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 _____ Осинкин Е. А.

Преподаватель _____ Шевская Н. В.

Санкт-Петербург

2020

Цель работы.

Изучение базовых конструкций языка Python и модуля Wikipedia.

Задание.

Написать 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 и 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, вывести последнее.

3. Строит список-цепочку из страниц и выводит полученный список на экран. Элементы списка-цепочки — это страницы из входных данных, между которыми может быть одна промежуточная страница или не быть промежуточных страниц.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ.

В данной работе были использованы такие конструкции языка Python как:

- Встроенные функции:
  - *print()* — выводит принимаемые значения на консоль;
  - *input()* — считывает входные данные, возвращает строку;
  - *len()* — принимает строку или список, возвращает целочисленное значение – длину входного объекта;
  - *range()* — генерирует ряд чисел в заданном диапазоне с определённым шагом;

- Функции модуля Wikipedia:
  - *page(title)* – возвращает объект класса WikipediaPage, который представляет 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;
- Операторы:
  - *if: else:* — 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:
  - *for <переменная> in <итерируемый объект>:* — для каждого значения переменной, находящегося в итерируемом объекте, выполняет блок кода с одинаковым уровнем отступа после двоеточия;
- Пользовательские функции:
  - *def <название функции>(<принимаемые параметры>):* — при вызове в тексте программы по названию функции выполняется блок кода, находящийся после двоеточия в определении функции, используя принимаемые параметры.
- Методы

- *str.split()* — метод класса str, принимает на вход разделитель - один или несколько символов (по умолчанию – пробел), разбивает строку, к которой применён, на подстроки по разделителю и возвращает список этих подстрок;
- *list.append()* — добавляет в конец списка list элемент из круглых скобок.

- Обращения к полям
  - *page.summary* — поле класса *page* модуля *Wikipedia*, возвращает многострочный 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 необходимо импортировать модуль wikipedia строкой *import wikipedia.*

Для решения поставленных задач необходимо сначала считать входные данные. Для этого используется переменная *list_input,* в которую при помощи функции *input()* и метода *split(', ')* записывает список, состоящий из подстрок входной строки, разделённой по запятой с пробелом.

1. Выполнение первой подзадачи.

Для выполнения первой подзадачи воспользуемся операторами *if: else:* и *in,* и функциями модуля Wikipedia: *languages()* и *set_lang(lang).* Проверяем поддерживает ли выбранный язык Wikipedia или нет при помощи конструкции if *list_input[-1] in wikipedia.languages():* Если да, то устанавливаем данный язык запросов с помощью функции *wikipedia.set_lang(list_input[-1]),* удаляем последний последний элемент массива с помощью функции *list_input[-1].pop(),* он нам больше не понадобится и продолжаем выполнение программы дальше. Если нет, то пишем при помощи функции *printf()* текст no results и завершаем программу.

2. Выполнение второй подзадачи.

Далее для реализации второй подзадачи используется пользовательская функция *max_page_summary(),* принимающая список введённых названий страниц, в которой существуют переменные:

- *max_page* – предназначена для хранения сроки – названия страницы с максимальный количеством слов в кратком содержании.
- *max* – предназначена для хранения целого числа – количества слов в самом длинном кратком содержании страницы;

В цикле *for*, количество итераций которого равно количеству введённых названий страниц, в каждой итерации оператором *if* с помощью функции *len* проверяется количество слов в кратком содержании очередной страницы, если оно больше текущего значения переменной *max,* происходит запись этого количества в переменную *max* и происходит запись нового названия страницы с наибольшим количеством слов в переменную *max_page*. Функция возвращает кортеж из двух элементов: *max_page, max.* Таким образом выполняется вторая подзадача программы и функцией *print(max_summary[1], max_summary[0])* выводится результат.

3. Выполнение третьей подзадачи.

