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. 0382 _____ Азаров М.С.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Изучить основные управляющие конструкции языка Python и получить общее представление о принципах программирования на языке Python.

Задание.

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.

Основные теоретические положения.

При выполнении поставленной задачи были использованы следующие конструкции языка :

1. Встроенные функции и методы языка:
   - *print()* - преобразует объекты и выводит их в поток стандартного вывода или записывает их в файл.
   - *Input()* - для ввода данных с клавиатуры
   - *len()* - возвращает целое число (int) равное количеству элементов объекта.
   - *range()* - генерирует арифметическую прогрессию в данном диапазоне с данным шагом.
   - *S.split(символ)* - Разбиение строки по разделителю
   - *list.append(x)* - Добавляет элемент в конец списка
2. Функции модуля *Wikipedia:*
   - *wikipedia.page(title)* - Объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой - строка title
   - *wikipedia.languages()* - Поиск всех возможных языков сервиса.Словарь, ключами которого являются сокращенные названия языков, а значениями — названия.

- *wikipedia.set_lang(lang)* - Установить язык *lang*, как язык запросов в текущей программе.

3. Поля модуля *Wikipedia* :
   - *page.title* - Название страницы page.
   - *page.links* - Список названий страниц, ссылки на которые содержит страница page.
   - *page.summary* - Краткое содержание страницы page.
4. Сторонние функции :
   - *is_page_valid(page_name)* - которая возвращает *True*, если wiki-страница с названием *page_name* существует и *False* в ином случае.
5. Описание собственные функции:

def <имя_функции>(<аргументы>):
<инструкции>
…
<инструкции>
return <значение_1>, … <значение_n> # необязательная часть

Здесь:

- def - инструкция, которая сообщает интерпретатору, что начинается определение функции;
- <имя_функции> - наименование функции;
- <аргументы> - параметры, которые передаются в функцию и которые вы можете использовать в функции;
- <инструкции> - код, который выполняется при вызове функции. Обратите внимание, что он расположен после отступов с начала строки, как и у любой составной инструкции ;
- <значения> - это данные, которые передаются обратно в программу, где был совершен вызов функции. Здесь необходимо обратить внимание, что данная инструкция не обязательна.

● *return* - оператор, который возвращает данные и управление в место вызова функции.

6. Управляющие конструкции языка :

- Ветвление :

  if <условие 1>:

  <действие 1> # Данный блок сработает, если <условие 1> истинно

  else :

  <дейтсвие 2> # Данный блок сработает, если <условие 1> ложно

- Цикл for:

  for <переменная> in <коллекция>:

  <действие 1>

  else: # необязательный блок

  <действие 2>

  Цикл for в предназначен для перебора элементов, находящихся в коллекции. С каждым элементом, последовательно выполняться одно и тоже действие, указанное в теле цикла for.

Выполнение работы.

Глобальные переменные :

- *names* — список названий страниц введенных в программу
- *lang* — строка , хранящая сокращенную форма языка, введенного в программу .
- *tit* — название страницы с максимальным количеством слов в кратком содержании.
- *maxw* — максимальное количество слов в кратком содержании страниц.

- *cur_ch* - список-цепочка страниц построенная по алгоритму из 3 подзадачи

Собственные функции :

- *find_lang(lang)* — функция проверяет наличие языка *lang* в в возможных языках сервиса. Если такой язык есть , возвращает значение True , в противном случае возвращает False .
  - Структура функции :

    С помощью конструкции *if-else* и операторы принадлежности *in* , а также оператора *return* проверяется наличие языка lang в списке доступных языков wikipedia.languages().
- *max_word(names)* — функция находит страницу с самым большим количеством слов в кратком содержании. И возвращает количество слов краткого содержания и название этой страницы.
  - Локальные переменные :
    - *names* — список введенных имен страниц .
    - *maxw* — переменная хранящая максимальное количество слов краткого содержания из известных страниц
    - *i* — имя текущей страницы в цикле.
    - *page* — объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой текущее значение *i.*
    - *tit — название* странички с самым большим количеством слов в кратком содержании в данный момент .
  - Структура функции :

    Обнуляется значение *maxw*, чтобы избавится от мусора в перемной и иметь возможность сравнивать ее с другими переменными .

    Используется цикл *for* и оператор *in* для перебора списка *names* в переменную *i* .

