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.

Студент гр. 0382 _____ Крючков А.М.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Изучить основные управляющие конструкции языка Python, научиться работать с модулем Wikipedia API.

Задание.

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.

Функции модуля Wikipedia API:

- *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, а значениями – названия.
- *set_lang(lang)* - устанавливает язык lang, как язык запросов в текущей программе.

Атрибуты класса WikipediaPage:

- *page.summary* – поле класса page модуля Wikipedia, которое возвращает строку, содержащую краткое содержание страницы page.
- *page.title* – поле класса page модуля Wikipedia, которое возвращает строку, содержащую краткое содержание страницы page.
- *page.links* – поле класса page модуля Wikipedia, которое возвращает список названий страниц, ссылка на которые содержит страница page.

Выполнение работы.

Решение происходило на базе ОС Linux Ubuntu 20.04 в среде разработке Visual Studio Code на языке python.

Сначала ,импортиует модуль *wikipedia*, далее считываются данные, введённые пользователем, затем, при прямом запуске программы, происходит выполнение функций и вывод возвращаемых ими значений

Inlet - массив, введенных через «, » значений.

Пользовательские функции.

0. *is_page_valid(name)* — проверяет существование страницы с названием *name.*

1. *is_language_available(lang) (lang* — язык, который надо установить) — Проверяет, есть ли такой язык в возможных языках сервиса, обращаясь к словарю *dict_of_languages*, если нет, выводит строку "no results" и завершает выполнение программы. В случае, если язык есть, устанавливает его как язык запросов в текущей программе, используя функцию *wikipedia.set_lang(language).*

2. get_max_words_in_summary(pages_name) – ищет максимальное число слов, при помощи цикла *for,* в кратком содержании страниц и возвращает это максимальное количество и название страницы, у которой оно обнаружилось.

page_name_with_max_words — имя страницы с максимальным количеством слов в кратком описании.

max_words_found — количество слов у найденной страницы

3. *get_page_chain(pages_name)* – *с*троит 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.

number_of_pages — количество страниц в pages_name

page_number — номер страницы на текущей итерации(страницы, до которой построили цепочку)

page_found — найдена или нет следующая страница в данный момент.

links_current_page — ссылки тукущей страницы.

links_inter_page — ссылки промежуточной страницы(когда путь через одного)

chain — возвращаемая цепочка страниц(массив)

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
1.	Айсберг, IBM, ru	115 IBM ['Айсберг', 'Буран', 'IBM']	Программа работает правильно
2.	Ya delau cto-to ne tak	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 с помощью функции *is_language_available()*.

Вторая подзадача реализована в функции *get_max_words_in_summary()* с помощью алгоритма поиска максимума в цикле for, результат подзадачи выводится встроенной функцией *print()*.

Третья подзадача реализована функцией *get_page_chain()*, в ней при помощи вложенных циклов происходит поиск прямого и, в случае неуспеха, поиска пути через одного. Вывод данных производится функцией *print()*.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src/2.py

```
import wikipedia

#is_page_valid(page_name)
def is_page_valid(page_name):
    try:
        wikipedia.page(page_name)
    except Exception:
        return False
    return True

def is_language_available(language):
    dict_of_languages = wikipedia.languages()
    if dict_of_languages.get(language, "not found")=="not found":
        return False
    else:
        wikipedia.set_lang(language)
        return True

def get_max_words_in_summary(pages_name):
    max_words_found = -1
    page_name_with_max_words = ""
    for page_name in pages_name:
        page = wikipedia.page(page_name)
                if max_words_found <= len(page.summary.split()):
#len(page.summary.split()) - количество слов в кратком содержании страницы
                max_words_found = len(page.summary.split())
                page_name_with_max_words = page.title
    return max_words_found, page_name_with_max_words

def get_page_chain(pages_name):
    number_of_pages = len(pages_name)
    if number_of_pages == 0: return []
    chain = [pages_name[0]]
    for page_number in range (number_of_pages-1):
        page_found = False
                                            links_current_page = wikipedia.page(pages_name[page_number]).links
        page_target_name = pages_name[page_number+1]
                for link_of_current_page in links_current_page: #Кратчайший путь
                if link_of_current_page == page_target_name:
                    chain.append(page_target_name)
                    page_found = True
                    break
            if page_found: continue #Если короткий путь уже найден, то путь через одну страницу не ищется
                for link_of_current_page in links_current_page: #Путь через одну страницу
```

```
                    if not is_page_valid(link_of_current_page): continue
#Если страницы с такой ссылкой нет, то проверяем следующую ссылку
                                                        links_inter_page    =
wikipedia.page(link_of_current_page).links
                    for link_from_inter_page in links_inter_page:
                        if link_from_inter_page == page_target_name:
                            chain.append(link_of_current_page)
                            chain.append(page_target_name)
                            page_found = True
                            break
                    if page_found: break
            return chain

        if __name__ == "__main__":
            inlet = input().split(", ")
            if not is_language_available(inlet[-1]):
                print("no results")
            else:
                 print(*get_max_words_in_summary(inlet[:-1])) #inlet[:-1]
список страниц
                print(get_page_chain(inlet[:-1]))
```