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. 0304 _____ Докучаев Р.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2020

Цель работы.

Изучить основные управляющие конструкции языка, функции, модули и встроенные функции языка Python.

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 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.

Используется модуль Wikipedia, с помощью которого можно работать wiki-страничками сервиса Wikipedia. Ввод данных осуществляется благодаря функции *input()*, вывод данных с разделением на отдельные строки осуществляется с помощью функции *print()*. Также используются управляющие конструкции *if*, *for*, *while*, *return*, ключевое слово для определения функции: *def*. Также используются функция *len()* и срезы. Для досрочного выхода из программы используется *exit()*, для продолжения работы программы: *continue*.

Выполнение работы.

1. Подключение модуля *Wikipedia*
2. Создание функции *is_page_valid()*, которая проверяет существование введённой страницы.
3. Создание функции *is_language_valid(lang)*, которая принимает в качестве параметра строку с названием языка. Функция проверяет наличие языка из ввода в языках *Wikipedia*. Если страница существует, то программа возвращает значение *lang*, иначе программа выводит на экран *'no results'* и досрочно завершает работу.
4. Создание функции *max_words_summary(pages)*, которая определяет страницу с наибольшим количеством слов в кратком описании страницы. На вход принимает список страниц, а возвращает страницу наибольшего краткого описания и количество срок в нем.
5. Создание функции *find(page_1, page_2)*, которая принимает на вход два объекта *WikipediaPage*, далее определяет промежуточную

страницу между двумя соседними страницами и возвращает либо заголовок промежуточной вики-страницы в формате строки, либо объект класса *None*.

6. Создание функции *list_page(pages)*, которая принимает список строк-названий страниц, а возвращает модифицированный список, в который были вставлены недостающие промежуточные страницы. Функция требуется для создания списка-цепочки страниц.
7. Создается основная часть программы.
    1) переменной *a* присваивается значение, возвращаемое функцией *input().split(', ')*. Это входные данные программы, причем последний элемент *a* – язык, который будет использоваться в *Wikipedia*
    2) после этого программа с помощью функции *set.languages(is_language_valid(lang))* устанавливает данный язык
    3) если такого языка не существует, программа выводит *“no results”* и завершает работу
    4) после этого в переменную *result*, отвечающую за результат выполнения программы, записываются значения, полученные фунцкией *max_words_summary* и выводятся на экран при помощи функции *print()*
    5) затем при помощи еще одного вызова функции *print()* выводится список wiki-страниц, возвращенный функцией *list_page*

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']	OK
2.	England, en	451 England ['England']	OK
3.	Россия, США, Корея	No results	OK

Выводы.

Были изучены основные управляющие конструкции языка Python, встроенные функции Python, модули, условия, функции.

Была разработана программа, которая считывает данные, которые вводит пользователь с консоли. Затем программа при помощи работы с wiki-страницами из модуля Wikipedia осуществляет работу с исходными данными и преобразует их: устанавливает введенный язык, находит среди перечисленных страницу с наибольшим кратким содержанием и добавляет недостающие страницы. После этого программа выводит на экран результаты всех преобразований.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import wikipedia

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

def is_language_valid(lang):
    if lang in wikipedia.languages():
        return lang
    else:
        print('no results')
        exit()

def max_words_summary(pages):
    maxi = 0
    maxi_page = None
    for i in pages:
        page = wikipedia.page(i)
        words = page.summary.split()
        if len(words) >= maxi:
            maxi = len(words)
            maxi_page = page.title
    return maxi, maxi_page

def find(page_1, page_2):
    for i in page_1.links:
        if not is_page_valid(i):
            continue
        page_i = wikipedia.page(i)
        if page_2.title == page_i.title:
            return None
        if page_2.title in page_i.links:
            return i
```

```
            return "ERROR"

        def list_page(pages):
            b = 0
            while b < len(pages) - 1:
                c = find(wikipedia.page(pages[b]), wikipedia.page(pages[b
+ 1]))
                if c is not None:
                    pages.insert(b + 1, c)
                    b += 1
                b += 1
            return pages

        a = input().split(', ')
        lang = a[-1]
        wikipedia.set_lang(is_language_valid(lang))

        result = max_words_summary(a[0:len(a) - 1])
        print(result[0], result[1])
print(list_page(a[0:len(a) - 1]))
```