**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**іНСТИТУТ КОМП’ютерних НАУК та ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

### Кафедра “Системи автоматизованого проектування”



Звіт до лабораторної роботи №2

З дисципліни «Комп’ютерна лінгвістика»

на тему

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

ОСНОВИ ПРОГРАМУВАННЯ НА МОВІ PYTHON (частина 1)»

Виконала

Ст.гр.ПРЛм-12

Гречух Ю.

Перевірив

Старший викладач кафедри САПР

Дупак П.

Львів-2015

**Мета роботи -** вивчення основ програмування на мові *Python,* ознайомлення з контрольними структурами та класом FreqDist.

**Теоретичні відомості**

1. ***Умовні оператори***

Python підтримує широкий набір операторів для встановлення взаємозв’язків між змінними (значеннями). Це такі оператори як «<», «>», «!=» тощо. Звичайно ми використовуємо умовні оператори, як частину *If* операторів. Для перевірки властивостей окремих слів існує набір функцій, до якого входять наприклад s.startswith(t), s.endswith(t)тощо.

Можна ставити і складніші умови. Якщо С це умова, то *not C* це також умова. Якщо є дві умови С1 та С2, то побудувати нову умову використовуючи оператори диз’юнкції та кон’юнкції: *c1 and c2, c1 or c2.*

Цілі, стрічки і списки є типами даних в *Python*. Кожне значення (змінна) має свій тип. Цей тип визначає, які операції ви можете виконувати зі змінною. Наприклад, ми можемо індексувати стрічки і списки, але не можемо індексувати цілі.

Більшість мов програмування дозволяють виконання окремих блоків програми, коли використовуються умовні вирази або *if* оператори.

*If* оператор, - це керуюча (управляюча) структура, тому що вона керує виконанням програми, вона вибирає, яка частина коду в програмі буде виконуватися. Іншою контрольную структурою є for оператор циклу.

При написанні програм можна поєднувати оператори if та for.

Якщо потрібно, щоб щось відбувалося, коли умовний вираз не справджується, використовується оператор *else* в *If* – твердженні.

Якщо залишити частину умов *if* твердження пустою, то не пуста стрічка або список буде оброблятися як *true* а пуста стрічка чи список, як *false.*

У випадку, коли *if.* вираз твердження задовольняється, *elif* вираз не виконується і програма ніколи не виведе на екран 2. *elif* вираз надає більше інформації ніж *if* вираз. Якщо *elif* вираз справджується то це означає, що не тільки одна умова справджується але і означає, що умова *if* виразу не справдилася.

1. ***Клас FreqDist для простих статистичних досліджень***

Для автоматичного визначення слів, які є найбільш інформативними для текстів певного жанру або певної тематики спочатку інтуітивно виникає думка побудувати частотний список або частотний розподіл. Частотний розподіл вказує на частоту з якою в тексті зустрічається кожне зі слів. Такий частотний список називають розподілом тому, що він вказує яким чином загальна кількість слів розподіляється між словниковими статями (оригінальні слова) в тексті. Враховуючи що побудова частотних розподілів часто необхідна при обробці природної мови в NLTK реалізовано окремий клас FreqDist в модулі nltk.probability .

1. ***Колокації та біграми***.

Колокація це словосполучення яке зустрічається дуже часто. red wine –це колокація а the wine – ні . Характерною рисою колокацій є те що вони є стійкі до заміни одного зі слів на інше подібне за змістом (maroon wine). Для того щоб побудувати колокації спочатку потрібно побудувати на основі тексту пари слів, або біграми. Для цього можна використати функцію bigrams():

>>> bigrams(['more', 'is', 'said', 'than', 'done'])

[('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]

Так як колокації це частотні біграми з врахуванням випадків рідкісних слів, то нам потрібно знайти такі біграми частота яких вища ніж частоти слів з яких він складається. Функція collocations() реалізує такі дії.

**Висновок:** я навчилась використовувати умовні оператори мови Python, а також ознайомилась з класом FreqDist , та та виконала індивідуальні завдання,застосувавши ці знання на практиці.

**Практичні завдання**

**Завдання 3.** Створіть змінну *sentence* і присвойте їй значення *‘she sells sea shells by the sea shore’* та напишіть фрагмент програми, яка генерує нову стрічку додаючи *‘like’* перед кожним зі слів , яке починається з *‘se’*.

*Програма:*

sentence='she sells sea shells by the sea shore'

words=sentence.split()

print words

str=''

for word in words:

if word.startswith('se'):

str=str+'like'+' '+word+' ';

print str

*Результат:*

['she', 'sells', 'sea', 'shells', 'by', 'the', 'sea', 'shore']

like sells like sea like sea

**Завданя 8:** Виконати наступні приклади і пояснити різницю між ними

w.isupper()

not w.islower()

*Програма:*

sentence='She Sells sea shells by the sea shore.'

print sentence.isupper()

print not sentence.islower()

*Результат:*

False

True

Функція isupper() перевіряє, чи всі літери великі – вданому випаду ні, тому результат False. Функція islower() перевіряє, чи всі літери малі – також ні, тому вона також повертає False, а вираз ‘ print not islower()’ означає, що потрібно вивести результат, протилежний тому, що повертає функція.

**Завдання 9.** Знайдіть в тексті № 5 всі слова довжина яких дорівнює 4 і побудуйте для них частотний розподіл.

*Програма:*

import nltk

from nltk.book import \*

from nltk import FreqDist

V=set(text5)

short\_words = [w for w in V if len(w) <= 4]

print (FreqDist(short\_words))

*Результат:*

<FreqDist with 2368 samples and 2368 outcomes>

**Завдання 11.** Напишіть вираз для знаходження в тексті №6 всіх слів які відповідають наступним вимогам: закінчуються на ize; містять літеру z; містять послідовність літер pt; написані з великої літери . Результат представити, як список слів.

