МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра САПР

Лабораторна робота №7

З дисципліни «Комп'ютерна лінгвістика»

На тему:

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

СТРУКТУРНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON (частина1).

Виконала:

Студентка гр.. ПРЛм-11

Зварич О. І.

Перевірив:

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.

Індивідуальні завдання

1. Знайти в Python's help додаткову інформацію про послідовності. В інтерпретаторі, набрати по черзі help(str), help(list), та help(tuple). На екрані буде відображено повний список функцій властивих кожному з типів. Деякі функції мають спеціальні імена з подвійними підкреслюваннями. Кожній такій функції відповідає і інший запис показаний в документації. Наприклад x.__getitem__(y) відповідає x[y].

class str(basestring)
| str(object) -> string
| Return a nice string representation of the object.
| If the argument is a string, the return value is the same object.
| Method resolution order:
| str
| basestring
| object
| Methods defined here:
| __add__(...)
| x.__add__(y) <==> x+y

class list(object)
| list() -> new empty list
| list(iterable) -> new list initialized from iterable's items
| Methods defined here:
| __add__(...)
| x.__add__(y) <==> x+y

class tuple(object)
| tuple() -> empty tuple
| tuple(iterable) -> tuple initialized from iterable's items
|
| If the argument is a tuple, the return value is the same object.
|
| Methods defined here:
|
| __add__(...)
| x.__add__(y) <==> x+y

2. Знайти три операції, які можна здійснювати і зі списками та із кортежами. Знайти три операції, які не можна здійснювати над кортежами. Знайдіть коли використання списку замість кортежу приводить до Python помилки.

Спільні операції для сисків і кортежів:
| __add__(...)
| x.__add__(y) <==> x+y
| __contains__(...)
| x.__contains__(y) <==> y in x

| __eq__(...)
| x.__eq__(y) <==> x==y
Оерації, які не можна здійснювати з кортежами:
| __imul__(...)
| x.__imul__(y) <==> x*=y
| __delitem__(...)
| x.__delitem__(y) <==> del x[y]
| __iadd__(...)
| x.__iadd__(y) <==> x+=y
Операції, які не здійснюють зі списками:

__hash__(...)

x.__hash__() <==> hash(x)

3. Яким чином можна створити кортеж з одного елемента. Продемонструвати два різні способи.

>>> words = ['You']

>>> tags =['noun']

>>> print zip(words,tags)

[('You', 'noun')]

>>> print list(enumerate(words))

[(0, 'You')]

>>>

4. Створити список words = ['is', 'NLP', 'fun', '?']. Використовуючи операції присвоювання подібні до words[1] = words[2] та тимчасову змінну tmp перетворити цей список в список ['NLP', 'is', 'fun', '!']. Здійснити аналогічні перетворення використовуючи присвоювання в кортежах.

```
>>> words = ['is', 'NLP', 'fun', '?']
>>> tmp=words[0]
>>> words[0]=words[1]
>>> words[1]=tmp
>>> words[3]='!'
>>> print words
['NLP', 'is', 'fun', '!']
>>> words = ['is', 'NLP', 'fun', '?']
>>> words[0], words[1], words[3]=words[1], words[0], '!'
>>> words
['NLP', 'is', 'fun', '!']
```

```
>>> words =[(1,'is'), (2,'NLP'), (3,'fun'), (4,'?')]
>>> tmp=words[0]
>>> words[0]=words[1]
>>> words[1]=tmp
>>> words[3]='!'
>>> print words
[(2, 'NLP'), (1, 'is'), (3, 'fun'), '!']
```

5. Прочитати про вбудовану функцію здійснення порівнянь cmp, набравши help(cmp). Продемонструвати чим поведінка цієї функції відрізняється від поведінки операторів порівняння.

```
>>> help(cmp)
Help on built-in function cmp in module __builtin__:

cmp(...)
    cmp(x, y) -> integer

    Return negative if x<y, zero if x==y, positive if x>y.
```

```
>>> x=3
>>> y=4
>>> cmp(x,y)
-1
>>> x=0.25
>>> y=1
>>> cmp(x,y)
-1
>>> x='week'
>>> y='one'
>>> cmp(x,y)
1
>>> x='you'
>>> y='you'
>>> cmp(x,y)
0
>>>
```

