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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Введение в информационные технологии»

Тема: Основные управляющие конструкции

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучить основные управляющие конструкции языка *Python.*

Задание.

Используя библиотеку *wikipedia*, 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.

Были подключены следующие модули, использованные в программе: *wikipedia*, используемая как *wk*, и *help_wiki_function* (из неё был выбран модуль *is_page_valid*)

Программа была поделена на следующие подзадачи:

1. Импортируем модуль *wikipedia* и *help_wiki_function.*
2. В переменную *titles* записываем входные данные, которые пользователь вводит с клавиатуры. В качестве типа переменной выберем список, чтобы была возможность выбирать каждую страницу через итерацию в цикле. Записываем в переменную *lang*, убирая последний элемент из списка. В неё записывается язык.
3. Вызываем функцию *is_lang_valid(titles)*, чтобы проверить, есть ли данный язык в возможных языках сервиса *Wikipedia*.
4. Функция *is_lang_valid(titles)* проверяет, есть ли данный язык в возможных языках сервиса *Wikipedia*. Если есть, то функция устанавливает его как язык запросов в текущей программе и возвращает *True*. В ином случае, она возвращает *False*.
5. В зависимости от возвращаемого значения программа выводит *«no results»* (если *False*) или устанавливает язык и печатает значения, возвращаемые функциями max_words_in_page(titles) и find_links(titles) (если *True*).

6. Функция *max_words_in_page(titles)*, проходясь по списку названий страниц, ищет их краткое описание и считает в нем количество слов. Возвращает максимальное количество слов в кратком содержании страницы и ее имя.
7. Функция *find_links(titles)* возвращает список-цепочку. Элементы списка - это страницы, между которыми может быть одна промежуточная страница или не быть промежуточных страниц. Возвращает список-цепочку

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']	
2.	Айсберг, IBM, 012	no results	Нет доступного языка '012'
3.	Канада, Валюта ячё, Тенке, cv	32 Литас ['Канада', 'Валюта ячё', 'ISO 4217', 'Тенке']	

Выводы.

Были изучены основные управляющие конструкции языка *Python*. Были использованы условные операторы *if-else* и циклы *for*. 4 Была разработана программа, выполняющая считывание с клавиатуры исходных данных и обрабатывающая их согласно условию. Для обработки данных использовались условные операторы *if-else* и циклы *for*. При выполнении работы были использованы и изучены модули *wikipedia* и *help_wiki_function*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

Код:

```
import wikipedia as wk
from help_wiki_function import is_page_valid


def is_lang_valid(language):
    if language in wk.languages():
        return True
    else:
        return False


def max_words_in_page(title_of_page):
    words = []
    title = ""
    count = 0
    for i in range(len(title_of_page)):
        words = wk.page(title_of_page[i]).summary.split()
        if len(words) >= count:
            count = len(words)
            title = wk.page(title_of_page[i]).title
    arguments = [count, title]
    return arguments


def find_links(title_of_page):
```

```
    chain = [title_of_page[0]]
    for i in range(len(title_of_page) - 1):
        if title_of_page[i + 1] in wk.page(title_of_page[i]).links:
            chain.append(title_of_page[i + 1])
        else:
            for x in wk.page(title_of_page[i]).links:
                if is_page_valid(x):
                    if title_of_page[i + 1] in wk.page(x).links:
                        chain.append(x)
                        chain.append(title_of_page[i + 1])
                        break
    return chain


titles = input().split(", ")
lang = titles.pop()
if is_lang_valid(lang):
    wk.set_lang(lang)
    args = max_words_in_page(titles)
    print(args[0], args[1])
    print(find_links(titles))
else:
    print("no results")
```