Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

«СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

Лабораторная работа № 3

«Разработка приложения с графическим интерфейсом. Модуль Pillow. Веб-парсинг»

Проверила:
Василькова А.Н.

Выполнил:
Островский Я.А.

Минск 2024

Вариант № 2

Цель: освоить разработку приложения с графическим интерфейсом. Освоить веб-парсинг с применением языка программирования Python.

Задание 1: Разработка программ с использованием классов. В разработке применить графический интерфейс. Создать класс, который проверяет число и определяет, что это – простое число, число Фибоначчи, комплексное число, целое, вещественное.

Задание 2: Получить на вход программы URL страницы интернет-ресурса, а так же параметр depth - уровень глубины исследования ссылок, присутствующих на заданной странице. С помощью пакетов scrapy (уже присутствует в Anaconda), либо BeatifulSoup, lxml плюс модуля для оформления запросов - Requests осуществить просмотр/парсинг заданной веб-страницы и собрать статистику следующего вида:

1. Количество ссылок на странице анализируемого уровня
2. Количество слов на странице анализируемого уровня
3. Построить гистограмму частоты встречаемости символов (заданного слова) на странице анализируемого уровня (русск., англ., спец. символы).
4. Построить гистограмму встречаемости длин слов (т.е. сколько слов состоит из 1,2,3,4… и т.п. k- символов). K ограничить константой, но не менее 7
5. Объединить статистику разных страниц в соответствующие таблицы (ш. 1-4), отображающие единую статистику по всем проанализированным страницам.

Дополнительные варианты получаемой статистики:

- количество изображений на странице
- общий и средний размер изображений на странице
- количество заголовков и подзаголовков (- в зависимости от тегов заголовков разных уровней)
- гистограмма встречаемости длин слов в подзаголовках

Файл check_number.py

```
import PySimpleGUI as sg


def true_checker(tf):
    if tf:
        return "з'яўляецца"
    else:
        return "не з'яўляецца"
```

```
class NumberClassifier:
    def __init__(self, number):
        self.number = number

    def is_prime(self):
        if self.number < 2:
            return False
        for i in range(2, int(self.number ** 0.5) + 1):
            if self.number % i == 0:
                return False
        return True

    def is_fibonacci(self):
        a, b = 0, 1
        while a < self.number:
            a, b = b, a + b
        return a == self.number

    def is_complex(self):
        return isinstance(self.number, complex)

    def is_integer(self):
        return isinstance(self.number, int)

    def is_real(self):
        return isinstance(self.number, float) or isinstance(self.number, int)


layout = [
    [sg.Text("Увядзіце лік:"), sg.InputText(key="number")],
    [sg.Button("Выканаць аналіз")]
]

window = sg.Window("Аналіз лікаў", layout)

while True:
    event, values = window.read()

    if event == sg.WIN_CLOSED:
        break

    try:
```

```
        number = int(values["number"])
        num = NumberClassifier(number)
        prime_result = true_checker(num.is_prime())
        fibonacci_result = true_checker(num.is_fibonacci())
        complex_result = true_checker(num.is_complex())
        integer_result = true_checker(num.is_integer())
        real_result = true_checker(num.is_real())

        sg.popup(f"Лік {num.number} {prime_result} простым.",
                 f"Лік {num.number} {fibonacci_result} лікам Фібаначы.",
                 f"Лік {num.number} {complex_result} камплексным.",
                 f"Лік {num.number} {integer_result} цэлым.",
                 f"Лік {num.number} {real_result} рэчаісным.")

    except ValueError:
        sg.popup("Некарэктны ўвод. Паспрабуйце яшчэ раз.")

window.close()
```