6. Написати програму для коректного виділення в тексті n-грамів з врахуванням граничних випадків: n = 1, та n = len(sent)?

```
>>> sent=['She', 'refused', 'to', 'answer', 'the', 'question']
>>> n=3
>>> [sent[i:i+n] for i in range(len(sent)-n+1)]
[['She', 'refused', 'to'], ['refused', 'to', 'answer'], ['to', 'answe
r', 'the'], ['answer', 'the', 'question']]
>>>
```

7. Використати оператори нерівності для порівняння стрічок, наприклад. 'Monty' < 'Python'. Що станеться, якщо виконати 'Z' < 'a'? Порівняти стрічки,як мають однаковий префікс, наприклад 'Monty' < 'Montague'. Спробувати порівняти структуровані об’єкти ,наприклад. ('Monty', 1) < ('Monty', 2). Чи отримали очікувані результати?

```
>>> 'Monty'<'Python'
True
>>> 'Z'<'a'
True
>>> 'Monty' < 'Montague'
False
>>> ('Monty', 1) < ('Monty', 2)
True
>>> cmp('Z','a')
-1
>>> cmp('Monty','Montague')
1
>>> cmp(('Monty', 1),('Monty', 2))
-1
>>>
```

8. Написати програму видалення пробілів на початку і в кінці стрічки та для видалення зайвих пробілів між словами. Використовувати split() та join(). Оформити у вигляді функції. Функція повинна містити повну стрічку документування.

```
str=' Today is monday '
def clean_spaces(str):
    """
Deletes the first and the last space in text and extra spaces
"""
    tmp=str.split()
    tmp=' '.join(tmp)
    return tmp
print clean_spaces(str)
print help(clean_spaces)


>>>
Today is monday
Help on function clean_spaces in module __main__:

clean_spaces(str)
    Deletes the first and the last space in text and extra spaces
```

9. Написати програму видалення пробілів на початку і в кінці стрічки та для видалення зайвих пробілів між словами. Використовувати re.sub() . Оформити у вигляді функції. Функція повинна містити повну стрічку документування

```
import re
str=' Today is monday '
def clean_spaces(str):
    """
Deletes the first and the last space in text and extra spaces
"""
    tmp=re.sub("^\s+","",str)
    tmp=re.sub("\s+$","",tmp)
    tmp=re.sub("\s{2}","",tmp)
    return tmp
print clean_spaces(str)
print help(clean_spaces)
```

```
>>>
Today is monday
Help on function clean_spaces in module __main__:

clean_spaces(str)
    Deletes the first and the last space in text and extra spaces
```

10. Написати програму сортування слів за їх довжиною. Визначити допоміжну функцію cmp_len, яка буде використовувати функцію cmp для порівняння довжин слів. Функція повинна містити повну стрічку документування.

```
def cmp_len(word1,word2):
    """Find out which word is longer"""
    return cmp(len(word1), len(word2))
def sort_by_len(input_list):
    """Sort list by length"""
    while cmp_len(input_list[0], input_list[1])==-1:
        for i in range(len(input_list)-1):
            if cmp_len(input_list[i], input_list[i+1])==-1: tmp=input_list[i]
            elif cmp_len(input_list[i],input_list[i+1])==1: tmp=input_list[i]
            input_list[i]=input_list[i+1]; input_list[i+1]=tmp
listout=['aa', 'ahahah', 'ahah']
sort_by_len(listout)
print listout
print sort_by_len(listout)
```

```
>>>
['ahahah', 'ahah', 'aa']
```

Висновок: виконавши цю лабораторну роботу, я роботи я закріпила знання, отримані при виконанні попередніх лабораторних робіт та покращила загальні навички у програмуванні.