

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра САПР

ЗВІТ

до лабораторної роботи № 8

на тему:

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ
ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.

СТРУКТУРНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON (частина 2)

з дисципліни “Комп’ютерна лінгвістика”

Виконала:

Студентка групи ПРЛм-12

Рибчак Х. В.

Перевірив:

Асистент кафедри САПР

Дупак Б. П.

Львів 2015

МЕТА РОБОТИ

Вивчення основ програмування на мові Python. Вивчення основ структурного програмування мовою Python. Повторення та закріплення знань отриманих при виконанні попередніх лабораторних робіт. Покращення загальних навичок у програмуванні.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1 Функція, як аргумент

Аргументи функцій, які розглядалися в попередніх лабораторних роботах, були простими об'єктами, такими як стрічка, або структурованими, такими як список. В Python аргументом функції також може бути і інша функція. Python підтримує ще один спосіб визначення функцій як аргументів іншої функції, це так званий лямбда-вираз (анонімна функція).

2 Функції накопичення

Виконання функцій накопичення починається з виділення певного об'єму пам'яті. Під час здійснення ітерацій над вхідними даними, цей об'єм заповнюється і тільки після цього функція повертає результат (велику структуру або узагальнений результат). Стандартний спосіб реалізувати таку функцію це створити пустий список, в списку накопичити дані і повернути цей список.

3 Функції вищого рівня

Python підтримує деякі функції вищого порядку, які є стандартними для мов функціонального програмування, таких як Haskell. Розглянемо ці функції разом з еквівалентними виразами, які використовують list comprehensions.

4 Зазначені (поіменовані) аргументи.

У випадку коли функція має багато параметрів то досить легко заплутатись в їх порядку при виклику функції. В мові Python реалізовано

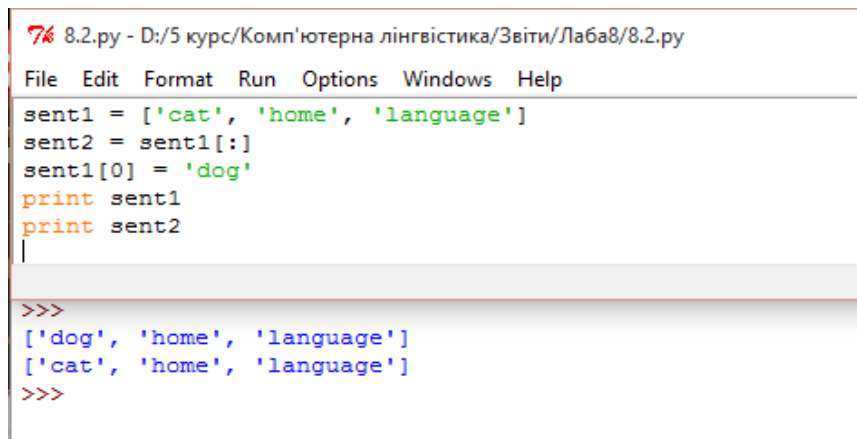
можливість явно визначати відповідність між значеннями і іменами аргументів при виклику функції. Ключі дозволяють встановити відповідність за іменами а не за позиціями. До параметрів можна звертатися за іменами (ключами) і присвоювати їм значення по замовчуванню. Застосовуючи такий підхід параметри функції можуть бути записані в довільному порядку, а також і опущені у випадку визначення їх значень по замовчуванню.

ТЕКСТИ ПРОГРАМ НА МОВІ PYTHON

ВАРІАНТ №8

2. Створити список слів і зберегти їх в змінній `sent1`. Здійснити операцію присвоювання `sent2 = sent1[:]`. Змінити один з елементів в `sent1` і перевірити чи змінився `sent2`. Результат письмово пояснити.

`Sent2` — копія списку `sent1`, тому заміна елемента відбувається лише у першому випадку.

