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

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Изучение основных управляющих конструкций языка Python, модуля Wikipedia API.

Задание.

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

Основные теоретические положения.

Встроенные функции: *input() , print()* (Ввод и вывод соответственно), *len()* (Длина объекта), *range()* (Ряд чисел в диапазоне).

Операторы: *if():*, *else():* (Проверка истинности), *break* (Прерывание текущего цикла), *in , not in* (Проверка принадлежности/не принадлежности), *return* (Возвращение из функции).

Цикл: *For … in …:* (Перебор элементов)

Методы: *.split()* (Разбивает строку на части, используя разделитель, и возвращает эти части списком), *.append(…)* (Добавляет элемент <…> в конец

списка), *.keys()* (Возвращает ключ в словаре), *.pop(…)* (Удаляет элемент <…> из списка).

Функции модуля Wikipedia: *page(…)* (Поиск страницы <…>), *languages()* (Поиск всех возможных языков сервиса), *set_lang(lang)* (Установить язык lang, как язык запросов в текущей программе)

Атрибуты класса WikipediaPage*:* *page.summary* (Краткое содержание страницы page), *page.title* (Название страницы page), *page.links* (Список названий страниц, ссылки на которые содержит страница page).

Выполнение работы.

Для подключения модуля Wikipedia была использована инструкция *import wikipedia.*

Объявляются переменная *max*, строка *maxt* (которые необходимы для решения второй подзадачи). Далее производится ввод данных в список *Initial* посредствам встроенной функции *input()* и метода *.split(“, “).* Объявляется список *res* (необходимый для третьей подзадачи).

Затем проверяется наличие введённого языка (последний элемент ввода) в возможных языках сервиса (является ли язык ключом словаря *languages()* - функции модуля Wikipedia) посредствам метода *.keys()* . Если языка нет - вывод строки "no results" посредствам функции *print()* и окончание работы программы. Иначе переход к следующим подзадачам.

При наличии языка, он устанавливается как язык запросов в текущей программе, при помощи функции *set_lang()* модуля Wikipedia. Далее методом *.pop()* из введённого списка удаляется последний элемент, хранящий язык.

Выполнение второй подзадачи реализуется перебором элементов введённого списка при помощи цикла *for i in range(),* где диапазоном является размер списка *Initial* - *(len(Initial)).* Затем, на каждой итерации, в *abb* записывается краткое содержание страницы (страница - элемент *Initial*). Далее осуществляется поиск страницы с наибольшим количеством слов в кратком содержании (*len(abb*)). Реализация происходит посредствам сравнения числа слов в кратком содержании страницы, при первой итерации происходит

сравнение с “0” . Вместе с тем, при выполнении условия, в переменную *max* записывается максимальное число слов краткого содержания страницы (для следующего сравнения) и в *maxt* - название соответствующей страницы (при использовании атрибута класса WikipediaPage - *.title*). Использование “>=” позволяет сохранять последнее, если максимальных значений несколько. После окончания перебора *max* и *maxt* хранят ответ ко второй подзадаче.

Далее выполняется третья подзадача. В список *res* посредствам метода *.append()* добавляется нулевой (первый при обычном счёте) элемент введённых данных. Вновь реализуется перебор элементов *Initial*, но в данном случае диапазон не включает последний элемент. Следующие операции выполняются внутри цикла. В переменную *lin* записывается, при помощи атрибута класса WikipediaPage - *.links* , список названий страниц, ссылки на которые содержит элемент *Initial.* Далее, для нахождения самой короткой цепочки, происходит проверка наличия следующего элемента *Initial* в списке названия ссылок элемента текущей итерации. Если это условие выполняется, то в список *res* добавляется этот элемент. Иначе (для поиска промежуточного звена цепочки) - реализуется вложенный цикл *for j in range()*, диапазоном выступает длина списка *lin*. Далее ведётся перебор элементов *lin.* Проверяется существование страницы (функция *is_page_valid),* если факт существования подтверждён - реализуется алгоритм подобный предыдущему и затем в список *res* добавляется промежуточный элемент - элемент строки *lin* и уже за ним следующий элемент списка *Initial,* далее цикл прерывается посредствам *break*. Так, список *res* хранит самую короткую цепочку и является ответом к третьей подзадаче.

Вывод результата работы программы реализуется посредствам вложенной функции *print()*. Сначала выводятся значения *max* и *maxt* (ответ ко второй подзадаче), следом список *res* (ответ к третьей подзадаче). Данный вывод реализуется, если условие первой подзадачи выполнено.

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']	Ответ верный.
2.	Айсберг, Буран, ru	134 Буран (космический корабль) ['Айсберг', 'Буран']	Ответ верный.
3.	Айсберг, IBM, c	no results	Ответ верный.

Выводы.

Были изучены основные управляющие конструкции языка Python и модуль Wikipedia API.

Разработана программа, выполняющая считывание с клавиатуры исходных данных с помощью функции *input()* и метода *.split()* При реализации первой подзадачи использовались: Операторы: *if():*, *else():* , *not in*; Встроенная функция: *print();* Метод: *.keys();* Функция модуля Wikipedia: *languages()*. При реализации второй подзадачи использовались: Оператор*: if():*; Встроенные функции: *len(), range()*; Цикл: *for … in …;* Методы: *.pop(), .split()*; Функция модуля Wikipedia: *set_lang(lang), page(…);* Атрибуты класса WikipediaPage: *page.summary, page.title.* При реализации третьей подзадачи использовались: Операторы: *if():*, *else():* , *in*, *break*; Встроенные функции: *len(), range();* Цикл: *for … in …*; Метод *.append();* Пользовательская функция: *is_page_valid();* Функция модуля Wikipedia: *page(…);*Атрибут класса WikipediaPage: *page.links.* Ответ к поставленной задаче выводиться с помощью функции *print().*

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.py

```
import Wikipedia

def is_page_valid(page):
    try:
        wikipedia.page(page)
    except Exception:
        return False
    return True
max=0
maxt=''
Initial=input().split(', ')
res=[]
if Initial[-1] not in wikipedia.languages().keys():
    print("no results")
else:
    wikipedia.set_lang(Initial[-1])
    Initial.pop(-1)
    for i in range(len(Initial)):
        abb=(wikipedia.page(Initial[i]).summary.split())
        if len(abb)>=max:
            max=len(abb)
            maxt=wikipedia.page(Initial[i]).title
    res.append(Initial[0])
    for i in range(len(Initial)-1):
        lin=wikipedia.page(Initial[i]).links
        if Initial[i+1] in lin:
            res.append(Initial[i+1])
```

```
    else:
        for j in range(len(lin)):
            if  is_page_valid(lin[j]):
                if Initial[i+1] in wikipedia.page(lin[j]).links:
                    res.append(lin[j])
                    res.append(Initial[i+1])
                    break
print(max, maxt)
print(res)
```