Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

### «СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

***Лабораторная работа № 3***

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

Проверила: Выполнил:

Василькова А.Н. 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. C 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.

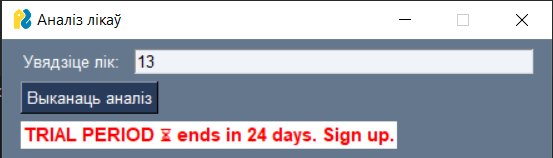


Рисунок 1 – Результат выполнения программы micola

Результат выполнения программы check\_number показан на рисунке 2.

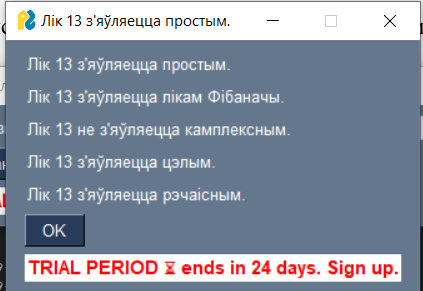


Рисунок 2 – Результат выполнения программы pnds

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

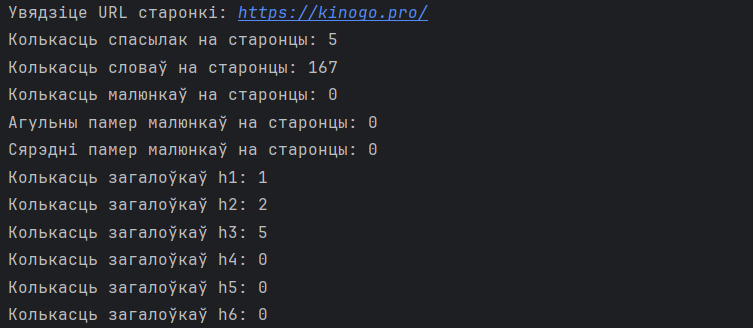


Рисунок 3 – Результат выполнения программы main

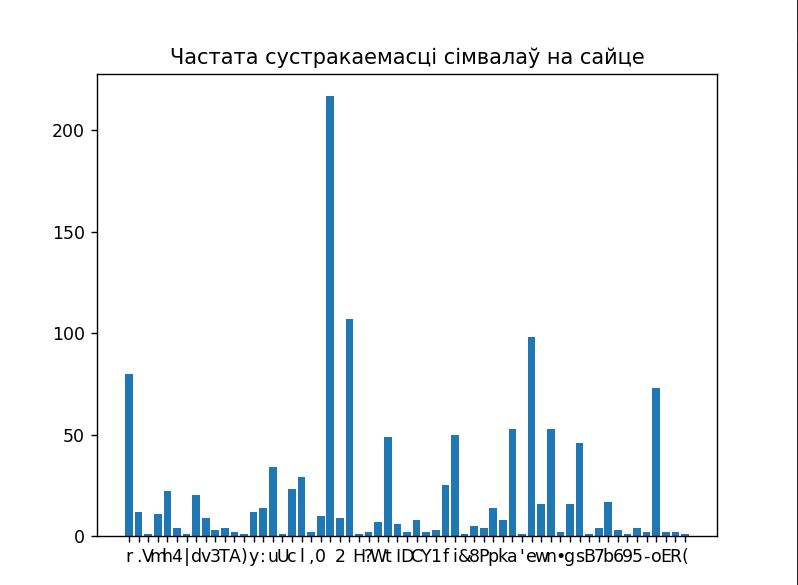


Рисунок 4 – Результат выполнения программы main

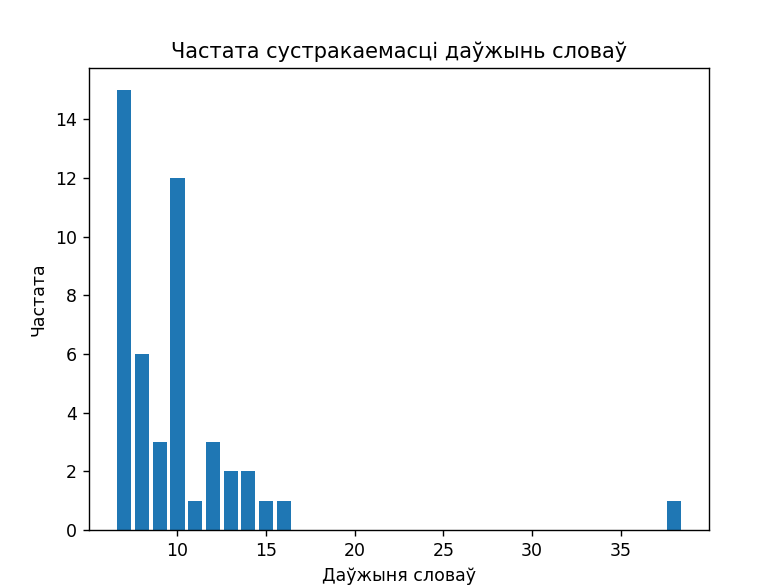


Рисунок 5 – Результат выполнения программы main

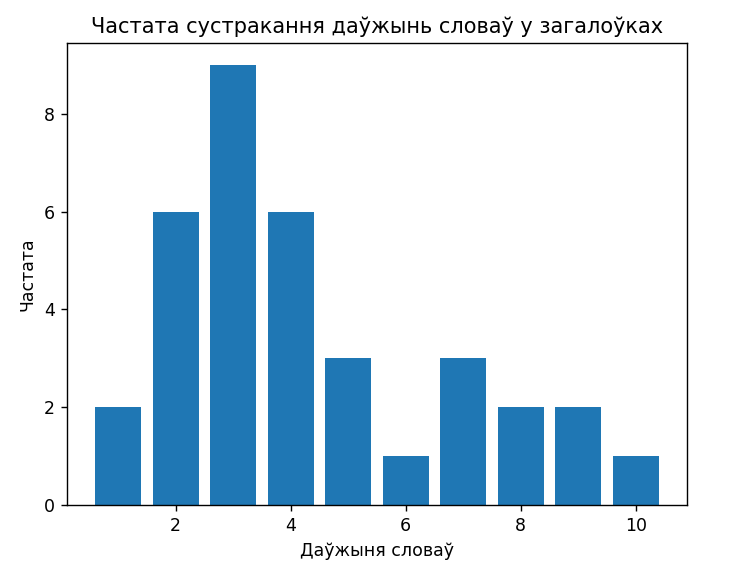


Рисунок 6 – Результат выполнения программы main