МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Математического Обеспечения и Применения ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Информатика»

Тема: Основные управляющие конструкции. Wikipedia API

Студент гр. 0382 _____ Павлов С. Р.

Преподаватель _____ Шевская Н. В.

Санкт-Петербург

2020

Цель работы.

Изучить основные управляющие конструкции Python и модуля 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 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.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.

В данной лабораторной работе были использованы следующие конструкции языка python:

- Встроенные функции Python:
  - *input()* - возвращает считываемое с консоли значение
  - *print()* - выводит на консоль принимаемое в качестве аргумента значение
  - *len()* - возвращает длину объекта, принятого в качестве аргумента
  - *range()* - возвращает последовательность чисел в заданном диапазоне с заданным шагом
- Встроенные методы Python:
  - *str.split()* - возвращает список всех слов в строке, используя значение аргумента в качестве разделителя (по умолчанию это пробел)
  - list.append() - добавляет переданный obj в существующий список.
  - str.strip() - удаляет се пробелы в начале и конце строки str
- Операторы:
  - *if*: <последовательность действий 1> else: <последовательность действий 2>— если значение выражения после оператора if и перед двоеточием - true, выполняет блок кода с одинаковым уровнем отступа после if, если false – блок кода после *else*
  - in — если объект перед оператором является подстрокой или элементом объекта после оператора – значение выражения – true, в противном случае – false
  - break —прерывает выполнение цикла
  - return — используется в функциях для возвращения каких-либо значений
- Циклы:

- or <переменная> in <итерируемый объект>:— для каждогозначения переменной, находящегося в итерируемом объекте,выполняет блок кода с одинаковым уровнем отступа после двоеточия

- Функции Wikipedia API:
  - page(title) – возвращает объект класса WikipediaPage, которыйпредставляет собой страничку сервиса Wikipedia, название которой - строка title
  - languages() –возвращает словарь, ключами которого являются сокращенные названия языков сервиса, а значениями – полные названия
  - set_lang(lang) – устанавливает язык lang как язык запросов в текущей программе
- Обращения к полям:
  - page.summary— поле класса page модуля Wikipedia, возвращаетмногострочный литерал – краткое содержание страницы page
  - page.title — поле класса page модуля Wikipedia, возвращает строку – название страницы page
  - page.links—поле класса page модуля Wikipedia, возвращает список строк – названий страниц, ссылки на которые содержит страница page

Выполнение работы.

В самом начале программы нужно подключить модуль *wikipedia.*

Первым делом для выполнения условий, поставленных задачей необходимо считать строку входных данных.

Для считывания входных данных используются переменные:

- *n* — является массивом строк (входных данных), в этой переменной хранятся строки (*название_страницы_1, название страницы_2, ... название_страницы_n, сокращенная_форма_языка*). Запись строк в массив осуществляется с помощью функции input() и метода split().

Далее для обработки входных данных создаются другие переменные, такие как *pages* и *lang.*

Создаеться массив pages, в который копируются все значения переменной *n*, это делается с помощью метода copy(). Затем, с помощью метода pop(), из массива *pages,* удаляется последний элемент, который является скоращеной формой языка.

Так же создается переменная *lang* типа строки, которая принимает значение последнего элемента из переменной *n (lang = n[-1]).*

Таким образом следующие перменные:

- *pages* — массив, хранящий в себе названия страниц;
- *lang* — строка, хранящая сокращенную форму языка;

1. Выполнение первой подзадачи программы. Проверка наличия введеного языка в системе сервиса Wikipedia. Это реализуется с помощью условной инструкции *if.*

Если условие (lang in wikipedia.languages()) - Истинно, то это значит что в сервисе Wikipedia, есть такой язык. Так же указаный язык устанавливаеться в систему поиска и обработки страниц с помощью команды wikipedia.set_lang(lang). Далее после успешной установки языка, происходит вызов двух функций print((max_words(pages))[0],(max_words(pages))[1]), print(links_chain(pages)) которые принимают значение решения 2-го и 3-го пункта подзадач программы соответственно.

Или же, если выражение — Ложно, (т.е указаный в вводе язык не входит в систему сервиса Wikipedia), то программа вывод на экран «no results» и завершает свою работу.

2. Выполение второй подзадачи программы. Решение второй подзадачи является значение возвращенной функцией *max_words(pages)*.

В качестве аргумента принимает массив строк, которые являются названиями страниц. Так же объявляется массив *maximum*, который принимает значение — [0,"name"].

Далее с помощью цикла *for* с каждой иттерацией в переменную *i* записываются индексы элементов массива, затем объявляется временная переменная page, которая является странницей Wikipedia, т.е *page* = wikipedia.page(name), где *name* являеться i-ным элементом массива (name = pages[i]), так же объявляется переменная *count,* которая хранит значение кол-ва слов в кратком содержании страницы, т.е *count* = len((page.summary).split()), метод split() используеться для того чтобы записать слова в массив, а функция len(), считает количество элементов массива = слов. После объявления временных переменных *page,* и *count,* в цикле с помощью условия if maximum[0] <= count, сравниваеться кол-во слов максимума и i-ной страницы, если в странице больше слов чем в максимуме, то первый элемента массива *maximum[0]* принимает значение *count,* и второй элемент название страницы (в которой больше слов), *maximum[1]* = page.title . Или же, если условие ложно, т.е кол-во слов в максимуме больше чем в данной странице, то цикл просто переходит на другую иттерацию. Таким образом будет найдена страница с максимальным количеством слов в кратком содержании.

С помощью оператора *return* функцией *max_words* будет возвращено значение массива *maximum*.

