МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота № 8

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.
СТРУКТУРНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON (частина2).

Виконала:
студентка групи ПРЛм-12
Іваськів М.Є.

Прийняв:
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

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.В наступному прикладі показано, яким чином вбудована функція len() або розроблена функція last_letter() передаються, як аргументи іншій функції:

```
>>> sent = ['Take', 'care', 'of', 'the', 'sense', ',', 'and', 'the',
...         'sounds', 'will', 'take', 'care', 'of', 'themselves', '.']
>>> def extract_property(prop):
...     return [prop(word) for word in sent]
...
>>> extract_property(len)
[4, 4, 2, 3, 5, 1, 3, 3, 6, 4, 4, 4, 2, 10, 1]
>>> def last_letter(word):
...     return word[-1]
>>> extract_property(last_letter)
['e', 'e', 'f', 'e', 'e', ',', 'd', 'e', 's', 'l', 'e', 'e', 'f', 's', '.']
```

Об'єкти len та last_letter передаються у функцію як списки та словники (тип даних словник буде розглянуто пізніше). Зауважимо, що дужки після імені функції використовуються тільки при її виклику, а якщо функція трактується, як об'єкт (аргумент іншої функції) то дужки опускаються.

Python підтримує ще один спосіб визначення функцій як аргументів іншої функції , це так званий лямбда-вираз (анонімна функція). Спробуємо виконати дії функції last_letter() без її створення і відповідно не використовуючи її імя. За допомогою лямбда-виразу отримаємо наступний результат:

```
>>> extract_property(lambda w: w[-1])
['e', 'e', 'f', 'e', 'e', ',', 'd', 'e', 's', 'l', 'e', 'e', 'f', 's', '.']
```

Наступний приклад ілюструє передавання функції до функції sorted(). У випадку виклику функції sorted() з одним аргументом #1 (сортування списку) ця функція використовує для порівняння вбудовану функцію cmp() (про що свідчить #2). Звичайно, можна, як аргумент, використати і власну функцію, наприклад функцію сортування в порядку спадання довжин елементів.

```
>>> sorted(sent) #1
[',', '.', 'Take', 'and', 'care', 'care', 'of', 'of', 'sense', 'sounds',
'take', 'the', 'the', 'themselves', 'will']
>>> sorted(sent, cmp) #2
[',', '.', 'Take', 'and', 'care', 'care', 'of', 'of', 'sense', 'sounds',
'take', 'the', 'the', 'themselves', 'will']
>>> sorted(sent, lambda x, y: cmp(len(y), len(x))) #3
['themselves', 'sounds', 'sense', 'Take', 'care', 'will', 'take', 'care',
'the', 'and', 'the', 'of', 'of', ',', '.']
```

ВИКОНАННЯ ЗАВДАНЬ

Завдання 1. 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.

В цьому прикладі відбувається присвоєння посилання на значення змінної sent1 новій змінній sent2. Якщо відбуваються зміни в sent1 то ці зміни також торкаються і sent2.

```
sent1 = ['I', 'like', 'hiking' ',', 'and', 'you', '?']
sent2=sent1
print sent2
sent1[2]='cooking'
print sent1
print sent2
>>>
['I', 'like', 'hiking,', 'and', 'you', '?']
['I', 'like', 'cooking', 'and', 'you', '?']
['I', 'like', 'cooking', 'and', 'you', '?']
```

Завдання 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.

```
import numpy
from numpy import array
def sets(text):
        l=[len(word) for word in text]
        v=[len([letter for letter in word if letter.lower() in 'aeiouy']) for wo
        word_vowels=array(([l],[v]))
        print word_vowels
sent=['Ukraine', 'is', 'marvelous']
print sets(sent)
>>>
[[[7 2 9]]

 [[4 1 4]]]
```