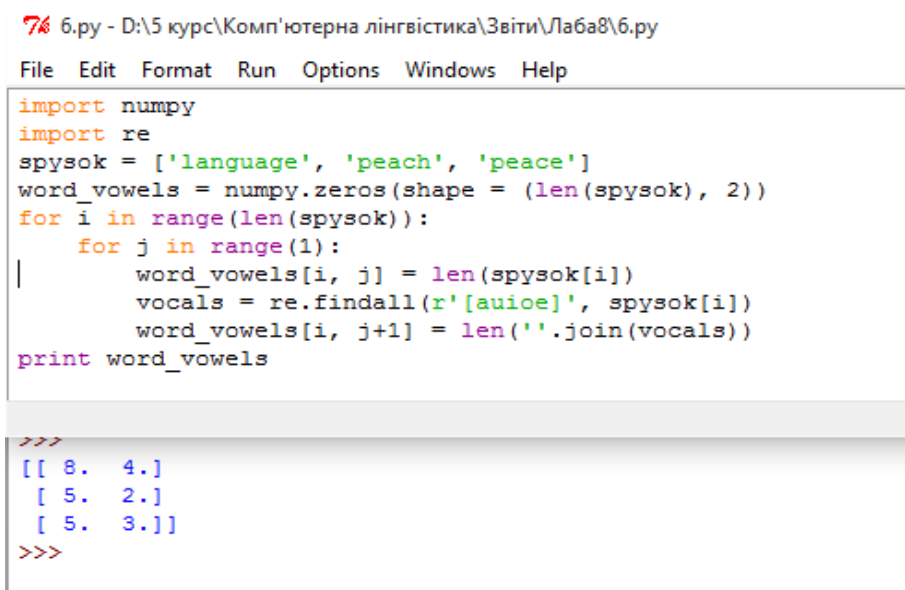


```
7% 8.2.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба8/8.2.py
File Edit Format Run Options Windows Help
sent1 = ['cat', 'home', 'language']
sent2 = sent1[:]
sent1[0] = 'dog'
print sent1
print sent2
|

>>>
['dog', 'home', 'language']
['cat', 'home', 'language']
>>>
```

Рис. 1. Текст програми №2.

6. Написати програму для створення двовимірного масиву `word_vowels` елементами якого є набори. Програма повинна обробити список слів і додати результати обробки до `word_vowels[l][v]` де `l` — довжина слова, `v` — кількість голосних у слові.



```
7% 6.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба8/6.py
File Edit Format Run Options Windows Help
import numpy
import re
spysok = ['language', 'peach', 'peace']
word_vowels = numpy.zeros(shape = (len(spysok), 2))
for i in range(len(spysok)):
    for j in range(1):
        word_vowels[i, j] = len(spysok[i])
        vocals = re.findall(r'[auioe]', spysok[i])
        word_vowels[i, j+1] = len(''.join(vocals))
print word_vowels

>>>
[[ 8.  4.]
 [ 5.  2.]
 [ 5.  3.]]
>>>
```

Рис. 2. Текст програми №6.

10. Гематрія – метод виявлення прихованого змісту слів на основі порівняння чисел, які відповідають словам. Слова з однаковими числами мають однаковий зміст. Число слова визначається сумуванням чисел, як відповідають його літерам. Здійснити аналіз корпусу (наприклад nltk.corpus.state_union). Для кожного з текстів визначити скільки слів мають номер 555 та 777.

```
File Edit Format Run Options Windows Help
from nltk.corpus import state_union
import re
letter_vals = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':80, 'g':3, 'h':8,
               'i':10, 'j':10, 'k':20, 'l':30, 'm':40, 'n':50, 'o':70,
               'p':80, 'q':100, 'r':200, 's':300, 't':400, 'u':6,
               'v':6, 'w':800, 'x':60, 'y':10, 'z':7, ' ':0}
for num_text in state_union.fileids():
    text_value = []
    text = re.findall(r"[a-zA-Z']+", state_union.raw(num_text))
    text = [word.lower() for word in text]
    for word in text:
        word_value = sum(letter_vals[w] for w in word)
        text_value.append(word_value)
    num_555 = text_value.count(555)
    num_777 = text_value.count(777)
    print num_text, '    555 = ', num_555, '    ', '777 = ', num_777

>>>
1945-Truman.txt      555 = 0      777 = 0
1946-Truman.txt      555 = 11     777 = 2
1947-Truman.txt      555 = 3      777 = 3
1948-Truman.txt      555 = 1      777 = 0
1949-Truman.txt      555 = 1      777 = 1
1950-Truman.txt      555 = 1      777 = 0
1951-Truman.txt      555 = 0      777 = 0
1953-Eisenhower.txt  555 = 2      777 = 2
1954-Eisenhower.txt  555 = 0      777 = 1
1955-Eisenhower.txt  555 = 2      777 = 3
1956-Eisenhower.txt  555 = 1      777 = 1
1957-Eisenhower.txt  555 = 1      777 = 0
1958-Eisenhower.txt  555 = 2      777 = 0
1959-Eisenhower.txt  555 = 1      777 = 2
1960-Eisenhower.txt  555 = 1      777 = 0
1961-Kennedy.txt     555 = 0      777 = 0
1962-Kennedy.txt     555 = 1      777 = 1
1963-Johnson.txt     555 = 0      777 = 0
1963-Kennedy.txt     555 = 2      777 = 0
1964-Johnson.txt     555 = 0      777 = 1
```

Рис. 3. Текст програми №10.