Файл main.py

```
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore", category=UserWarning)


# Вяртае змест вэб-старонкі па зададзеным URL.
def get_page_content(url):
    response = requests.get(url)
    return response.text


# Падлічвае колькасць спасылак і словаў на вэб-старонцы.
def count_links_and_words(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    links = len(soup.find_all('a'))
    words = len(soup.get_text().split())
    return links, words
```

```
# Будуе гістаграму частаты сустракаемасці сімвалаў ў зададзеным слове.
def plot_character_frequency(text):
    char_freq = {char: text.count(char) for char in set(text)}
    plt.bar(char_freq.keys(), char_freq.values())
    plt.title(f"Частата сустракаемасці сімвалаў на сайце")
    plt.show()


# Будуе гістаграму частаты сустракаемасці даўжыняў словаў у зададзеным
тэксце.
def plot_word_length_frequency(text):
    word_lengths = [len(word) for word in text.split() if len(word) >= 7]
    plt.hist(word_lengths, bins=range(7, max(word_lengths) + 2), align='left',
rwidth=0.8)
    plt.title("Частата сустракаемасці даўжынь словаў")
    plt.xlabel("Даўжыня словаў")
    plt.ylabel("Частата")
    plt.show()


# Падлічвае колькасць малюнкаў на вэб-старонцы.
def count_images(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    images = len(soup.find_all('img'))
    return images


# Падлічвае агульны і сярэдні памер малюнкаў на вэб-старонцы.
def get_image_sizes(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    images = soup.find_all('img')
    total_size = 0
    for image in images:
        total_size += int(image['width']) * int(image['height'])
    average_size = total_size / len(images) if len(images) > 0 else 0
    return total_size, average_size


# Падлічвае колькасць загалоўкаў розных узроўняў на вэб-старонцы.
def count_headings(html_content):
```

```
    soup = BeautifulSoup(html_content, 'html.parser')
    headings = {}
    for level in range(1, 7):  # Заголовки обычно от h1 до h6
        count = len(soup.find_all(f'h{level}'))
        headings[f'h{level}'] = count
    return headings


# Будуе гістаграму частаты сустракаемасці даўжынь слоў у загалоўках на вэб-старонцы.
def plot_heading_word_length_frequency(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    headings_text = ' '.join([heading.get_text() for heading in soup.find_all(['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])])
    word_lengths = [len(word) for word in headings_text.split()]
    plt.hist(word_lengths, bins=range(1, max(word_lengths) + 2), align='left', rwidth=0.8)
    plt.title("Частата сустракання даўжынь словаў у загалоўках")
    plt.xlabel("Даўжыня словаў")
    plt.ylabel("Частата")
    plt.show()


url = input("Введите URL страницы: ")

html_content = get_page_content(url)
links, words = count_links_and_words(html_content)
images = count_images(html_content)
total_size, average_size = get_image_sizes(html_content)
headings_count = count_headings(html_content)

print(f"Колькасць спасылак на старонцы: {links}")
print(f"Колькасць словаў на старонцы: {words}")
print(f"Колькасць малюнкаў на старонцы: {images}")
print(f"Агульны памер малюнкаў на старонцы: {total_size}")
print(f"Сярэдні памер малюнкаў на старонцы: {average_size}")
for heading, count in headings_count.items():
    print(f"Колькасць загалоўкаў {heading}: {count}")

text = BeautifulSoup(html_content, 'html.parser').get_text()
plot_character_frequency(text)
```

plot_word_length_frequency(text)
plot_heading_word_length_frequency(html_content)

Процесс работы программы check_number показан на рисунке 1.

Рисунок 1 – Результат выполнения программы micola

Результат выполнения программы check_number показан на рисунке 2.

