НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
Факультет Інформатики та обчислювальної техніки
Кафедра Інформаційних систем та технологій

ЛАБОРАТОРНА РОБОТА №1

з дисципліни «Обробка та аналіз текстових даних на Python»

на тему «Обробка текстових даних у Python: основи»

Варіант №2

Виконала: студентка групи ІС-з21
Коломієць Катерина Миколаївна
22.05.2025

Перевірив: асист. Мягкий М. Ю.

Київ - 2025

GitHub: https://github.com/kate-miiets-uni/oatd-py

Теоретичні відомості

Requests – бібліотека python для надсилання HTTP запитів. Має зрозумілий та елегантний синтаксис, що дозволяє зосередитися на логіці взаємодії з API.

Регулярні вирази в Python (скор. “RegEx” від англ. “Regular Expression”) — це послідовність символів, які визначають шаблон для пошуку відповідностей. Для написання регулярних виразів використовуються метасимволи. Метасимволи — це символи, що інтерпретуються особливим чином у механізмі регулярного виразу. Список метасимволів у Python: [] . ^ $ * + ? {} () \ |.

Модуль string у Python - це спеціальний модуль, який містить сталі та функції, що працюють зі строками. Він дозволяє виконувати різні операції зі строками, такі як перевірка типу символів, формування строків та інші.

Хід роботи

1. Імпортуємо бібліотеку requests, за допомогою функції get якої ми можемо надіслати HTTP GET-запит для отримання даних з сервера.

```
import requests
import re

url = 'https://www.bbc.com/ukrainian/articles/cqj77vv07x2o'

# Надсилаємо запит на отримання тексту
response = requests.get(url)
```

2. Робимо перевірку на відповідь сервера на запит. Якщо все добре status_code == 200, якщо status_code == 404 - сторінку не знайдено, status_code == 500 - помилка сервера.

```
# Якщо запит вдалий
if response.status_code == 200:
```

3. У випадку status_code == 200, виконуємо очищення тексту від зайвих пробілів та пунктуації за допомогою функції re.sub() та регулярного виразу r'[^\w\s.]', що означає, що йде пошук та заміна усіх символів, які НЕ є словами, пробілами або крапками, на '' у тексті.

```
    text = response.text

    # Очищення тексту від зайвих пробілів та пунктуації
    clean_text = re.sub( pattern: r'[^\w\s.]',  repl: '', text)
```

Те саме можна зробити за допомогою модуля String, а конкретно його сталої string.punction, яка зберігає рядок символів «!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~».

4. Далі розбиваємо очищений текст на речення по крапці за допомогою методу .split().

```
    # Розбиття тексту за крапками на окремі речення
    sentences = clean_text.split('.')
```

5. Оскільки split розбиває текст на речення по кожній крапці, є шанс утворення порожніх речень якщо в тексті наявні випадки викладення декількох крапок підряд (типу «…»). Для такого випадку використовуємо метод .strip(), який видаляє зайві пробіли на початку та в кінці речення, а оскільки деякі речення можуть бути пустими, strip видалятиме їх взагалі.

```
    # Видаляємо пусті речення
    sentences = [s.strip() for s in sentences if s.strip()]
```

6. У кінці підраховуємо кількість речень у тексті.

```
    # Визначаємо загальну кількість речень
    sentence_count = len(sentences)

    print(f"Загальна кількість речень у тексті: {sentence_count}")
```

7. У випадку status_code == 404 - сторінку не знайдено або status_code == 500 - помилка сервера, отримуємо про це повідомлення.

```
else:
    print("Не вдалося завантажити сторінку.")
```

Результат:

```
Загальна кількість речень у тексті: 7305
```

Висновки

Отже, у ході лабораторної роботи я познайомилася з бібліотеками requests та re, а також навчилася діставати текст з веб-сторінки, навчилася очищувати текст від зайвих пробілів та пунктуації. Також, навчилася розбивати текст на речення за окремими символами. Також дізналася про необхідність додаткової перевірки вдалості відповіді серверу на запит, про необхідність видаляти пусті рядки зі списку поділених речень через властивість функції split додавати пустий рядок у список, у випадках, коли текст закінчується на крапку.

Відповіді на контрольні запитання

1. Як завантажити текст із сайту?

