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. 1382 _____ Шушков Е. В.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2021

Цель работы.

Изучение основных управляющих конструкций языка Python и модуля 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 больше ничего не делает. В случае, если язык есть, устанавливает его как язык запросов в текущей программе и выполняет еще два действия:

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. Предположим, нам на вход поступила строка (данный пример актуализирован к состоянию страниц wikipedia на 2021 год):

Айсберг, IBM, ru

В числе ссылок страницы с названием "Айсберг", есть страница с названием , которая содержит ссылку на страницу с названием "1959 год", у

которой есть ссылка на страницу с названием "IBM" -- это и есть цепочка с промежуточным звеном в виде страницы "1959 год".

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.

Пример входных данных:

Айсберг, IBM, ru

Пример вывода:

115 IBM

['Айсберг', '1959 год', 'IBM']

Первая строка содержит решение подзадачи №2, вторая - №3.

Важное уточнение: каждую подзадачу (1, 2, 3) оформите в виде отдельных функций. Функции должны быть "чистыми". Мы с этим определением ближе познакомимся в разделе №3 на лекциях, на данный момент следует выполнить требования:

1. Ваши функции не должны выводить что-либо на экран (только возвращать результат)

2. Ваши функции не должны изменять глобальные переменные (те переменные, которые существуют вне функции, то есть во внешней программе)

3. Ваши функции не должны изменять и свои аргументы, которые передаются в функцию (лучше возвращать измененную копию аргумента).

Выполнение работы.

В начале программы импортируем модуль wikipedia, переименовывая его в wk с помощью as.

Определяем функцию is_page_valid, которая получает на вход название страницы и проверяет, существуют ли она в Википедии.

Выполнение программы начинается с создание листа Page_names, элементы которого вводятся с клавиатуры и записываются в массив с помощью метода .split(“, “).

Дальше присваиваем переменной ans значение, которое возвращает функция Language_search, выполняющая первый подпункт задачи. Она получает на вход список Page_names с названиями страниц и языком (последний элемент). Проверяется, есть ли этот язык в википедии с помощью метода languages(), который возвращает сокращения для всех доступных языков. Если такой язык имеется, то с помощью метода set_lang() он устанавливается в качестве языка для модуля wikipedia в последующей программе. В таком случае функция возвращает результат функций Max_len_in_summary и Page_chains, которым на вход подаётся срез массива с первого элемента до элемента, в котором указан язык не включительно. Иначе возвращается “no results”.

Рассмотрим, что выполняет функция Max_len_in_summary. Это функция для решения второй подзадачи. Сначала определяем переменные max_len и max_len_name для количества слов в кратком содержании и название страницы соответственно. С помощью цикла for рассматриваем все элементы Page_names. Определяем переменную summary, в которой будет краткое содержание по названию страницы, которое мы получаем с помощью метода .summary(название страницы). Если количество элементов (ищем с помощью функции len()) в массиве, полученном с помощью метода .split() больше чем максимальное количество элементов max_len во время данной итерации, то

переприсваиваиваем max_len это количество и изменяем название страницы max_len_name, используя метод .title, который возвращает заголовок. Функция возвращает max_len и max_len_name.

Рассмотрим, что выполняет функция Page_chains. Это функция для решения третьей подзадачи. Сначала определяем массив chain_arr, в который будем сохранять цепочку, первый элемент уже занесен. Потом рассматриваем в цикле for все значения Page_names без языка, кроме последнего названия страницы, чтобы не выйти в дальнейшем за пределы. Для каждой страницы получаем лист links с ссылками из неё. Если следующее название страницы из Page_names находится среди них, то записываем его в массив chain_arr. Если нет, то для каждой ссылки из links рассматриваем второй лист sec_links с ссылками из них. И если страница с ссылки существует (проверяем с помощью функции is_page_valid) и следующее название страницы из Page_names есть среди них, то добавляем сначала промежуточную ссылку из первого листа, а потом и второй элемент из Page_names и останавливаем нынешний цикл. Таким образом, будут проверяться все элементы из Page_names на цепочку. Функция возвращает chain_arr.

В итоге, функция Language_search(Page_names) будет возвращать массив из двух массивов или “no result”. Если второй вариант, то через if выводим ans, которое равно “no results”. Else выводим сначала распакованный первый массив с помощью “*”, а на следующей строке второй массив. Так мы получаем результат программы.

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', '1959 год', 'IBM']	Ответ верный.
2.	Айсберг, IBM, apples	no results	Ответ верный.
3.	Айсберг, 1959 год, ru	73 Айсберг ['Айсберг', '1959 год']	Ответ верный.

Выводы.

Были изучены основные управляющие конструкции, а также модуль Wikipedia API.

Разработана программа, выполняющая считывание данных с клавиатуры пользователя и их обработку. Была реализована с помощью условных операторов *if-else*, циклов *for* для обработки массива и модуля Wikipedia API для работы программы со страницами одноимённой базы.

Приложение А

Исходный код программы

Название файла: lb1.py

```
import wikipedia as wk


def is_page_valid(page):
    try:
        wk.page(page)
    except Exception:
        return False
    return True


def Max_len_in_summary(page_names):
    max_len = 0
    max_len_name = ""
    for i in range(0, len(page_names)):
        summary = wk.summary(page_names[i])
        if len(summary.split()) >= max_len:
            max_len = len(summary.split())
                                        max_len_name    =
wk.page(page_names[i]).title
    return [max_len, max_len_name]


def Page_chains(page_names):
    chain_arr = [page_names[0]]
```

```
        for i in range(0, len(page_names) - 1):
            links = wk.page(page_names[i]).links

            if page_names[i + 1] in links:
                chain_arr.append(page_names[i + 1])
            else:
                for li in links:

                    sec_links = wk.page(li).links
                    if is_page_valid(li) and page_names[i
+ 1] in sec_links:
                        chain_arr.append(li)
                          chain_arr.append(page_names[i +
1])
                        break
        return chain_arr


    def Language_search(page_names):
        if page_names[-1] in wk.languages():
            wk.set_lang(page_names[-1])

                                                  return
[Max_len_in_summary(page_names[0:len(page_names)  -  1]),
Page_chains(page_names[0:len(page_names) - 1])]
        else:
            return ("no results")
```

```
Page_names = input().split(", ")
ans = Language_search(Page_names)
if ans == "no results":
    print(ans)
else:
    print(*ans[0])
    print(ans[1])
```