**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Колованов Р.А. |
| Преподаватель |  | Размочаева Н.В. |

Санкт-Петербург

2019

## Цель работы.

Изучение основ языка Python и получение навыков работы с основными управляющими конструкциями, типами данных, функциями.

## 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.

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.

## Выполнение работы.

Для начала считываем входные данные, заносим их в строку, далее при помощи метода ***split()*** разделяем строку на ключевые слова. Полученный список запишем в переменную ***data***. Все элементы списка кроме последнего — названия страниц из Википедии. Занесем их в переменную ***titles***, а последний элемент (язык для поисковых запросов) запишем в переменную ***lang***. Далее при помощи конструкции ***if ... in ...*** проверяем, что данный язык есть в словаре поддерживаемых языков библиотеки ***wikipedia***. Если язык не поддерживается, то выведем сообщение «no results» и завершим выполнение программы при помощи ***exit()***. Иначе установим данный язык как язык поисковых запросов при помощи метода ***set\_lang()***.

Далее мы создаем переменные ***max\_words\_count*** и ***max\_words\_title***, в которых мы будем хранить максимальное количество слов в странице и название этой страницы соответственно. Пробегаемся по списку ***titles*** при помощи цикла ***for*** и находим количество слов при помощи поля ***summary*** и метода ***split()*** для каждой страницы. Название страницы с максимальным количеством слов запишем в переменную ***max\_words\_title***, а количество слов в ней — в переменную ***max\_words\_count***.

Далее найдем ответ на третью подзадачу. Для этого создаем список ***path***, который будет хранить цепочку из страниц. Далее начинаем последовательно связывать страницы в цикле ***for***, для этого используем поле ***links***. Если в списке ***links*** есть ссылка на следующий элемент списка ***titles***, то мы можем связать два элемента списка ***titles*** без промежуточных страниц напрямую, тогда запишем следующий элемент списка ***titles*** в список ***path***. Если мы не нашли ссылку, то пробегаемся по ссылкам текущей страницы, обращаемся по ним к другим страницам и уже производим поиск среди ссылок этих страниц. При нахождении нужной связи записываем промежуточную страницу и следующее звено списка ***titles*** в список ***path***.

В конце выведем ответы на подзадачи.

Функция ***get\_wikipedia\_page()*** предназначена для проверки существования страницы с таким названием при помощи обработки возникающих исключений. Функция вернет объект класса page, если страница существует, в другом случае вернет ***None***.

Функция ***get\_max\_word\_page\_and\_count()*** возвращает ответ на 3 подзадачу, реализация которой описана выше.

Функция ***get\_links\_path()*** возвращает ответ на 2 подзадачу, реализация которой описана выше.

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'] |  |

## Выводы.

Были изучены основ языка Python; получены навыки работы с основными управляющими конструкциями, типами данных, функциями. Разработана программа, выполняющая считывание с клавиатуры исходных данных. Для обработки команд пользователя использовались условные операторы *if-else*. Для решения поставленной задачи использовалась библиотека wikipedia. Для отлавливания исключительных ситуаций при попытке поиска несуществующей страницы был использован блок *try-except*.

# Приложение А Исходный код программы

Название файла: main.py

import wikipedia

def get\_wikipedia\_page(title):

try:

page = wikipedia.page(title)

except Exception:

return None

return page

def get\_max\_word\_page\_and\_count(titles):

max\_words\_count = -1

max\_words\_title = ""

for title in titles:

page = get\_wikipedia\_page(title)

if page == None:

continue

count = len(page.summary.split())

if count >= max\_words\_count:

max\_words\_count = count

max\_words\_title = page.title

return (max\_words\_count, max\_words\_title)

def get\_links\_path(titles):

path = [titles[0]]

for i in range(len(titles) - 1):

page = get\_wikipedia\_page(titles[i])

if page == None:

continue

links = page.links

if titles[i + 1] in links:

path.append(titles[i + 1])

else:

for link in links:

link\_page = get\_wikipedia\_page(link)

if link\_page == None:

continue

if titles[i + 1] in link\_page.links:

path.append(link)

path.append(titles[i + 1])

break

return path

data = input().split(", ")

titles = data[0:-1]

lang = data[-1]

if lang in wikipedia.languages().keys():

wikipedia.set\_lang(lang)

max\_word\_info = get\_max\_word\_page\_and\_count(titles)

print(max\_word\_info[0], max\_word\_info[1])

print(get\_links\_path(titles))

else:

print("no results")