МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Основные управляющие конструкции. Wikipedia API

Студент гр. 0383 _____ Шквиря Е.В.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Изучить устройство основных управляющих конструкций языка Python и научиться работать с Wikipedia API.

Задание.

Используя вышеописанные инструменты, напишите программу, которая принимает на вход строку вида

```
название_страницы_1, название страницы_2, ... название_страницы_n, сокращенная_форма_языка
```

и делает следующее:

1. Проверяет, есть ли такой язык в возможных языках сервиса, если нет, выводит строку "no results" и завершает выполнение программы. В случае, если язык есть, устанавливает его как язык запросов в текущей программе.

2. Ищет максимальное число слов в кратком содержании страниц "название_страницы_1", "название страницы_2", ... "название_страницы_n", выводит на экран это максимальное количество и название страницы (т.е. её **title**), у которой оно обнаружилось. Считается, что слова разделены пробельными символами.

Если максимальных значений несколько, выведите последнее.

3. Строит список-цепочку из страниц и выводит полученный список на экран.

Элементы списка-цепочки - это страницы "название_страницы_1", "название страницы_2", ... "название_страницы_n", между которыми может быть одна промежуточная страница или не быть промежуточных страниц.

Гарантируется, что существует или одна промежуточная страница или ноль: т.е. в числе ссылок первой страницы можно обнаружить вторую.

Цепочка должна быть кратчайшей, т.е. если существуют две цепочки, одна из которых содержит промежуточную страницу, а вторая нет, стройте цепочку без промежуточного элемента.

Выполнение работы.

Порядок выполнения поставленной задачи программой:

1. Считывание строки с названиями страниц и его последующее разделение на элементы списка.

2. Проверка существования языка путём его поиска в списке всевозможных. В случае его отсутствия - вывод «no results» и завершение программы, иначе данный язык устанавливается как «язык запросов».

3. Подсчёт максимального количества слов в кратком содержании каждой страницы осуществляется путём получения каждого содержания. После получения строки в ней заменяются символы \n и \t на пробелы, чтобы корректно обрабатывать тексты из нескольких абзацев и с символами табуляции. В дальнейшем из строки делается список и подсчитывается количество элементов в нём. Так происходит подсчёт слов и находится искомая страница.

4. Цепочка ищется путём поиска следующей страницы в списке ссылок прошлой. Такой подход гарантирует нам, что мы не будем использовать лишние промежуточные звенья, значит наша цепочка будет кратчайшей. В случае отсутствия следующей страницы в списке ссылок, поиск будет производиться в списке ссылок каждой страницы из списка ссылкок прошлой страницы.

Используемые переменные:

array — список страниц, поступающих из потока ввода.

lang — язык, поступающий от пользователя для проверки и дальнейшей установки.

string — строка, у которой подсчитывается количество слов.

page_array — список страниц, введённых пользователем, который подаётся как аргумент функции.

maxCnt — переменная, которая хранит в себе максимальное количество слов в кратком содержании страницы.

ansPage — название страницы, в которой было обнаружено максимальное количество слов.

page — название страницы из page_array.

cntInStr — количество слов в строке.

i — индекс элемента, для которого ищется следующий элемент цепочки.

linksArray — список ссылок прошлой страницы.

answerArray — вычисленная цепочка.

Используемые функции:

is_page_valid — проверка существования страницы.

is_right_lang — проверка существования языка.

cnt_word — количество слов в строке.

max_cnt_words_summary — поиск страницы с максимальным количеством слов и его вычисление.

find_support_chain — поиск промежуточного звена.

find_chain — поиск цепочки.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Айсберг, IBM, ru	115 IBM ['Айсберг', 'Буран', 'IBM']	Верный ответ
2.	Лукерьино (Омская область), Васютино (Омская область), Культжугут, Озёрка (Омская область), ru	14 Озёрка (Омская область) ['Лукерьино (Омская область)', 'Васютино (Омская область)', 'Культжугут', 'Озёрка (Омская область)']	Верный ответ
3.	Candy, Атеизм, ru	228 Атеизм ['Candy', '2010 год', 'Атеизм']	Верный ответ
4.	2012 BX34, IGR J17329-2731, en	139 2012 BX34 ['2012 BX34', '2012 TC4', 'IGR J17329-2731']	Верный ответ
5.	2012 BX34, IGR J17329-2731, en-ru	no results	Верный ответ

Выводы.

Были изучены основные управляющие конструкции языка Python и способы взаимодействия с Wikipedia API.

Разработана программа, считывающая от пользователя список названий страниц и язык для запросов, а также позволяющая найти максимальное

количество слов в странице из списка и построить с помощью названий страниц кратчайшую цепочку с добавлением промежуточных звеньев. Для реализации использовались циклы, условные операторы if-else, функции и подключение модуля wikipedia.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1.py

```
import wikipedia



# существует ли такая страница
def is_page_valid(page):
    try:
        wikipedia.page(page)
    except Exception:
        return False
    return True



# проверка существования языка
def is_right_lang(lang):
    return lang in wikipedia.languages()



# количество слов в строке
def cnt_word(string):
    return len(string.replace('\n', ' ').replace('\t', ' ').split(' '))
```

```
# максимальное количество слов в строке
def max_cnt_words_summary(page_array):
    maxCnt = 0
    ansPage = ""
    for page in page_array:
        cntInStr = cnt_word(wikipedia.page(page).summary)
        if cntInStr >= maxCnt:
            maxCnt = cntInStr
            ansPage = page
    return [maxCnt, wikipedia.page(ansPage).title]



# поиск промежуточного звена
def find_support_chain(page_array, i):
    linksArray = wikipedia.page(page_array[i]).links
    for item in linksArray:
        if not is_page_valid(item):
            continue
        if page_array[i + 1] in wikipedia.page(item).links:
            return item



# поиск цепочки
def find_chain(page_array):
    answerArray = [page_array[0]]
    for i in range(0, len(page_array) - 1):
        if page_array[i + 1] not in wikipedia.page(page_array[i]).links:
            answerArray.append(find_support_chain(page_array, i))
        answerArray.append(page_array[i + 1])
    return answerArray



# точка начала программы
array = input().split(', ')
if is_right_lang(array[-1]):
    wikipedia.set_lang(array.pop())
else:
```

```
    print("no results")
    quit(0)
print(*max_cnt_words_summary(array))
print(find_chain(array))
```