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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Введение в информационные технологии»

Тема: Основные управляющие конструкции

Студент гр. 9383 _____ Корсунов А.А.

Преподаватель _____ Розмочева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучение основных управляющих конструкций языка Python

Задание.

Используя библиотеку wikipedia, 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.у. 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.

Выполнение работы.

Программа импортирует предложенную библиотеку Wikipedia.

Созданы 3-и функции для выполнения поставленных задач:

1) st_language(language_r):

Функция принимает на вход значение выбранного языка (последний элемент созданного мной из вводимой строки списка) и проверяет его на вхождение в языки Wiki с помощью функции languages(), в зависимости от результата, функция возвращает либо False либо True ;

2) search_max_words_and_its_name(long, a):

Функция принимает на вход длину списка (кроме последнего элемента) и сам список. Далее идет самый типичный процесс нахождение минимального элемента и его индекса. В самой функции используется функция page() для нахождения страницы на Wiki и поля .title (поиск индекса (названия) страницы) и .summary (для счета всех слов с помощью функции len());

3) ze_pochka(long, a):

Функция принимает точно такие же значение как и вторая функция. Далее идет процесс перебора значения с помощью поля .links: если k+1 эелемент не входит в ссылки k элемента, то идет проверка, существует ли промежуточный элемент между ними, если существует, то его добавляют в цепочку. В конце также идет добавление предпоследнего элемента списка, т.к. вышеописанная конструкция работает до long-1 элемента. В самом конце происходит возвращение цепочки.

В основной части кода вызывается первая функция, и если она возвращает не False, то происходит вывод вызовов функций 2 и 3.

Тестирование.

№ п/п	Входные данные	Выходные данные
1	Айсберг, IBM, ru	115 IBM ['Айсберг', 'Буран', 'IBM']
2	Чуумпу делай, Хоту Америка, Континент, sah	77 Чуумпу делай ['Чуумпу делай', 'Американалар', 'Хоту Америка', 'Континент']

Выводы.

Были изучены базовые конструкции Python , также написана программа, в которой использовались методы, функции, поля и модули.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
from wikipedia import *


def st_language(language_r):
    if language_r in languages():
        set_lang(language_r)
        return True
    else:
        return False


def search_for_maximumes(long, a):
    max_words = int(0)
    name_of_page = ''
    for m in range(long):
        words = len(page(a[m]).summary.split())
        if words >= max_words:
            max_words = words
            name_of_page = page(a[m]).title
max_words = str(max_words)
return max_words, name_of_page


def maximum_chain_search(long, a):
    drr = []
    for k in range(long-1):
        drr.append(a[k])
        links_of_ak = page(a[k]).links
```

```
        if a[k+1] in links_of_ak:
            continue
        else:
            for n in range(len(links_of_ak)):
                links_of_akn = page(links_of_ak[n]).links
                if a[k+1] in links_of_akn:
                    drr.append(links_of_ak[n])
                    break
                else:
                    continue
    drr.append(a[-2])
    return drr


arr = input().split(', ')
long_r = len(arr)
language = arr[-1]
if st_language(language) :
    print(' '.join(search_for_maximumes(long_r - 1, arr))
    print(maximum_chain_search(long_r - 1, arr))
else:
    print('no result')
```