Для завантаження тексту із сайту можна використати бібліотеку requests, яка є дуже популярною для здійснення HTTP-запитів. За допомогою функції get('url') надсилається запит на сервер, у відповідь на який можна отримати текст зі сторінки.

2. Як видалити зайві пробіли в Python?

Зайві пробіли можна прибрати за допомогою метасимволу \s, який прибирає пробіли (пропуск, табуляція, новий рядок), або за допомогою методу .strip(), яке прибирає зайві пробіли на початку та на кінці речення.

3. Що таке пунктуація і як її позбутися?

Пунктуація — це набір знаків, які використовуються в письмі для структурування тексту та надання йому правильного змісту. Вона включає такі символи, як крапка (.), кома (,), знак питання (?), знак оклику (!), двокрапка (:), лапки (" ") та інші.

Якщо потрібно видалити всі знаки пунктуації з тексту, можна використати string.punctuation або регулярні вирази re.sub(). Функція re.sub() може приймати три обов’язкові аргументи: pattern – регулярний вираз для

пошуку, replacement – текст, яким замінюємо знайдені збіги, string – рядок, у якому здійснюється пошук і заміна. Стала string.punction зберігає рядок символів «!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~» також може статися в нагоді при очищенні тексту від пунктуації, проте потрібно звертати увагу, що за допомогою сталої також можна прибрати крапки, що не завжди є метою.

4. Як розбити текст на речення?

У python розбити текст на речення можна за допомогою методу .split(). split() — це метод у Python, який розділяє рядок на частини за заданим роздільником і повертає список.

5. Що робить метод .split('.')?

Метод .split('.') перевіряє кожен символ рядка від початку, щойно знаходить першу крапку, відразу розбиває рядок і продовжує читати далі. Потім шукає наступну крапку і знову розбиває, поки не пройде весь рядок. Повертає список, у якому зберігаються розділені по крапці частини рядка. Якщо текст закінчується крапкою, останній елемент у списку буде порожнім рядком ('').

6. Які проблеми можуть виникнути при розбитті тексту на речення?

Проблемою може бути те, що крапку («.») можливо використовувати не тільки як закінчення речення, а і як скорочення слів та абревіатуру. Також, речення можуть закінчувати й іншими знаками, як наприклад, знаком оклику «!» або знаком питання «?». Крім того, метод .split('.') може утворювати пусті речення через випадки використання крапки як трикрапки «…» або інших. Проблемою можуть стати і HTML-теги та спеціальні символи, які переносяться з веб-сторінки, і їх бажано прибирати на початку роботи з текстом, щоб не виникло проблем у майбутньому.

7. Як Python обробляє кодування тексту?

Python обробляє текст за допомогою Юнікоду, де всі рядки є послідовностями символів Юнікоду. Кодування символів в байти (наприклад, UTF-8) та навпаки, обробляється за допомогою вбудованих функцій, таких як .encode() та .decode()

8. Як перевірити результат розбиття?

Щоб перевірити результат розбиття тексту на речення, можна вивести список sentences після виконання .split('.'). Це допоможе побачити, чи правильно розділився текст і чи є порожні елементи.

9. Що таке регулярні вирази (re) у Python?

Регулярні вирази в Python (скор. “RegEx” від англ. “Regular Expression”) — це послідовність символів, які визначають шаблон для пошуку відповідностей. Для написання регулярних виразів використовуються метасимволи. Метасимволи — це символи, що інтерпретуються особливим чином у механізмі регулярного виразу. Список метасимволів у Python: [] . ^ $ * + ? {} () \ |.

10. Який модуль найчастіше використовують для роботи з текстом?

Найчастіше для роботи з текстом в Python використовують вбудований тип даних str та його методи, а також модуль re для роботи з регулярними виразами.

В Python текст представляється у вигляді об'єктів str. Ці об'єкти мають багато вбудованих методів для роботи з текстом, таких як lower(), upper(), replace(), split(), join(), strip() та інші. Ці методи дозволяють змінювати текст, конвертувати його в інші форми, а також виконувати різні операції з текстом.

В Python текст представляється у вигляді об'єктів str. Ці об'єкти мають багато вбудованих методів для роботи з текстом, таких як lower(), upper(), replace(), split(), join(), strip() та інші. Ці методи дозволяють змінювати текст, конвертувати його в інші форми, а також виконувати різні операції з текстом, як наприклад, string.punctuation, string.whitespace.