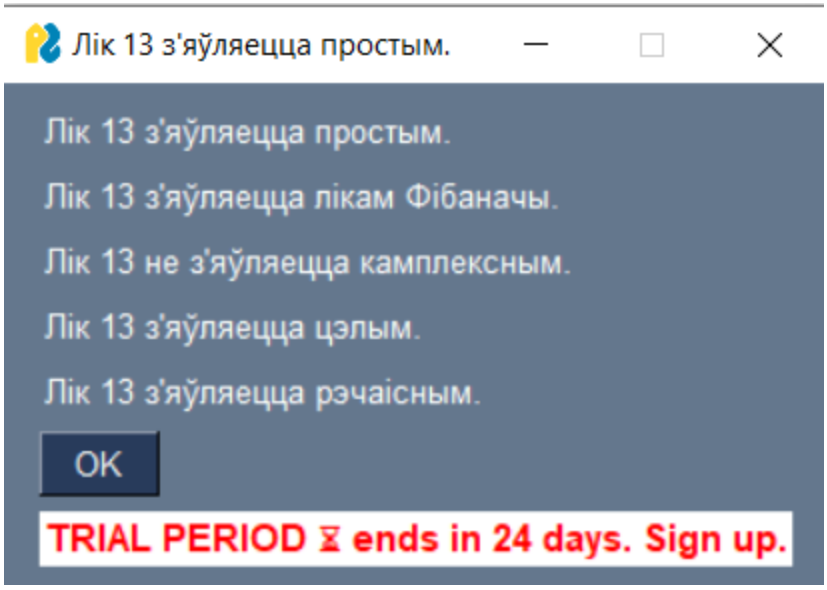

Рисунок 2 – Результат выполнения программы pnds

Результат выполнения программы main представлен на рисунках 3, 4, 5, 6.

```
Увядзіце URL старонкі: https://kinogo.pro/
Колькасць спасылак на старонцы: 5
Колькасць словаў на старонцы: 167
Колькасць малюнкаў на старонцы: 0
Агульны памер малюнкаў на старонцы: 0
Сярэдні памер малюнкаў на старонцы: 0
Колькасць загалоўкаў h1: 1
Колькасць загалоўкаў h2: 2
Колькасць загалоўкаў h3: 5
Колькасць загалоўкаў h4: 0
Колькасць загалоўкаў h5: 0
Колькасць загалоўкаў h6: 0
```

Рисунок 3 – Результат выполнения программы main

Рисунок 4 – Результат выполнения программы main

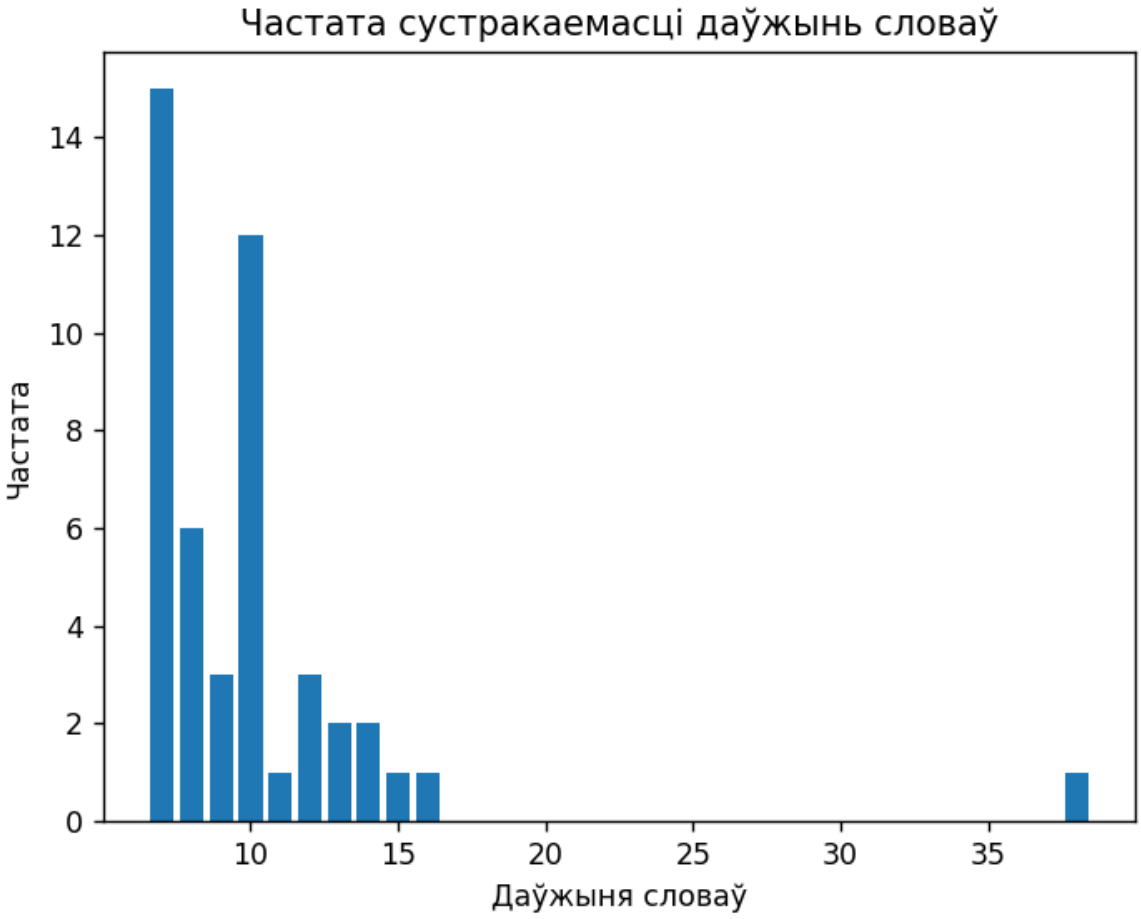

Рисунок 5 – Результат выполнения программы main

Рисунок 6 – Результат выполнения программы main