Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектронике»

Кафедра инженерной психологии и эргономики

Отчет по

ЛАБОРАТОРНОЙ РАБОТЕ №3

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

Подготовил:

Студент гр. 010101

Листванович А.А.

Проверила:

Василькова А.Н.

Минск 2023

Задача №1 – Вариант 5

Условие:

Разработка программ с использованием классов. В разработке применить графический интерфейс. Создать класс для вычисления корней квадратного уравнения. Предусмотреть все возможные варианты.

Решение:

import PySimpleGUI as sg

import math

class QuadraticEquationSolver:

def \_\_init\_\_(self):

self.layout = [

[sg.Text('Quadratic Equation Solver')],

[sg.Text('Enter the values of a, b, and c:')],

[sg.Text('a:'), sg.InputText()],

[sg.Text('b:'), sg.InputText()],

[sg.Text('c:'), sg.InputText()],

[sg.Button('Solve'), sg.Button('Clear'), sg.Button('Exit')],

[sg.Text('', size=(40, 1), key='output')]

]

self.window = sg.Window('Quadratic Equation Solver', self.layout)

def solve(self, a, b, c):

try:

a = float(a)

b = float(b)

c = float(c)

discriminant = b\*\*2 - 4\*a\*c

if discriminant < 0:

return 'No real roots'

elif discriminant == 0:

root = -b / (2\*a)

return f'One real root: {root}'

else:

root1 = (-b + math.sqrt(discriminant)) / (2\*a)

root2 = (-b - math.sqrt(discriminant)) / (2\*a)

return f'Two real roots: {root1}, {root2}'

except ValueError:

return 'Invalid input'

def run(self):

while True:

event, values = self.window.read()

if event == sg.WINDOW\_CLOSED or event == 'Exit':

break

elif event == 'Clear':

self.window['output'].update('')

elif event == 'Solve':

a, b, c = values[0], values[1], values[2]

output = self.solve(a, b, c)

self.window['output'].update(output)

self.window.close()

def pillow\_practice() -> None:

solver = QuadraticEquationSolver()

solver.run()

Результат работы программы отображен на рисунке 1.

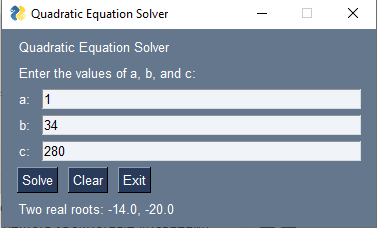


Рисунок 1 – Результат первой задачи

Задача №2

Условие:

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.

Решение:

import requests

from bs4 import BeautifulSoup

import pandas as pd

import matplotlib.pyplot as plt

def bs4\_practice() -> None:

# Define a list of URLs to scrape

urls = ['https://www.w3schools.com/css/css3\_borders.asp',

'https://www.w3schools.com/python/default.asp',

'https://www.w3schools.com/cpp/default.asp',

'https://www.w3schools.com/sql/default.asp',

'https://www.w3schools.com/js/default.asp'

]

# Initialize an empty DataFrame to store the results

data = pd.DataFrame(columns=['url', 'link\_count', 'word\_count'])

# Loop through each URL

for url in urls:

# Send a GET request to the URL

response = requests.get(url)

# Parse the HTML content using BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

# Get the number of links on the page

link\_count = len(soup.find\_all('a'))

# Get the number of words on the page

words = soup.get\_text().split()

word\_count = len(words)

# Append the results to the DataFrame

new\_data = pd.DataFrame({

'url': [url],

'link\_count': [link\_count],

'word\_count': [word\_count]

})

data = pd.concat([data, new\_data], ignore\_index=True)

# Print the results

print(data)

make\_histograms(urls[0])

def make\_histograms(url: str) -> None:

import matplotlib

matplotlib.use('TkAgg')

response = requests.get(url)

soup = BeautifulSoup(response.content, 'html.parser')

text = soup.get\_text()

words = text.split()

# count frequency of each character

char\_counts = {}

for char in text:

if char in char\_counts:

char\_counts[char] += 1

else:

char\_counts[char] = 1

# plot histogram of character frequency

plt.bar(char\_counts.keys(), char\_counts.values())

plt.show()

# count frequency of word lengths

word\_lengths = [len(word) for word in words]

max\_length = 7

length\_counts = {}

for i in range(max\_length, max(word\_lengths) + 1):

length\_counts[i] = word\_lengths.count(i)

# plot histogram of word length frequency

plt.bar(length\_counts.keys(), length\_counts.values())

plt.show()

Результат работы программы отображен на рисунках 2, 3, 4.

