МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Математического обеспечения электронно-вычислительных машин

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

ТЕМА: ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ. WIKIPEDIA API

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Отработать на практике базовые принципы построения программ. Изучить и научиться работать с основными управляющими конструкциями языка Python, а также с Wikipedia API.

Задание.

Напишите программу, выделив каждую подзадачу в отдельную функцию.

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.

Предположим, нам 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 "Буран", у которой есть ссылка на страницу с названием "IBM" -- это и есть цепочка с промежуточным звеном в виде страницы "Буран".

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', 'Буран', 'IBM']

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

Основные теоретические положения.

Функция	Описание	Возвращаемое значение
page(title)	Поиск страницы	Объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой - строка title
languages()	Поиск всех возможных языков сервиса	Словарь, ключами которого являются сокращенные названия языков, а значениями - названия. Например: *>>> wikipedia.languages()['ru']* *'русский'*
set_lang(lang)	Установить язык lang, как язык	None

	запросов в текущей программе.	

Ниже представлены атрибуты класса WikipediaPage (страницы сервиса Wikipedia):

Поле класса	Описание	Возвращаемое значение
page.summary	Краткое содержание страницы page	Строка
page.title	Название страницы page	Строка
page.links	Список названий страниц, ссылки на которые содержит страница page	Список строк

Выполнение работы.

1. Обработаем входные данные, так как на вход подается строка, с помощью метода строк *split()* преобразуем эту строку в список, который будет храниться в переменной s(разделитель при этом – комбинация символов ‘, ‘)
2. Далее необходимо проверить, есть ли введенный язык в возможных языках сервиса. Для этого обратимся к последнему элементу списка через отрицательный индекс *s[-1]*. И вызовем нашу «будущую»(так как определение запишем следующим шагом) функцию set_language() для этого элемента.
3. Определим функцию *set_language*().

Функция принимает аргумент – строку с сокращенным названием языка. Функция сразу возвращает значение - результат проверки на вхождение этого языка в ключи словаря, в которых хранятся сокращенные названия возможных языков сервиса.

4.Если введенный язык не существует(возвращено значение *False*), то программа выведет *‘no results’* и завершит свое выполнение, если существует(возвращено *True*), то с помощью функции модуля wikipedia.set_lang(s[-1]) этот язык установится в качестве языка запросов и программа продолжит свое выполнение.

5.Далее в переменную *names* помещаем срез списка *s* с 1 по предпоследний элемент. То есть в *names* у нас будет храниться список названий страниц.

6.Нам нужно определить максимальное количество слов в кратком содержании страницы и страницу, у которой оно максимально. Для этого вызываем нашу «будущую» функцию search_max() для списка страниц.

7. Определим функцию search_max().

Функция принимает аргумент - список названий страниц

Инициализируем переменные *m= -1, title=’’,* которые будут в дальнейшем хранить максимальное количество слов в кратком содержании страниц и название страницы, у которой это число максимально.

Далее циклом *for* перебираем элементы списка названий страниц, т.е. сами названия страниц.

Нужно оценить количество слов в кратком содержании итерируемой страницы. Так как функция *wikipedia.page(i).summary* будет возвращать нам строку, нам нужно разделить слова этой строки по пробелу между ними с помощью функции split(). Получим список слов краткого содержания этой страницы. Следовательно, можем посчитать их количество с помощью функции len(), которая будет считать количество элементов полученного списка. В общем виде мы запишем так: *len(wikipedia.page(i).summary.split()).*

Если данная величина будет больше *m,* то на данной итерации в *m* запишется значение той величины. А в *title мы* запишем название итерируемой страницы с помощью функции wikipedia.page(i).title.

Создадим список *res,* в который запишем полученные после всех итераций значения *m* и *title*(строкового типа, чтобы в дальнейшем применить метод строк .join())

Функция будет возвращать список *res.*

8. Чтобы вывести на экран результат работы функции search_max(names), два значения: *m* и *title* через пробел, применим метод ' '.join(search_max(names)), и все это «обернем» функцией print()

9.Далее нам надо вывести на экран список-цепочку, требуемую в подзадаче 3. Для этого вызовем нашу «будущую» функцию sequence(), передав ей в качестве аргумента список названий страниц *names* и выведем на экран возвращенное значение.

10.Определим функцию sequence().

Функция принимает в качестве аргумента список названий страниц.

Инициализируем переменную res=[], в которой будет храниться список-искомая цепочка.

Добавим в этот список элемент – название первой страницы.

Далее с помощью цикла *for* будем перебирать названия страниц.

Если следующее по счету название страницы(L[i+1]) содержится в ссылках первой страницы(L[i]), то значит можно обойтись без промежуточной страницы, следовательно, записываем следующим «звеном» цепочки следующую страницу(*res.append(L[i+1])*.

Если же следующее название страницы(L[i+1]) не содержится в ссылках первой страницы(L[i]), значит, для перехода нужна промежуточная страница. Следовательно, будем искать название следующей страницы(L[i+1]) среди ссылок страниц, ссылки на которые есть в первой странице(L[i]). То есть организуем вложенный цикл for, перебирающий ссылки первой страницы:

for j in wikipedia.page(L[i]).links

Далее проверяется, если название следующей страницы есть среди ссылок страницы(*if L[i+1] in wikipedia.page(j).links*), то j – название промежуточной страницы. Значит, добавим его в res, а также название страницы, до которой искался «путь»(res.append(L[i+1]). Будем прерывать перебор страниц, если промежуточная страница найдена, с помощью break.

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']	Верно, так как язык запросов установился - русский, максимальное число слов в кратком содержании страницы IBM – 115 слов, цепочка кратчайшая возможная
2.	Gravitational_acceleration, Gravity, en	459 Gravity ['Gravitational_acceleration', 'Acceleration', 'Gravity']	Верно, так как язык запросов установился - английский, максимальное число слов в кратком содержании страницы Gravity – 456 слов, цепочка кратчайшая возможная

Выводы.

Были отработаны базовые принципы построения программ, а также исследованы и изучены основные управляющие конструкции языка, Wikipedia API.

Разработана программа, выполняющая считывание с клавиатуры названий страниц и языка. Для работы с сервисом Wikipedia подключался модуль Wikipedia. Выполнение каждой подзадачи реализовывалось в соответствующей функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

def set_language(str):
    return str in wikipedia.languages()

def search_max(L):
    m,title=-1,''
    for i in L:
        if len(wikipedia.page(i).summary.split())>=m:
            m=len(wikipedia.page(i).summary.split())
            title=wikipedia.page(i).title
    res=[str(m),title]
    return res

def sequence(L):
    res=[]
    res.append(L[0])
    for i in range(len(L)-1):
        if L[i+1] in wikipedia.page(L[i]).links:
            res.append(L[i+1])
        else:
            for j in wikipedia.page(L[i]).links:
                if L[i+1] in wikipedia.page(j).links:
                    res.append(j)
                    res.append(L[i+1])
                    break
    return res

s=input().split(', ')
if set_language(s[-1]):
    wikipedia.set_lang(s[-1])
    names = s[:len(s) - 1]
    print(' '.join(search_max(names)))
    print(sequence(names))

else:
    print('no results')
```