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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: «Основные управляющие конструкции. Wikipedia API».

Студент гр. 1304 _____ Байков Е.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы

Изучить основные управляющие конструкции 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 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 страницы “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, выводите 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 на вход поступила строка: 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 существует две цепочки, одна из которых содержит промежуточную страницу, а вторая нет, стройте цепочку без промежуточного элемента.

Выполнение работы

В самом начале программы импортируется модуль *wikipedia*. Затем описывается функция *is_page_valid(page)*, которая принимает на вход название страницы и проверяет существует ли эта страница, возвращая *True*, если страница существует или *False* в ином случае.

По заданию было описано три функции: *is_language_valid*, *max_count_words*, *create_chain*.

Первая возвращает значения *True/False*, а на вход ей подается значение *lang_form* (сокращенная форма языка). Внутри функции с помощью оператора вхождения проверяется нахождение этой формы в словаре *wikipedia.languges()*. Если язык не является одним из ключей словаря, то с помощью *print* выводится строка *"no results"* и возвращается *False*.

Вторая возвращает список из максимального количества слов в описании страницы и ее *title*. На вход подается список страниц (*some_pages*), которые надо обработать. Внутри функции создаются две переменные: *max_len* – хранит в себе максимальное количество слов (изначально присваивается значение 0), *wiki_title* – хранит в себе название страницы, на которой обнаружилось наибольшее количество слов (изначально присваивается пустая строка). С помощью цикла *for* перебираются все элементы в списке, который был подан на вход. С помощью функции *is_page_valid* проверяется существование страницы. Если она не существует, возвращается значение *False*, иначе заводится переменная *words*, которая хранит в себе длину списка, состоящего из всех слов в кратком описании данной страницы (*element*). Затем проверяется какое из значений больше *words* или *max_len*, в случае того, если первое больше или равно второму, то *max_len*

присваивается значение *words*, а *wiki_title* принимает значение *wikipedia.page(element).title* (*title* данной страницы).

Третья возвращает цепочку, состоящую из страниц и промежуточных страниц. На вход ей подается список страниц (*some_pages*). Внутри объявляется список *chain,* который состоит из одного элемента – нулевого элемента из списка, полученного на вход. Затем с помощью цикла *for* перебираются все значения, полученного списка кроме последнего элемента. Объявляется список *links_on_page*, которой хранит в себе все ссылки, находящиеся на данной странице (*some_pages[i]*). Затем с помощью оператора вхождения *in* проверяется наличие *some_pages[i+1]* среди ссылок на предшествующей ей странице. Если ссылка на страницу находится в списке, то в *chain* прибавляется значение *some_pages[i+1]*, иначе с помощью цикла *for* перебираются все элементы из списка *links_on_page*. С помощью оператора вхождения проверяется наличие ссылки на страницу *some_pages[i+1]* и с помощью функции *is_page_valid* проверяется существование просматриваемой страницы (*linker_page*). Как только находится нужная страница, к значению *chain* прибавляется список *[linker_page, some_pages[i+1]]* и с помощью *break* цикл завершается досрочно. Функция возвращает значение *chain*.

Затем объявляется переменная *pages*, которая будет хранить в себе список из строк, введенных пользователем. С помощью условного оператора и функции *is_language_valid*, которая принимает на вход последний элемент списка, убранный из списка с помощью метода *pop*. Если функция возвращает *True*, то программа с помощью функций *print* выводит значения двух других функций, которые были описаны.

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']	-
2.	Франция, Страна, Египет, ru	197 Франция ['Франция', 'ISO 3166-1', 'Страна', 'Алфавитный список стран и территорий', 'Египет']	-
3.	Айсберг, IBM, ssssssddsd	no results	-
4.	Edgar Allan Poe, Romanticism, Europe, en	609 Europe ['Edgar Allan Poe', 'Romanticism', 'Europe']	-
5.	Gothic (series), Piranha Bytes, Cologne, en	577 Cologne ['Gothic (series)', 'Piranha Bytes', 'Bochum', 'Cologne']	-

Выводы

Изучены основные управляющие конструкции языка Python и принцип работы с Wikipedia API.

Была написана программа, выполняющая считывание с клавиатуры исходных данных которые поступают на вход программы в виде строки: название_страницы_1, название_страницы_2, … название_страницы_n, сокращенная_форма_языка, а затем обрабатывает ее согласно заданиям.

Было написано 2 функции (не считая *is_page_valid*): *max_count_words*, которая принимает на вход список страниц и возвращает список вида: *[максимальное_количество_слов_в_описании, title_страницы]*, и *create_chain*, принимающая на вход список из страниц и возвращающая список страниц, которые представляют цепочку из ссылок.

Вывод происходит с помощью функции *print* и использованием f-строки.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Baykov_Egor_lb1.py

```
import wikipedia

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

def is_language_valid(lang_form):
    if lang_form not in wikipedia.languages():
        print("no results")
        return False
    else:
        wikipedia.set_lang(lang_form)
        return True

def max_count_words(some_pages):
    max_len = 0
    wiki_title = ''
    for element in some_pages:
        if is_page_valid(element):
            words = len(wikipedia.page(element).summary.split())
            if words >= max_len:
                max_len = words
                wiki_title = wikipedia.page(element).title
        else:
            return False
    return [max_len, wiki_title]

def create_chain(some_pages):
    chain = [some_pages[0]]
    for i in range(len(some_pages)-1):
        links_on_page = wikipedia.page(some_pages[i]).links
        if some_pages[i+1] in links_on_page:
            chain += [some_pages[i+1]]
        else:
            for linker_page in links_on_page:
                if some_pages[i+1] in wikipedia.page(linker_page).links
and is_page_valid(linker_page):
                    chain += [linker_page, some_pages[i+1]]
                    break
    return chain

pages = input().split(', ')
if is_language_valid(pages.pop()):
    print(f"{max_count_words(pages)[0]} {max_count_words(pages)[1]}")
    print(create_chain(pages))
```