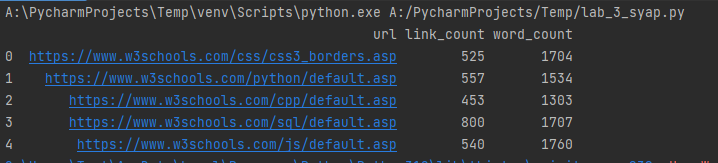


Рисунок 2 – Dataframe второй задачи

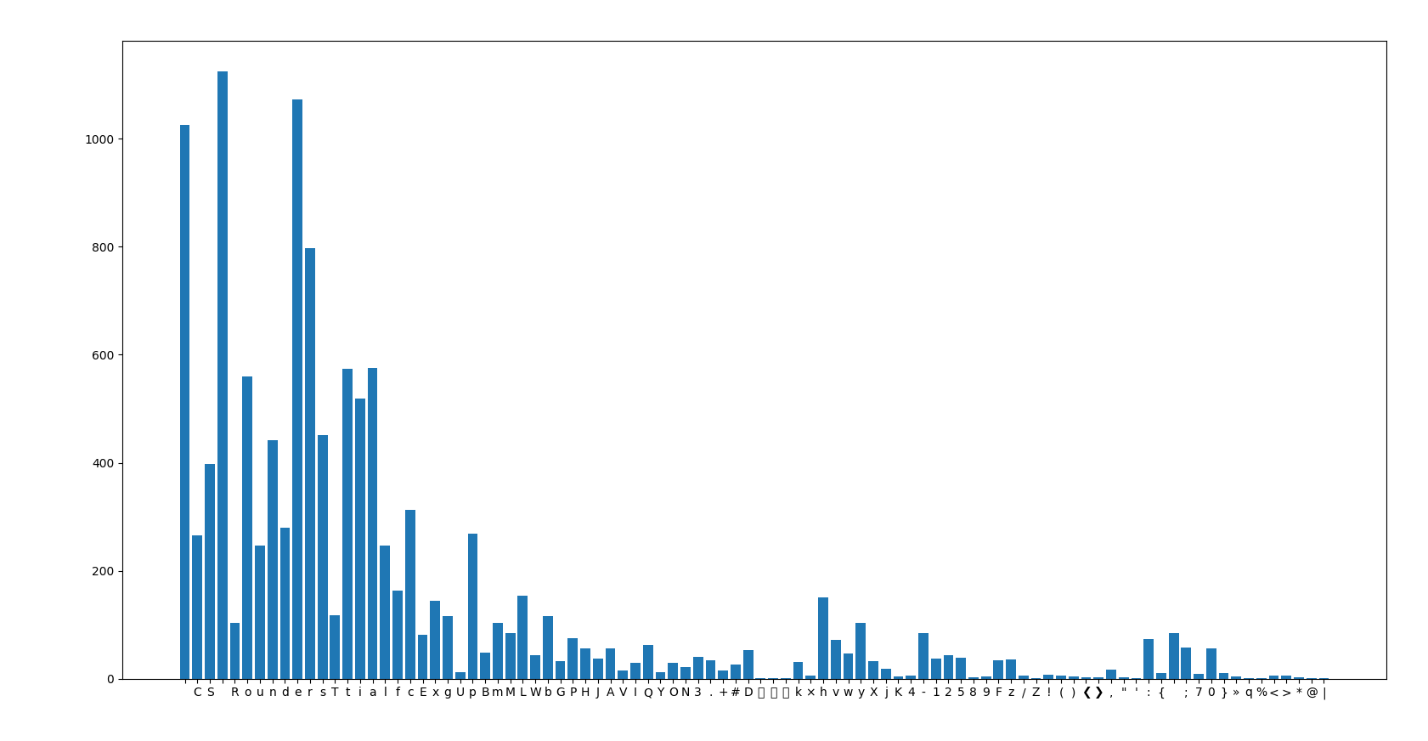


