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. 1382 _____ Коренев Д.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2021

Цель работы.

Изучить основные управляющие конструкции языка Python. Научится работать с модулем 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 больше ничего не делает. В случае, если язык есть, устанавливает его как язык запросов в текущей программе и выполняет еще два действия:

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 *(данный пример актуализирован к состоянию страниц wikipedia на 2021 год)*:

Айсберг, IBM, ru

В числе ссылок страницы с названием "Айсберг", есть страница с названием , которая содержит ссылку на страницу с названием "1959 год", у которой есть

ссылка на страницу с названием "IBM" -- это и есть цепочка с промежуточным звеном в виде страницы "1959 год".
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.
Пример входных данных:
Айсберг, IBM, ru
Пример вывода:
115 IBM
['Айсберг', '1959 год', 'IBM']
Первая строка содержит решение подзадачи №2, вторая - №3.
Важное уточнение: каждую подзадачу (1, 2, 3) оформите в виде отдельных функций.
Функции должны быть "чистыми". Мы с этим определением ближе познакомимся в *разделе №3 на лекциях*, на данный момент следует выполнить требования:
1. Ваши функции не должны выводить что-либо на экран (только возвращать результат)
2. Ваши функции не должны изменять глобальные переменные (те переменные, которые существуют вне функции, то есть во внешней программе)
3. Ваши функции не должны изменять и свои аргументы, которые передаются в функцию (лучше возвращать измененную копию аргумента).

Выполнение работы.

Для использования модуля Wikipedia API, его необходимо импортировать: import wikipedia as wiki для более короткого доступа к нему.

Программа имеет 4 функции:

is_page_valid — принимает название страницы и проверяет ее наличие на wipiedia. Возвращает True — если такая страница есть, и — False — если такая страница отсутствует.

is_language_valid — принимает на вход строку — название языка - и проверяет, есть ли этот язык в сервисах и возвращает True и False в противном случае.

max_word — принимает на вход массив названий страниц, и при помощи цикла for подсчитывает количество слов в кратком содержании вики страницы. Чтобы это сделать необходимо использовать функцию len(page.summary.split()), где page — страница класса wiki. Каждую итерацию проверятся, превышает ли количество слов максимальной найденных раньше, и присваивает в переменные maximum и answer_title максимальное количество слов и заголовок страницы, если количество слов больше, чем было найдено раньше. Возвращает массив, в котором на 0 индексе находится переменная maximum, а на 1 индексе - answer_title.

Chain - Строит список-цепочку из страниц и возвращает полученный список. Создается список chain в который будут добавляться новые элементы. При помощи цикла for перебираю все кроме последнего индексы поступившего на вход функции списка названий страниц. Добавляю в chain название очередной страницы, а далее создаю объект класса wiki для этой страницы. Проверяю, есть ли следующий элемент списка в ссылках нынешнего элемента, т.к. если он есть, то искать промежуточный элемент нет необходимости. Если нет, то при помощи еще одного цикла for для каждой ссылки из списка ссылок данной страницы, создаю из нее объект класса wiki и если следующий элемент во входном списке есть в ссылках этого wiki объекта, то добавляю эту ссылку в список chain. Добавляю последний элемент входного списка в список chain и возвращаю этот список.

Далее идет обработка входных данных и вызов вышеописанных функций. Считываю входные данные с помощью функции input() и разделяю при помощи метода split, получаю массив и присваиваю его переменной name_pages. При

помощи метода pop() извлекаю из списка name_pages последний элемент — язык- и присваюваю его переменной lang. Далее проверяю, существует ли такой язык при помощи функции is_language_valid, если нет — то вывожу «no results». Если такой язык есть, то устанавливаю его. Создаю массив true_pages в который буду добавлять только существующие страницы из входных данных при помощи цикла for и функции is_page_valid. Присваиваю переменной max_ans возвращаемое значение функции max_word аргументом которой был список true_pages. Вывожу элементы 0 и 1 индекса списка max_ans. Вызываю вывожу возвращаемое значение функции chain аргументом которой так же являлся список true_pages. Программа завершилась, все ответы выведены.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 — Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарий
1	Айсберг, IBM, ru	115 IBM ['Айсберг', '1959 год', 'IBM']	Страницей с самым большим количеством слов в кратком содержании является IBM, слов — 115, Промежуточная страница между страницами 'Айсберг' и 'IBM' - '1959 год'. Код работает верно.
2	Айсберг, Личность, ru	95 Личность ['Айсберг', 'Gemeinsame Normdatei', 'Личность']	Страницей с самым большим количеством слов в кратком содержании является Личность, слов — 95, Промежуточная страница между страницами 'Айсберг' и 'Личность', - 'Gemeinsame Normdatei'. Код работает верно.

3	X86-64, AMD64, ru	228 X86-64 ['X86-64', '64 бит', 'AMD64']	Страницей с самым большим количеством слов в кратком содержании является X86-64, слов — 228, Промежуточная страница между страницами 'X86-64' и 'AMD64', - '64 бит'. Код работает верно.
4	leti, hell, erq	no results	Язык erq отсутствует в списке языков сервиса, выводится no results, код работает верно

Выводы.

Были изучены основные управляющие конструкции языка Python. Написана программа, использующая модуль Wikipedia API, которая выполняет такие задачи как: проверка существования языка в возможных языках сервиса, установка языка, поиск максимального количества слов в кратком содержании страницы, построение списка-цепочки и выведение полученного на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название программы: CS_Korenev_Danil_lb1.py

```
import wikipedia as wiki

def is_page_valid(page):
    try:
        wiki.page(page)
    except Exception:
        return False
    return True

def is_language_valid(lang):
    if lang in wiki.languages():
        return True
    else:
        return False

def max_word(arr):
    maximum, answer_title = 0, ""
    for title in arr:
        page = wiki.page(title)
        count_word = len(page.summary.split())
        if count_word >= maximum:
            maximum, answer_title = count_word, page.title
    return [maximum, answer_title]

def chain(arr):
    chain = []
    for index_page in range(len(arr)-1):
        chain.append(arr[index_page])
        page = wiki.page(arr[index_page])
        if (arr[index_page+1]) not in page.links:
            for link in page.links:
                if is_page_valid(link):
                    link_page = wiki.page(link) # вики объект страницы в
ссылках
                    if arr[index_page+1] in link_page.links:
                        chain.append(link)
                        break
    chain.append(arr[-1])
    return chain

name_pages = input().split(", ")  # вход строки
lang = name_pages.pop()  # отделяем язык


if (is_language_valid(lang)):
    wiki.set_lang(lang)
    true_pages = []
    for i in name_pages:
        if is_page_valid(i):
            true_pages.append(i)
```

```
    max_ans = max_word(true_pages)
    print(max_ans[0], max_ans[1])
    print(chain(true_pages))
else:
    print("no results")
```