Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Скриптові мови програмування (Python)

Лабораторна робота №6
Тема: «ЗБІР ДАНИХ З ВЕБ-ДОКУМЕНТІВ ЗА ДОПОМОГОЮ МОВИ PYTHON»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: ассистент
Ткаченко О.С.

Кропивницький 2025

Мета роботи - навчитися одержувати дані з html-сторінок та здійснювати їх аналіз, використовуючи можливості мови Python.

ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Завдання:

Реалізуйте програму, яка для довільної сторінки будь-якого сайту новин буде підраховувати частоту появи слів у тексті новини, частоту появи htmlтегів, кількість посилань та зображень.

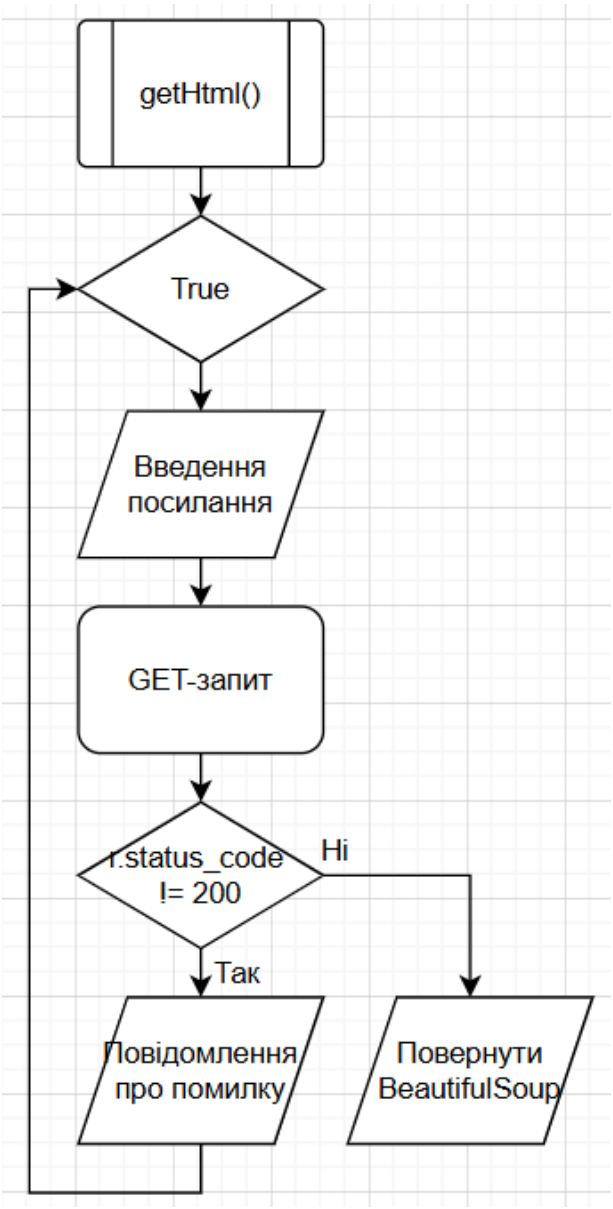
Принцип роботи функції getHtml()

1. Користувач вводить посилання.
2. Функція надсилає HTTP-запит.
3. Якщо статус-код не 200 — показує помилку й просить посилання знову.
4. Якщо все добре — повертає HTML-сторінку у вигляді об'єкта BeautifulSoup.

Проєктні рішення

- **Цикл while True**: дозволяє вводити посилання доти, поки не буде успішної відповіді.
- **Перевірка статус-коду**: захищає від неправильних або недоступних сторінок.
- **Використання BeautifulSoup**: одразу готує HTML до аналізу.

Блок схема:



Лістинг функції:

```
def getHtml():
        while True:
        link = input("Enter the link: ")
        r = requests.get(link)
        if r.status_code != 200:
            print("Error: Unable to access the link.")
            continue
        return BeautifulSoup(r.text, 'html.parser')
```

Принцип роботи функції freqWordCounter(page)

