**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**](http://e.moevm.info/mod/quiz/view.php?id=76)

|  |  |  |
| --- | --- | --- |
| Студент гр. 0383 |  | Шквиря Е.В. |
| Преподаватель |  | Шевская Н.В. |

Санкт-Петербург

2020

## Цель работы.

Изучить устройство основных управляющих конструкций языка Python и научиться работать с 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.

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.

## Выполнение работы.

Порядок выполнения поставленной задачи программой:

1. Считывание строки с названиями страниц и его последующее разделение на элементы списка.

2. Проверка существования языка путём его поиска в списке всевозможных. В случае его отсутствия - вывод «no results» и завершение программы, иначе данный язык устанавливается как «язык запросов».

3. Подсчёт максимального количества слов в кратком содержании каждой страницы осуществляется путём получения каждого содержания. После получения строки в ней заменяются символы \n и \t на пробелы, чтобы корректно обрабатывать тексты из нескольких абзацев и с символами табуляции. В дальнейшем из строки делается список и подсчитывается количество элементов в нём. Так происходит подсчёт слов и находится искомая страница.

4. Цепочка ищется путём поиска следующей страницы в списке ссылок прошлой. Такой подход гарантирует нам, что мы не будем использовать лишние промежуточные звенья, значит наша цепочка будет кратчайшей. В случае отсутствия следующей страницы в списке ссылок, поиск будет производиться в списке ссылок каждой страницы из списка ссылкок прошлой страницы.

Используемые переменные:

array — список страниц, поступающих из потока ввода.

lang — язык, поступающий от пользователя для проверки и дальнейшей установки.

string — строка, у которой подсчитывается количество слов.

page\_array — список страниц, введённых пользователем, который подаётся как аргумент функции.

maxCnt — переменная, которая хранит в себе максимальное количество слов в кратком содержании страницы.

ansPage — название страницы, в которой было обнаружено максимальное количество слов.

page — название страницы из page\_array.

cntInStr — количество слов в строке.

i — индекс элемента, для которого ищется следующий элемент цепочки.

linksArray — список ссылок прошлой страницы.

answerArray — вычисленная цепочка.

Используемые функции:

is\_page\_valid — проверка существования страницы.

is\_right\_lang — проверка существования языка.

cnt\_word — количество слов в строке.

max\_cnt\_words\_summary — поиск страницы с максимальным количеством слов и его вычисление.

find\_support\_chain — поиск промежуточного звена.

find\_chain — поиск цепочки.

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 |
|  | Айсберг, IBM, ru | 115 IBM  ['Айсберг', 'Буран', 'IBM'] | Верный ответ |
|  | Лукерьино (Омская область), Васютино (Омская область), Культжугут, Озёрка (Омская область), ru | 14 Озёрка (Омская область)  ['Лукерьино (Омская область)', 'Васютино (Омская область)', 'Культжугут', 'Озёрка (Омская область)'] | Верный ответ |
|  | Candy, Атеизм, ru | 228 Атеизм  ['Candy', '2010 год', 'Атеизм'] | Верный ответ |
| 4. | 2012 BX34, IGR J17329-2731, en | 139 2012 BX34  ['2012 BX34', '2012 TC4', 'IGR J17329-2731'] | Верный ответ |
| 5. | 2012 BX34, IGR J17329-2731, en-ru | no results | Верный ответ |

## Выводы.

Были изучены основные управляющие конструкции языка Python и способы взаимодействия с Wikipedia API.

Разработана программа, считывающая от пользователя список названий страниц и язык для запросов, а также позволяющая найти максимальное количество слов в странице из списка и построить с помощью названий страниц кратчайшую цепочку с добавлением промежуточных звеньев. Для реализации использовались циклы, условные операторы if-else, функции и подключение модуля wikipedia.

# Приложение А Исходный код программы

Название файла: lab1.py

import wikipedia  
  
  
# существует ли такая страница  
def is\_page\_valid(page):  
 try:  
 wikipedia.page(page)  
 except Exception:  
 return False  
 return True  
  
  
# проверка существования языка  
def is\_right\_lang(lang):  
 return lang in wikipedia.languages()  
  
  
# количество слов в строке  
def cnt\_word(string):  
 return len(string.replace('\n', ' ').replace('\t', ' ').split(' '))  
  
  
# максимальное количество слов в строке  
def max\_cnt\_words\_summary(page\_array):  
 maxCnt = 0  
 ansPage = ""  
 for page in page\_array:  
 cntInStr = cnt\_word(wikipedia.page(page).summary)  
 if cntInStr >= maxCnt:  
 maxCnt = cntInStr  
 ansPage = page  
 return [maxCnt, wikipedia.page(ansPage).title]  
  
  
# поиск промежуточного звена  
def find\_support\_chain(page\_array, i):  
 linksArray = wikipedia.page(page\_array[i]).links  
 for item in linksArray:  
 if not is\_page\_valid(item):  
 continue  
 if page\_array[i + 1] in wikipedia.page(item).links:  
 return item  
  
  
# поиск цепочки  
def find\_chain(page\_array):  
 answerArray = [page\_array[0]]  
 for i in range(0, len(page\_array) - 1):  
 if page\_array[i + 1] not in wikipedia.page(page\_array[i]).links:  
 answerArray.append(find\_support\_chain(page\_array, i))  
 answerArray.append(page\_array[i + 1])  
 return answerArray  
  
  
# точка начала программы  
array = input().split(', ')  
if is\_right\_lang(array[-1]):  
 wikipedia.set\_lang(array.pop())  
else:  
 print("no results")  
 quit(0)  
print(\*max\_cnt\_words\_summary(array))  
print(find\_chain(array))