В теле цикла переменной page присваивается объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой текущее значение *i.* Затем используется конструкция *if* для того чтобы если maxw (максимальное количество слов краткого содержания из известных страниц) меньше количества слов краткого содержания текущей страницы *page* ,то присвоить maxw количества слов краткого содержания текущей страницы *page,*а *tit* присвоить название этой страницы *page.* Тело цикла заканчивается .

С помощью оператора *return* вернуть вместо функции значение переменных *maxw* и *tit* .

- *page_chain(names)* - Строит список-цепочку из страниц согласно алгоритму из 3 подзадачи и возвращает полученный список .
  - Локальные переменные:
    - *names* — список введенных имен страниц .
    - *cer_ch* — список-цепочка страниц построенная по алгоритму из 3 подзадачи.
    - *i* — счетчик и индекс текущего имени в списке *names*.
    - *page* — объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой текущее значение *names[i]*.
    - *l* - текущая ссылка из списка ссылок страницы page .
    - *subpage* - объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой текущее значение *l* .
  - Структура функции :

    Присваивание *cer_ch* первого элемента *names* как старт .

Начало цикла for где i индекс текущего имени в списке *names* начиная с 0 до предпоследнего элемента. *page* присваивается объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой текущее значение *names[i]*. После идет конструкция *if-else* ,которая проверяет если следующее имя *names[i+1]* есть в списке ссылок текущей страницы page , то добавить это имя *names[i+1]* к списку *cer_ch.*

Иначе вложенный цикл *for* , который перебирает каждую ссылку текущей *page* в *l* . Тело цикла : проверяется существует ли страница по ссылке *l*. Если да то *subpage* присваивается объект класса WikipediaPage, название которой текущее значение *l,* и дальше если существует имя names[i+1] в списке ссылок *subpage* ,то в список *cer_ch* добавляется ссылка *l* и имя names[i+1] и оператор break заканчивает работу вложенного цикла .Тело вложенного цикла заканчивается.

Тело цикла заканчивается .Оператор *return* возвращает *cer_ch* список-цепочка страниц построенная по алгоритму из 3 подзадачи.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Айсберг, IBM, ru	115 IBM ['Айсберг', 'Буран', 'IBM']	Программа работает правильно
2.	Айсберг, Буран, ru	73 Айсберг ['Айсберг', 'Буран']	Программа работает правильно
3.	Айсберг, IBM, qwer	no results	Программа работает правильно

Выводы.

Изучены основные управляющие конструкции языка Python и получены общее представление о принципах программирования на языке Python.

Разработана программа, выполняющая считывание с клавиатуры исходных данных и установку языка,если это возможно , нахождение страница с самым большим количеством слов в кратком содержании и построение список-цепочек согласно подзадаче 3 . Для установки языка использовалась собственная функция *find_lang(lang),* которая в случае е*сли язык не был найден , возвращала false* . Используя это и ветвление if-else , при не нахождении языка выводилось *«no results»,* в противном случае программа продолжала работу. С помощью собственной функция max_word(names) находилась название и странички с максимальным количеством слов в кратком содержании и ее количество слов. А с помощью собственной функции *page_chain(names)* строились список-цепочек согласно подзадаче 3.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: inf_lab1.py

```
import wikipedia


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


def find_lang(lang):
    if lang in wikipedia.languages():
        return True
    else:
        return False


def max_word(names):
    maxw = 0
    for i in names:
        page = wikipedia.page(i)
        if maxw <= len(page.summary.split()):
            maxw = len(page.summary.split())
            tit = page.title
    return maxw, tit


def page_chain(names):
    cer_ch = [names[0]]
    for i in range(len(names)-1):
        page = wikipedia.page(names[i])
        if names[i+1] in page.links :
             cer_ch.append(names[i+1])
        else :
            for l in page.links:
                if is_page_valid(l) :
                    subpage = wikipedia.page(l)
                    if names[i+1] in subpage.links :
                        cer_ch.append(l)
                        cer_ch.append(names[i+1])
                        break
    return cer_ch

#Начало осн функции

*names, lang = input().split(', ')

if find_lang(lang):
    wikipedia.set_lang(lang)
```

```
    maxw, tit = max_word(names)
    print(maxw, tit)
    cur_ch = page_chain(names)
    print(cur_ch)

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