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. 1304 _____ Шаврин А.П

Преподаватель _____ Берленко Т.А

Санкт-Петербург

2021

Цель работы.

Изучить основные управляющие конструкции языка Python, функции, типы данных, циклы, условные операторы и получить навыки работы с ними, а также научиться работать с новым модулем Wikipedia.

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).

Основные теоретические положения.

В данной лабораторной работе нам предстоит работать с модулем wikipedia, который позволяет программно работать с wiki-страничками сервиса Wikipedia. Ниже представлены функции модуля wikipedia, которые могут нам понадобиться:

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 запросов в текущей программе.	None

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.

Импорт библиотеки *Wikipedia*.

Считывание входных данных, с помощью метода строки *split()* преобразуем данные в список, который записывается в переменную *inp*.

В переменную *titles* с помощью среза копирую только список названий страниц, а в переменную *lang* передаю строку введенного пользователем языка.

Для решения первой задачи создается функция *is_in_lang_wiki()*, которая принимает в качестве аргумента переменную, содержащую информацию о выбранном пользователем языке (*lang*). Функция проверяет наличие введенного языка в возможных языках сервиса *Wikipedia* и возвращает значения *True*, если есть или *False*, если нет.

Для решения второй задачи создается функция *max_words()*, которая принимает в качестве аргумента список названий страниц (*titles*). В функции создаются переменные *mx_words=0* и *mx_title* (типа *str*) для записи максимального кол-ва слов в краткой записи страницы и названия этой страницы соответственно. С помощью цикла *for* просматриваются краткие содержания всех страниц. Обе локальные переменные возвращаются функцией в виде кортежа.

Для решения третьей задачи создается функция *get_chain()*, которая принимает в качестве аргумента список названий страниц (*titles*). В функции создается список *chain*, который сразу заполняется первым названием страницы из списка *titles*. Затем с помощью цикла *for* функция проходит по всем названиям страниц из списка *titles*. В цикле создается список *links*, который заполняется ссылками находящимися на текущей странице. После этого с помощью условного оператора *if-else* проверяется наличие следующего названия страницы из списка *titles* в списке ссылок текущей страницы (*links*). Если такая ссылка присутствует, то в список *chain* добавляется следующее название страницы из списка *titles* и происходит переход к следующей итерации. В случае, если такой ссылки нет на текущей странице, другой цикл *for* проходит по списку ссылок текущей страницы (*links*). Каждая ссылка проверяется на существование на сайте *Wikipedia* функцией *is_page_valid()*, которую уже предоставили вместе с заданием работы. В случае отсутствия страницы по данной ссылке происходит переход к следующей итерации. Если же страница по ссылке существует, то в переменную *intermediate_page* записывается промежуточная страница, а в список *intermediate_links* записываются ссылки с промежуточной страницы. Далее идет проверка на существование ссылок: если их нет, происходит переход к новой итерации, если

они есть, то при помощи условного оператора *if* проверяется наличие следующего названия страницы из списка *titles* в списке ссылок промежуточной страницы (*intermediate_links*). Если такая ссылка находится, то в список *chain* записывается сначала промежуточная страница (*intermediate_page*), затем следующее название страницы из списка *titles*. Цикл for, проходящий по списку ссылок текущей страницы (*links*) прерывается оператором *break*. В случае, если такой ссылки не найдено, происходит переход цикла, проходящего по списку названий страниц, к следующей итерации.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Я изучил основные управляющие конструкции языка Python, функции, типы данных, циклы, условные операторы и получил навыки работы с ними, а также научился работать с новым модулем Wikipedia.

Мною была написана программа, считывающая с клавиатуры исходные данные, команды пользователя и выполняющая ряд поставленных задач.

Для обработки данных и команд пользователя использовались условные операторы *if-else*, цикл for, оператор break, а для отлавливания исключительных ситуаций был использован блок *try-except* в функции *is_page_valid()*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Lab1.py

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

def is_in_lang_wiki(l):
    lang_wiki = wikipedia.languages()
    return l in lang_wiki

def max_words(titles):
    mx_words = 0
    mx_title = ''
    for title in titles:
        page = wikipedia.page(title)
        summary = page.summary.split()
        count_words = len(summary)
        if mx_words <= count_words:
            mx_words = count_words
            mx_title = page.title
    return mx_words, mx_title

def get_chain(titles):
    chain = [titles[0]]
    for i in range(len(titles)-1):
        links_list = wikipedia.page(titles[i]).
```

```
        if titles[i+1] in links_list:
            chain.append(titles[i+1])
        else:
            for j in range(len(links_list)):
                if not is_page_valid(links_list[j]):
                    continue
                intermediate_page = wikipedia.page(links_list[j])
                intermediate_links = intermediate_page.links
                if not intermediate_links:
                    continue
                if titles[i+1] in intermediate_links:
                    chain.append(links_list[j])
                    chain.append(titles[i+1])
                    break
    return chain

def main():
    inp = input().split(', ')
    lang = inp[-1]
    titles = inp[:-1]
    if is_in_lang_wiki(lang):
        wikipedia.set_lang(lang)
        print(*max_words(titles))
        print(get_chain(titles))
    else:
        print('no results')

if __name__ == '__main__':
    main()
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.2 – Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Айсберг, Роскартография, Смоуправление, ru	131 Самоуправление ['Айсберг', 'Южный океан', 'Роскартография', 'Орган государственной власти', 'Самоуправление']	Все выведенные данные верны.
2.	Россия, Обряд, Акционерное общество, ru	426 Россия ['Россия', 'Венчание на царство', 'Обряд', 'Большая советская энциклопедия', 'Акционерное общество']	Все выведенные данные верны.
3.	Автор, Авторское право, Фильм, ru	198 Фильм ['Автор', 'Авторское право', 'Аудиовизуальное произведение', 'Фильм']	Все выведенные данные верны.
4.	Айсберг, IBM, ru	115 IBM ['Айсберг', '1959 год', 'IBM']	Все выведенные данные верны.
5.	Алгоритм, Астроном, Галактика, Скрытая масса, ru	401 Галактика ['Алгоритм', 'Аль-Хорезми', 'Астроном', 'Астрономия', 'Галактика', 'Скрытая масса']	Все выведенные данные верны.
6.	God, Atheism, eng	519 Atheism [' God ', Atheism '']	Все выведенные данные верны.
7.	Aurum, gold, en	No results	Все выведенные данные верны.