Которое будет передано в функцию print((max_words(pages))[0], (max_words(pages))[1]), где сначала будет выведено кол-во страниц, а затем ее название.

3. Выполение третьей подзадачи программы. Решением третьей подзадачи является значение возвращенной функцией *links_chain(pages)*.

В качестве аргумента принимает массив строк, которые являются названиями страниц. Так же объявляется массив *ans*, который принимает значение названия первой страницы в цепочке, т.е pages[0].

Далее с помощью цикла *for* с каждой иттерацией в переменную *i* записываются индексы элементов массива до последнего, начиная с нуля заканчивая (len(pages)-1).

Все описаное ниже будет являться частью тела основного цикла, поэтому чтобы не запутаться основый цикл будет обозначаться как **for-1** , а вложенный **for-2**.

Основной цикл **for-1** начинает каждую иттерацию, с Объявляния переменных. Переменная *A*, являеться i-ной страницей (начальным звеном в цепи сыллок на след. страницу) , и принимает значение wikipedia.page(pages[i]); Переменная *B,* являеться названием второя страницы, принимает строку pages[i+1]. Так же создаеться переменная A_links, массив содержащий в себе сыллки этой страницы, прнимает значение с помощью метода links.

Далее после объявления переменных начинаеться основная работа функции. В первую очередь после объявления переменных проверяется, есть ли прямая сыллка из первой страницы во вторую, с помощью *if*, проверяеться условие (B in A_links).

Если — True, то тогда в массив *ans,* добавляеться название второй страницы. Т.е между первой и второй страницей, нет промежуточного звена в виде другой страницы, а есть сразу сыллка на вторую в перой.

Если — False, то тогда запускаеться алгоритм поиска звена в цепи между первой и второй страницей. Запускаеться цикл **for-2**, который являеться вложеным в основной цикл **for-1**.

Далее цикл **for-2**, иттерирует перменную *link* принимающию в значение строку, которая являеться названием страницы из массива *A_links,* далее производиться условие с помощью функции *is_page_valid(link),* которая проверяет существует ли такая страница, и возвращает значение true/false. Если страница не существует то цикл **for-2**, переходит на новую иттерацию. Если же существует, то объявляются две временных переменных, *tmp_page*, которая принимает значение wikipedia.page(link) и *tmp_links,* равная tmp_page.links . Далее с помощью условия (B in tmp_links), которое проверяет если ли среди сыллок промежуточного звена, сыллка на вторую страницу, если — True, то цепочка найдена, и промежуточная страница и вторая страница записываются в массив *ans*. Так же с помощью опертаора break, цикл **for-2**, завершается и основной цикл **for-1**, переходит на след. иттерацию. *Если — False,* то цикл **for-2**, переходит на другую иттерацию.

С помощью оператора *return* функцией *links_chain* будет возвращен массив *ans.*

Которое будет передано в функцию links_chain(pages), где будет выведен список, в котором храниться цепочка страниц.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Примеры полных цепочек:

Айсберг → Буран → IBM

Владимир Путин → 1952 год → 10 августа → 1267 год → 1190 год

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Айсберг, Вода, zz	no results	Программа работает правильно
2.	Айсберг, IBM, ru	115 IBM ['Айсберг', 'Буран', 'IBM']	Программа работает правильно
3.	Владимир Путин, 10 августа, 1190 год, ru	309 Путин, Владимир Владимирович ['Владимир Путин', '1952 год', '10 августа', '1267 год', '1190 год']	Программа работает правильно
4.	Владимир Путин, 1952 год, 1267 год, ru	309 Путин, Владимир Владимирович ['Владимир Путин', '1952 год', '10 августа', '1267 год']	Программа работает правильно
5.	Владимир Путин, 1952 год, 10 августа, ru	309 Путин, Владимир Владимирович ['Владимир Путин', '1952 год',	Программа работает правильно

		'10 августа']	

Выводы.

Были изучены основные управляющие конструкции Python и модуля Wikipedia API

Была написана программа, которая считывает данные с помощью функции input() и метода .split() и выводит результат с помощью функции print().

Первая подзадача была решена с помощью функции set_lang(lang).

Вторая подзадача была решена с помощью функции max_outline(pages_names), в которой с помощью цикла for и проверки условия if находится название страницы с самым длинным описанием.

Третья подзадача была решена с помощью функции list_chain(pages_names), которая с помощью циклов for и проверки условий if else проверяла, есть ли на одной странице ссылки на следующую (или есть, но через промежуточную страницу) и создавала список-цепочку этих страниц с помощью метода .append().

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 max_words(pages):
    maximum = [0,"name"]
    for i in range(0,len(pages)):
        name = pages[i]
        page = wikipedia.page(name)
        count = len((page.summary).split())
        if maximum[0] <= count:
            maximum[0] = count
            maximum[1] = page.title
    return maximum

def links_chain(pages):
     ans = []
     ans.append(pages[0])
     for i in range(0,len(pages)-1):
          A = wikipedia.page(pages[i])
          B = pages[i+1]
          A_links = A.links

          if (B in A_links):
               ans.append(pages[i+1])
          else:
               for link in A_links:
                    if is_page_valid(link):
                         tmp_page = wikipedia.page(link)
                         tmp_links = tmp_page.links
                         if (B in tmp_links):

ans.append(A_links[A_links.index(link)])
                                   ans.append(pages[i+1])
                                   break
                         else:
                                   continue
                    else:
                         continue
     return ans


n = input().split(', ')
pages = n.copy()
pages.pop()
```

```
lang = n[-1]

if (lang in wikipedia.languages()):
    wikipedia.set_lang(lang)
    print((max_words(pages))[0],(max_words(pages))[1])
    print(links_chain(pages))
else:
    print("no results")
```