**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**

Студентка гр. 0382 _____ Ситченко К.С.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

**Цель работы.**

Изучить основные управляющие конструкции языка Python, а также научиться 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 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.

Пример входных данных:

Айсберг, IBM, ru

Пример вывода:

115 IBM

['Айсберг', 'Буран', 'IBM']

Первая строка содержит решение подзадачи №2, вторая - №3.

**Основные теоретические положения.**

Функции модуля Wikipedia представлены в таблице 1. Атрибуты класса WikipediaPage (страницы сервиса Wikipedia) представлены в таблице 2.

Таблица 1 – Функции модуля Wikipedia

| **Функция** | **Описание** | **Возвращаемое значение** |
|---|---|---|
| page(title) | Поиск страницы | Объект класса WikipediaPage, который представляет собой страничку сервиса Wikipedia, название которой - строка title |
| languages() | Поиск всех возможных языков сервиса | Словарь, ключами которого являются сокращенные названия языков, а значениями - названия. Например:<br><br>>>> wikipedia.languages()['ru']<br>'русский' |
| set_lang(lang) | Установить язык lang, как язык запросов в текущей программе. | None |

Таблица 2 – Атрибуты класса WikipediaPage (страницы сервиса Wikipedia)

| **Поле класса** | **Описание** | **Возвращаемое значение** |
|---|---|---|
| page.summary | Краткое содержание страницы page | Строка |
| page.title | Название страницы page | Строка |
| page.links | Список названий страниц, ссылки на которые содержит страница page | Список строк |

**Выполнение работы.**

Для решения данной задачи было написано несколько функций:

1. is_page_valid(page) – функция проверяет наличие страницы с таким именем в Wikipedia.
2. set_lang(lang) – функция проверяет существование указанного языка и, если да, то устанавливает его как язык запросов в текущей программе.
3. max_words_on_page(pages) – функция находит среди данных страницу с максимальным количеством слов в кратком содержании и выводит на экран это количество и название страницы.
4. min_chain(pages) – функция строит список-цепочку из страниц и выводит полученный список на экран.

На вход программе подается строка. С помощью метода *split()* преобразовываем эту строку в список, хранящийся в переменной *st* и последний элемент которого является сокращенной формой языка и проверяется во второй функции. Если введенной формы языка нет в Wikipedia, то программа выведет на экран “no results” и прекратит работу. Если есть, то установит его как язык запросов данной программы. Затем в переменную *inp_pages* мы помещаем срез списка до предпоследнего элемента (список названий страниц). Далее мы отправляем этот список в третью функцию, где проверяется факт существования страницы с помощью функции *is_page_valid(page)*, а также посредством функций *len()* и *page(title)*

ищется максимальное количество слов в кратком содержании страницы и ее название. С помощью вложенного цикла в функции *min_chain(pages)* построим кратчайшую список-цепочку из страниц.

Разработанный программный код см. в приложении А.

**Тестирование.**

Результаты тестирования представлены в таблице 3.

Таблица 3 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1 | Айсберг, IBM, ru | 115 IBM<br>['Айсберг', 'Буран', 'IBM'] | Программа работает верно |
| 2 | Seoul, ocean, enl | no results | Программа работает верно |
| 3 | 불교, 탄트라, ko | 198 불교<br>['불교', '탄트라'] | Программа работает верно |

**Выводы.**

Были изучены основные управляющие конструкции языка Python, а также проведена работа с модулем Wikipedia API.

Разработана программа, которая считывала строку, преобразовывала ее в список, а затем отправляла ее ф функции. Для работы подключался модуль Wikipedia API и использовались его функции.

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

1. Название файла: main.py

```
import wikipedia

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

def set_lang(lang):
  if lang in wikipedia.languages():
    wikipedia.set_lang(lang)
    return True
  else:
    return False

def max_words_on_page(pages):
  max_page = ''
  max_words = 0
  for i in range(len(pages)):
    if is_page_valid(pages[i]) == True:
      page = wikipedia.page(pages[i])
      if len((page.summary).split()) >= max_words:
        max_words = len((page.summary).split())
        max_page = page.title
  return max_words, max_page

def min_chain(pages):
  arr = []
  arr.append(pages[0])
  for i in range(len(pages)-1):
    if pages[i+1] in wikipedia.page(pages[i]).links:
      arr.append(pages[i+1])
    else:
      for j in wikipedia.page(pages[i]).links:
        if pages[i+1] in wikipedia.page(j).links:
          arr.append(j)
          arr.append(pages[i+1])
          break
  return arr

st = input().split(', ')
if set_lang(st[-1]):
  inp_pages = st[:len(st) - 1]
  print(max_words_on_page(inp_pages)[0],
max_words_on_page(inp_pages)[1])
  print(min_chain(inp_pages))
else:
  print("no results")
```