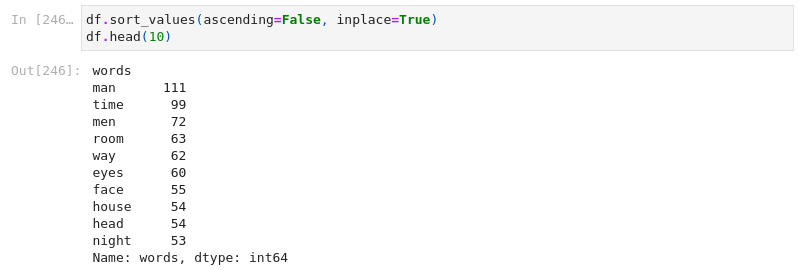
Узагальнимо частини мови.

  
  
Рисунок 3.13 - Узагальнення частин мови

Виділимо лише іменники, згрупуємо їх та застосуємо метод count для підрахунку кожного.

  
  
Рисунок 3.14 - Підрахунок іменників

Визначимо десять іменників, що зустрічаються найчастіше.

  
  
Рисунок 3.15 - Іменники, що зустрічаються найчастіше

# Висновок

Під час виконання даної лабораторної роботи я ознайомився з представленням тексту Python в та регулярними виразами.

У першому завданні було розбито текст на речення, підраховано їхню кількість, розділено на слова та підраховано кількість слів. Також було проведено лематизацію.

У другому завданні зчитано перші п’ять речень відповідної категорії та виведено десять найбільш уживаних іменників.