Завдання 9. 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. Написати функцію gematria() для сумування числових значень літер в слові згідно наступних значень letter_vals:

```
>>> 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}
```

```
letter_vals = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':80, 'g':3, 'h':8,'i':10, '
def gematria(word):
    return sum([letter_vals[char] for char in word.lower()])

>>> gematria('university')
997
>>> gematria('hello')
143
>>> gematria('legendary')
308
```

Завдання 12. Написати функцію shorten(text, n) обробки тексту , для вилучення *n* найбільш частотних слів в тексті. Яким чином змінилась читабельність тексту, після вилучення цих слів?

```
text = 'On the evening of 13 November 2015, a series of coordinated terrorist at
print text
import nltk, re
def shorten(text,n):
        fd=nltk.FreqDist(nltk.word_tokenize(text))
        b=fd.keys()[:n]
        c=[]
        for i in nltk.word_tokenize(text):
                if i not in b:
                        c+=[i]
        import string
        print'most frequent qords:' ,b
        return string.join(c)

>>>
On the evening of 13 November 2015, a series of coordinated terrorist attacks—co
nsisting of mass shootings, suicide bombings and hostage-taking—occurred in Pari
s, France, and Saint-Denis, its northern suburb. Beginning at 21:16 CET, six mas
s shootings and three separate suicide bombings near the Stade de France occurre
d. The deadliest attack was at the Bataclan theatre, where attackers took hostag
es and engaged in a stand-off with police which ended at 00:58 on 14 November.
>>> shorten(text,5)
most frequent qords: [',', 'and', 'at', 'of', 'the']
'On evening 13 November 2015 a series coordinated terrorist attacks\x97consistin
g mass shootings suicide bombings hostage-taking\x97occurred in Paris France Sai
nt-Denis its northern suburb. Beginning 21:16 CET six mass shootings three separ
ate suicide bombings near Stade de France occurred. The deadliest attack was Bat
aclan theatre where attackers took hostages engaged in a stand-off with police w
hich ended 00:58 on 14 November .'
>>> shorten(text,15)
most frequent qords: [',', 'and', 'at', 'of', 'the', 'France', 'November', 'a',
'bombings', 'in', 'mass', 'shootings', 'suicide', '.', '00:58']
'On evening 13 2015 series coordinated terrorist attacks\x97consisting hostage-t
aking\x97occurred Paris Saint-Denis its northern suburb. Beginning 21:16 CET six
 three separate near Stade de occurred. The deadliest attack was Bataclan theatr
e where attackers took hostages engaged stand-off with police which ended on 14'
```

Завдання 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().

Функція itemgetter() використовується з функцією sorted() і тому відбувається сортування слів за літерою, що вказана як індекс, у даному випадку індекс = 1 та -1.

```
from operator import itemgetter
words=['there', 'is', 'an', 'awesome','weather']
print sorted(words, key=itemgetter(1))
print sorted(words, key=itemgetter(-1))
>>>
['weather', 'there', 'an', 'is', 'awesome']
['there', 'awesome', 'an', 'weather', 'is']
```

Завдання 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? Пояснити письмово.

```
import nltk
print help(nltk.edit_distance)
word1 = 'hate'
word2 = 'love'
print nltk.edit_distance(word1, word2)
print nltk.edit_distance('cook','dinner')
>>>
Help on function edit_distance in module nltk.metrics.distance:

edit_distance(s1, s2)
    Calculate the Levenshtein edit-distance between two strings.
    The edit distance is the number of characters that need to be
    substituted, inserted, or deleted, to transform s1 into s2.  For
    example, transforming "rain" to "shine" requires three steps,
    consisting of two substitutions and one insertion:
    "rain" -> "sain" -> "shin" -> "shine".  These operations could have
    been done in other orders, but at least three steps are needed.

    :param s1, s2: The strings to be analysed
    :type s1: str
    :type s2: str
    :rtype int

None
3
6
```

В цьому модулі використовується динамічне програмування. Підхід знизу-вверх. Спершу визначається довжина стрічок, потім відстань редагування між ними.