Рисунок 3 – Гистограмма символов второй задачи

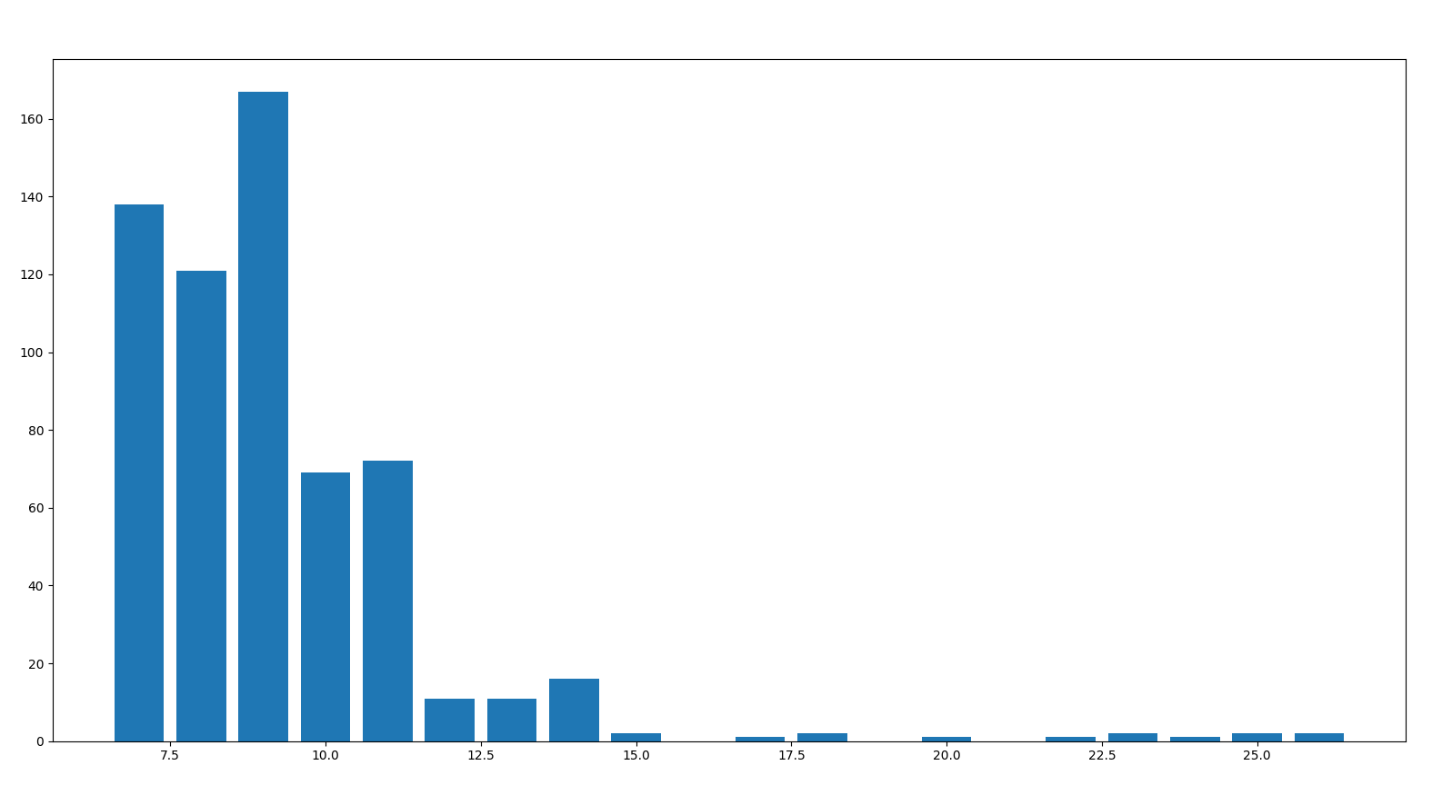


Рисунок 4 – Гистограмма длин слов второй задачи