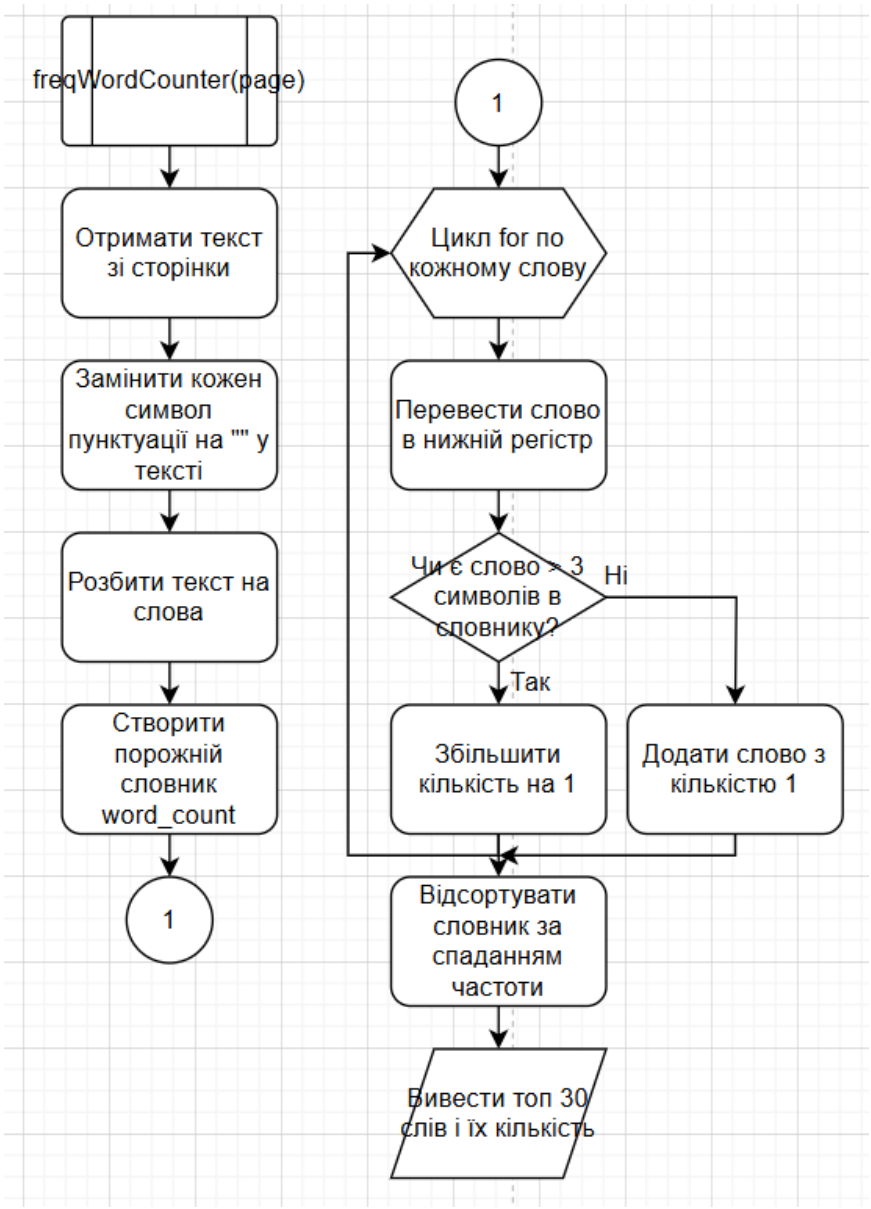
1. Отримує текст зі сторінки через page.get_text().

2. Видаляє всі розділові знаки.
3. Розбиває текст на слова.
4. Ігнорує слова з довжиною ≤ 3.
5. Рахує кількість повторень кожного слова.
6. Виводить 30 найчастіших слів.

Проєктні рішення

- **Використання get_text()**: виділяє лише текст без HTML-тегів.
- **Очищення символів вручну**: спрощує слова для точнішого підрахунку.
- **Ігнорування коротких слів (≤3)**: фільтрує малозначущі слова (як "і", "на", "це").
- **Сортування за частотою**: дозволяє легко знайти найпоширеніші слова.
- **Вивід у форматі списку**: зручно для читання результату.

Блок схема:



Лістинг функції:

```
def freqWordCounter(page):
    text = page.get_text()
    for i in ".,!?:;()[]{}<>-_\"'+`~@#$%^&*|\\/": text = text.replace(i, "")
    words = text.split()
    word_count = {}
    for word in words:
        word = word.lower()
        if len(word) <= 3: continue
        if word not in word_count: word_count[word] = 1
        else: word_count[word] += 1

    sorted_word_count = sorted(word_count.items(), key=lambda x: x[1],
reverse=True)
```

```
print("Most frequent words:")
for idx, (word, count) in enumerate(sorted_word_count[:30], 1):
    print(f"\t{idx}. {word}: {count}")
```

Результат виконання:

```
Most frequent words:
        1. правда: 9
        2. новини: 7
        3. травня: 7
        4. україна: 6
        5. мита: 4
        6. економічна: 4
        7. матеріали: 4
        8. зброю: 3
        9. проти: 3
        10. українська: 3
        11. спецпроєкти: 3
        12. реклама: 3
        13. редакція: 3
        14. отримала: 2
        15. розділи: 2
        16. публікації: 2
        17. погляди: 2
        18. компаній: 2
        19. фінанси: 2
        20. інтервю: 2
        21. weekly: 2
        22. charts: 2
        23. земельний: 2
        24. захист: 2
        25. країни: 2
        26. україни: 2
        27. каральні: 2
        28. посилання: 2
        29. сьогодні: 2
        30. зброї: 2
```

Принцип роботи freqTagCounter(page)

1. Збирає всі HTML-теги зі сторінки за допомогою find_all().
2. Підраховує, скільки разів зустрічається кожен тег.
3. Сортує теги за частотою в порядку спадання.
4. Виводить список тегів з їх кількістю.