13. Написати list comprehension для сортування списку синсетів WordNet за близькістю до заданого синсету. Наприклад, дані синсети minke_whale.n.01, orca.n.01, novel.n.01, та tortoise.n.01, потрібно їх відсортувати згідно їх path_distance() від right_whale.n.01.

```
File Edit Format Run Options Windows Help
from nltk.corpus import wordnet as wn
right_whale = wn.synset('right_whale.n.01')
synsets = ['minke_whale.n.01', 'orca.n.01', 'novel.n.01', 'tortoise.n.01']
list_sim = [right_whale.path_similarity(wn.synset(w)) for w in synsets]
final = sorted(list_sim)
slownyk = {}
for w in synsets:
    slownyk[right_whale.path_similarity(wn.synset(w))] = w
syn_final = [(slownyk[w], w) for w in final]
print syn_final
|
Ln: 11 C
>>>
[('novel.n.01', 0.043478260869565216), ('tortoise.n.01', 0.07692307692307693),
('orca.n.01', 0.16666666666666666), ('minke_whale.n.01', 0.25)]
>>>
```

Рис. 4. Текст програми №13.

16. Імпортувати функцію itemgetter() модуля operator зі стандартної бібліотеки Python (from operator import itemgetter). Створити список words , який містить декілька слів. Спробувати виконати: sorted(words, key=itemgetter(1)), та sorted(words, key=itemgetter(-1)). Пояснити письмово роботу функції itemgetter().

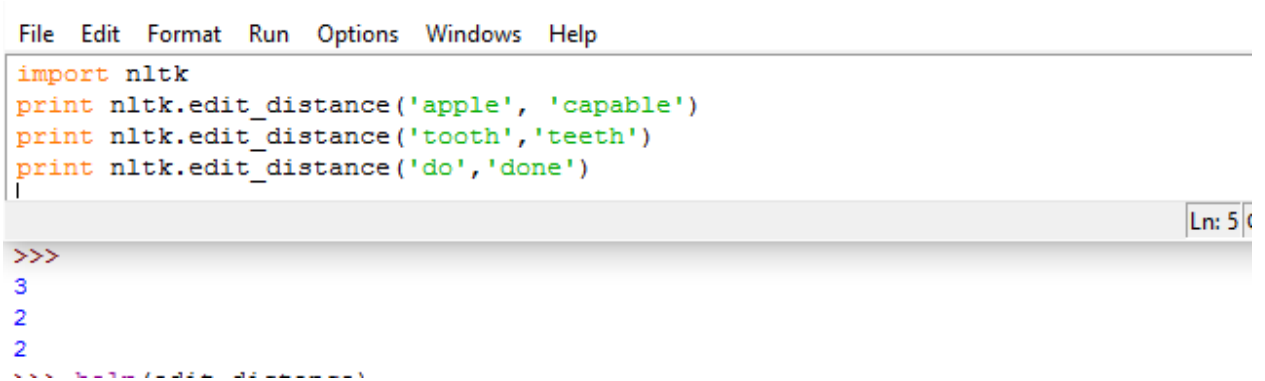
Функції itemgetter() повертає позицію, за якою треба здійснити сортування. -1 — сортування за останньою буквою, 1 — сортування за другою буквою, 0 — сортування за першою буквою.

```
File Edit Format Run Options Windows Help
from operator import itemgetter
words = ['apple', 'peach', 'plum', 'orange']
print 'Result of "itemgetter(0)"      ', sorted(words, key = itemgetter(0))
print 'Result of "itemgetter(1)"      ', sorted(words, key = itemgetter(1))
print 'Result of "itemgetter(-1)"     ', sorted(words, key = itemgetter(-1))
|
Ln: 6
>>>
Result of "itemgetter(0)"      ['apple', 'orange', 'peach', 'plum']
Result of "itemgetter(1)"      ['peach', 'plum', 'apple', 'orange']
Result of "itemgetter(-1)"     ['apple', 'orange', 'peach', 'plum']
>>>
```

Рис. 5. Текст програми №16.

17. В NLTK реалізовано алгоритм Левінштейна для порівняння стрічок. Спробуйте скористатись цим модулем `nltk.edit_dist()`. Яким чином в цьому модулі використовується динамічне програмування? Який підхід використовується знизу-вверх чи зверху-вниз?

Динамічне програмування полягає у тому, що на кожному кроці треба робити вибір, що є для нас найвигіднішим (у даному випадку, вибирати найменше значення).



```
File Edit Format Run Options Windows Help
import nltk
print nltk.edit_distance('apple', 'capable')
print nltk.edit_distance('tooth', 'teeth')
print nltk.edit_distance('do', 'done')
|
Ln: 5

>>>
3
2
2
... 
```

Рис. 6. Текст програми №17.

ВИСНОВОК

У цій лабораторній роботі я вивчила основи структурного програмування мовою Python. Повторила та закріпила знання, отримані при виконанні попередніх лабораторних робіт. Покращила загальні навички у програмуванні.