Решением третьей подзадачи является значение возвращенной функцией *chain_links(list_input).*

В качестве аргумента принимает массив строк, которые являются названиями страниц. Так же объявляется массив *result*, который принимает значение названия первой страницы в цепочке, то есть *list_input[0]*.

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

Все описанное ниже будет являться частью тела основного цикла, поэтому чтобы не запутаться основной цикл будет обозначаться как **for-1**, а вложенный **for-2**. Основной цикл **for-1** начинает каждую итерацию, с объявления переменной *list_links* с помощью функций *wikipedia.page(list_input[i]).links*. Эта переменная является списком ссылок, которые находятся на i-ной страницы.

Далее после объявления переменных начинается основная работа функции. В первую очередь после объявления переменных проверяется, есть ли прямая ссылка из первой страницы во вторую, с помощью *if*, проверяется условие *list_input[i + 1] in list_links*.

Если — True, то тогда в массив *result,* добавляется название второй страницы. То есть между первой и второй страницей, нет промежуточного звена в виде другой страницы, а есть сразу ссылка на вторую в первой.

Если — False, то тогда запускается алгоритм поиска звена в цепи между первой и второй страницей. Запускается цикл **for-2**, который является вложенным в основной цикл **for-1**.

Далее цикл **for-2**, итерирует переменную *link*, которая принимает в значение строку, которая является названием страницы из массива *list_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 итерацию. Если же существует, то объявляется промежуточная переменная *list_links_next*, которая равна *wikipedia.page(link).links*. Далее с помощью условия *list_input[i + 1] in list_links_next*, которое проверяет если ли среди ссылок промежуточного звена, ссылка на вторую страницу, если — True, то цепочка найдена, и промежуточная страница и вторая страница записываются в массив *result.* Так же с помощью оператора break, цикл **for-2**, завершается и основной цикл **for-1**, переходит на следующую итерацию. *Если — False,* то цикл **for-2**, переходит на другую итерацию.

С помощью оператора *return* функцией *chain_links* будет возвращен массив *result.*

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']	Программа работает правильно
2.	Айсберг, IBM, qweert	no results	Программа работает правильно

Выводы.

В ходе работы были изучены основные управляющие конструкции языка Python и модуль wikipedia.

Разработана программа, считывающая с помощью функции *input()* и метода *split()* входные данные.

Первая подзадача программы реализована в основной функции программы с помощью ветвления.

Вторая подзадача реализована в функции *max_page_summary()* с помощью алгоритма поиска максимума в цикле *for*. Вывод данных производится функцией *print().*

Третья подзадача реализована функцией *chain_links()*, в ней при помощи списка *result* и двух циклов *for,* в которых производится нахождение минимальной цепочки ссылок и записывается список этих ссылок в переменную *result.* Вывод данных производится функцией *print().*

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_page_summary(list_input):
     max_page = ''
     max = 0
     for i in range(0, len(list_input)):
          if max <=
len(wikipedia.page(list_input[i]).summary.split()):
               max_page = wikipedia.page(list_input[i]).title
               max =
len(wikipedia.page(list_input[i]).summary.split())
     return max_page, max

def chain_links(list_input):
     result = []
     result.append(list_input[0])
     for i in range(0, len(list_input) - 1):
          list_links = wikipedia.page(list_input[i]).links
          if list_input[i + 1] in list_links:
               result.append(list_input[i + 1])
          else:
               for link in list_links:
                    if is_page_valid(link):
                         list_links_next =
wikipedia.page(link).links
                         if (list_input[i + 1] in
list_links_next):

     result.append(list_links[list_links.index(link)])
                              result.append(list_input[i + 1])
                              break
                         else:
                              continue
                    else:
                         continue
     return result

list_input = input().split(', ')
if list_input[-1] in wikipedia.languages():
     wikipedia.set_lang(list_input[-1])
     list_input.pop(-1)
     print(max_page_summary(list_input)[1],
max_page_summary(list_input)[0])
     print(chain_links(list_input))
else:
     print('no results')
```