Проєктні рішення

- **find_all() без аргументів**: знаходить **усі теги**, що забезпечує повну картину структури сторінки.
- **Словник tag_count**: ефективний для підрахунку повторень.
- **Сортування за частотою**: дає змогу бачити найпоширеніші елементи HTML.
- **Форматований вивід**: зручний для перегляду та аналізу.

Блок схема:

Лістинг функції:

```
def freqTagCounter(page):
    tags = page.find_all()
    tag_count = {}
    for tag in tags:
        tag_name = tag.name
        if tag_name not in tag_count: tag_count[tag_name] = 1
        else: tag_count[tag_name] += 1

    sorted_tag_count = sorted(tag_count.items(), key=lambda x: x[1],
reverse=True)
    print("Most frequent tags:")
    for idx, (tag, count) in enumerate(sorted_tag_count, 1):
        print(f"\t{idx}. {tag}: {count}")
```

Результат виконання:

```
Most frequent tags:
        1. a: 80
        2. div: 69
        3. li: 48
        4. link: 32
        5. span: 26
        6. meta: 24
        7. script: 21
        8. path: 19
        9. svg: 17
        10. p: 13
        11. strong: 10
        12. g: 5
        13. nav: 5
        14. ul: 5
        15. title: 3
        16. style: 3
        17. rect: 3
        18. aside: 3
        19. h3: 3
        20. header: 2
        21. input: 2
        22. source: 2
        23. h2: 2
        24. ins: 2
        25. html: 1
        26. head: 1
        27. body: 1
        28. noscript: 1
        29. iframe: 1
        30. form: 1
        29. iframe: 1
        30. form: 1
        31. article: 1
        32. h1: 1
        33. picture: 1
        34. img: 1
        35. h4: 1
        36. footer: 1
        37. b: 1
```

Принцип роботи Amount(page, tag)

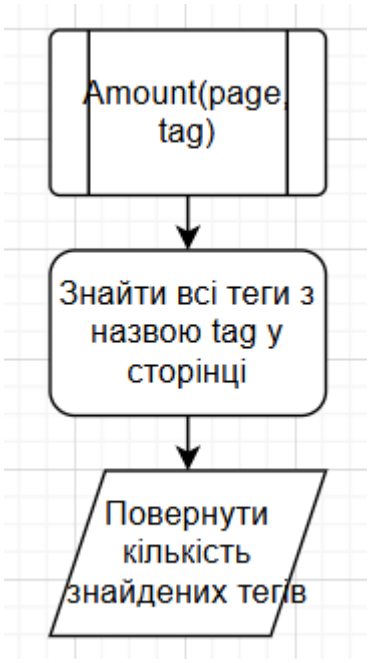
1. Приймає HTML-сторінку (page) та назву тега (tag).
2. Знаходить усі елементи з таким тегом.
3. Повертає кількість знайдених тегів.

Проєктні рішення

- **find_all(tag)**: шукає лише потрібний тег, що робить функцію точною й швидкою.
- **len(tags)**: рахує кількість елементів без додаткових циклів.

- **Проста структура**: легко використовувати в інших частинах програми чи тестах.

Блок схема:



Лістинг функції:

```
def Amount(page, tag):
    tags = page.find_all(tag)
    return len(tags)
```

Результат виконання:

```
Number of links: 80
Number of images: 1
```

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як виконати GET-запит до веб-сайту засобами мови Python?

За допомогою модуля requests:

r = requests.get(link)

2. Який модуль/модулі можна використати для збору даних з вебсторінки?

Найчастіше — BeautifulSoup з модуля bs4.

Він дозволяє аналізувати HTML та знаходити теги.

3. Яку структуру має стандартна HTML-сторінка?

Стандартна структура:

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>...</body>
</html>
```

4. Які засоби мова Python надає для роботи з реляційними СУБД?

Python має кілька інструментів для роботи з базами даних:

- **sqlite3** — вбудований модуль для роботи з локальними SQLite-базами без потреби в окремому сервері.
- **SQLAlchemy** — потужна бібліотека для роботи з різними СУБД (MySQL, PostgreSQL, SQLite тощо) з підтримкою ORM (об'єктно-реляційного відображення) або «чистого» SQL.

Вони дозволяють створювати таблиці, додавати, змінювати та читати дані з БД.

5. Що таке парсер? Для чого він потрібен?

Парсер — це програма, що аналізує структуру тексту (наприклад, HTML).

У getHtml() використовується BeautifulSoup(r.text, 'html.parser'), щоб перетворити HTML у зручну для обробки форму.