*Програма:*

import nltk

from nltk.book import \*

from nltk import FreqDist

str=''

e='ize'

z='z'

pt='pt'

for word in text6:

if word.endswith(e) or (z in word) or (pt in word):

str=str+' '+word+' ';

for word in text6:

if word.istitle():

str=str+' '+word+' ';

print(str.split())

*Результат:*

['empty', 'zone', 'aptly', 'amazes', 'Thpppppt', 'Thppt', 'Thppt', 'empty', 'Fetchez', 'Fetchez', 'Thppppt', 'temptress', 'temptation', 'ptoo', 'zoop', 'zoo', 'zhiv', 'frozen', 'zoosh', 'Chapter', 'excepting', 'Thpppt', 'Whoa', 'Halt', 'Who', 'It', 'I', 'Arthur', 'Uther', 'Pendragon', 'Camelot', 'King', 'Britons', 'Saxons', 'England', 'Pull', 'I', 'Patsy', 'We', 'Camelot', 'I', 'What', 'Ridden', 'Yes', 'You', 'What', 'You', 'So', 'We', 'Mercea', 'Where', 'We', 'Found', 'In', 'Mercea', 'The', 'What', 'Well', 'The', 'Are', 'Not', 'They', 'What', 'A', 'It', 'It', 'It', 'A', 'Well', 'Will', 'Arthur', 'Court', 'Camelot', 'Listen', 'In', 'Please', 'Am', 'I', 'I', 'It', 'African', 'Oh', 'African', 'European', 'That', 'Oh', 'I', 'Will', 'Camelot', 'But', 'African', 'Oh', 'So', 'Wait', 'Supposing', 'No', 'Well', 'They', 'What',...]

**Завдання 16.** Побудуйте колокації для текстів №1 та №5. Результати порівняйте.

*Програма:*   
import nltk

from nltk.book import \*

from nltk import FreqDist

print ('Collocations from text1:')

print(text1.collocations())

print ('Collocations from text5:')

print(text5.collocations())

[len(w) for w in text1]

fdist = FreqDist([len(w) for w in text1])

print ('List of possible lenghts of words of text1:')

print (fdist.keys())

print ('List of possible lenghts and the corresponding number of usages of text1:')

print (fdist.items())

print ('The most frequent lenght of words in text1 is ', fdist.max())

print ('The number of usages of ',fdist.max(),'-literal words is ',fdist[fdist.max()])

print('The fequency of ',fdist.max(),'-literal words in text1 is ', fdist.freq(fdist.max())\*100, '%')

[len(w) for w in text5]

fdist = FreqDist([len(w) for w in text5])

print ('List of possible lenghts of words of text5:')

print (fdist.keys())

print ('List of possible lenghts and the corresponding number of usages of text5:')

print (fdist.items())

print ('The most frequent lenght of words in text5 is ', fdist.max())

print ('The number of usages of ',fdist.max(),'-literal words in text5 is ',fdist[fdist.max()])

print('The fequency of ',fdist.max(),'-literal words in text5 in text5 is ', fdist.freq(fdist.max())\*100, '%')

*Результат:*

Collocations from text1:

Sperm Whale; Moby Dick; White Whale; old man; Captain Ahab; sperm

whale; Right Whale; Captain Peleg; New Bedford; Cape Horn; cried Ahab;

years ago; lower jaw; never mind; Father Mapple; cried Stubb; chief

mate; white whale; ivory leg; one hand

Collocations from text5:

wanna chat; PART JOIN; MODE #14-19teens; JOIN PART; PART PART;

cute.-ass MP3; MP3 player; JOIN JOIN; times .. .; ACTION watches; guys

wanna; song lasts; last night; ACTION sits; -...)...- S.M.R.; Lime

Player; Player 12%; dont know; lez gurls; long time

List of possible lenghts of words of text1:

dict\_keys([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20])

List of possible lenghts and the corresponding number of usages of text1:

dict\_items([(1, 47933), (2, 38513), (3, 50223), (4, 42345), (5, 26597), (6, 17111), (7, 14399), (8, 9966), (9, 6428), (10, 3528), (11, 1873), (12, 1053), (13, 567), (14, 177), (15, 70), (16, 22), (17, 12), (18, 1), (20, 1)])

The most frequent lenght of words in text1 is 3

The number of usages of 3 -literal words is 50223

The fequency of 3 -literal words in text1 is 19.255882431878046 %

List of possible lenghts of words of text5:

dict\_keys([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 43, 50, 51, 54, 67, 70])

List of possible lenghts and the corresponding number of usages of text5:

dict\_items([(0, 1), (1, 6236), (2, 8129), (3, 10938), (4, 10204), (5, 3789), (6, 2442), (7, 1457), (8, 796), (9, 419), (10, 222), (11, 140), (12, 63), (13, 29), (14, 12), (15, 15), (16, 28), (17, 9), (18, 8), (19, 6), (20, 9), (21, 8), (22, 5), (23, 6), (24, 3), (25, 8), (26, 2), (27, 2), (28, 3), (29, 1), (30, 2), (31, 2), (32, 2), (34, 1), (35, 1), (36, 2), (37, 1), (38, 1), (43, 3), (50, 1), (51, 1), (54, 1), (67, 1), (70, 1)])

The most frequent lenght of words in text5 is 3

The number of usages of 3 -literal words in text5 is 10938

The fequency of 3 -literal words in text5 in text5 is 24.30126638524772 %