

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ

Кафедра “Системи автоматизованого проектування”

Звіт

до лабораторної роботи №7

на тему: ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ
ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. СТРУКТУРНЕ
ПРОГРАМУВАННЯ МОВОЮ PYTHON (частина1).
з дисципліни “Комп’ютерна лінгвістика”

Виконала:
студентка групи ПРЛм-11
Гарбуз Л.В.
Прийняв:
викладач
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.

Тексти програм на мові *Python*.

Варіант – 3

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]`.

```
>>> help(str)
Help on class str in module __builtin__:

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
|
|   __contains__(...)
|       x.__contains__(y) <==> y in x
|
|   __eq__(...)
|       x.__eq__(y) <==> x==y
|
|   __format__(...)
|       S.__format__(format_spec) -> unicode
```

```
>>> help(list)
Help on class list in module __builtin__:

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
|
| __contains__(...)
|     x.__contains__(y) <==> y in x
|
| __delitem__(...)
|     x.__delitem__(y) <==> del x[y]
|
| __delslice__(...)
|     x.__delslice__(i, j) <==> del x[i:j]
|
|     Use of negative indices is not supported.
|
| __eq__(...)
|     x.__eq__(y) <==> x==y
|
| __ge__(...)
|     x.__ge__(y) <==> x>=y
```

```
>>> help(tuple)
Help on class tuple in module __builtin__:
```

```
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
|
| __contains__(...)
|     x.__contains__(y) <==> y in x
|
| __eq__(...)
|     x.__eq__(y) <==> x==y
|
| __ge__(...)
|     x.__ge__(y) <==> x>=y
|
| __getattr__(...)
|     x.__getattr__('name') <==> x.name
|
| getitem (...)
|
```

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.

Операції, які можна здійснювати і зі списками та із кортежами:

```
| __add__(...)  
  
| x.__add__(y) <==> x+y  
  
| __contains__(...)  
  
| x.__contains__(y) <==> y in x  
  
| __eq__(...)  
  
| x.__eq__(y) <==> x==y
```

Використання списку замість кортежу приводить до Python помилки:

```
| __imul__(...)  
  
| x.__imul__(y) <==> x*=y  
  
| __delitem__(...)  
  
| x.__delitem__(y) <==> del x[y]  
  
| __iadd__(...)  
  
| x.__iadd__(y)<==>x+=y
```

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

```
>>> words = ['go']  
>>> tags = ['verb']  
>>> zip(words, tags)  
[('go', 'verb')]  
>>> list(enumerate(words))  
[(0, 'go')]  
>>> word = (3, 'letters')  
>>> word  
(3, 'letters')  
>>> words = ['is', 'NLP', 'fun', '?']  
>>> a='!'  
>>> (word[0],word[1],word[3])=(word[1],word[0],a)
```

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]='!'
>>> words
['NLP', 'is', 'fun', '!']

>>> words = ['is', 'NLP', 'fun', '?']
>>> aaa='!'
>>> (words[0],words[1],words[3])=(words[1],words[0],aaa)
>>> words
['NLP', 'is', '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=7
>>> y=5
>>> cmp(x,y)
1
>>> x=5
>>> y=7
>>> cmp(x,y)
-1
>>> x=5
>>> y=5
>>> cmp(x,y)
0
>>> x='yes'
>>> y='no'
>>> cmp(x,y)
1

>>> x='red'
>>> y='redish'
>>> cmp(x,y)
-1

```

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

```

-
>>> sent = ['They', 'have', 'everything', 'in', 'this', 'world']
>>> n=3
>>> [sent[i:i+n] for i in range(len(sent)-n+1)]
[['They', 'have', 'everything'], ['have', 'everything', 'in'], ['everything', 'in',
'this'], ['in', 'this', 'world']]
>>> |

```

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=' You are simply the best '
def clean_spaces(str):
    'Delete first, last and extra spaces'
    aaa=str.split()
    aaa=' '.join(aaa)
    return aaa
print clean_spaces(str)
print help(clean_spaces)

```

```

You are simply the best
Help on function clean_spaces in module __main__:

clean_spaces(str)
    Delete first, last 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='    Show must go on    '
def clean_spaces(str):
    'Delete first,last and extra spaces'
    aaa=re.sub("^\s+","",str)
    aaa=re.sub("\s+$","",aaa)
    aaa=re.sub("\s{2}","",aaa)
    return aaa
print clean_spaces(str)
print help(clean_spaces)

>>>
Show must go on
Help on function clean_spaces in module __main__:

clean_spaces(str)
    Delete first,last 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):
    return cmp(len(word1),len(word2))

>>> def sort_by_len(input_list):
    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=['we','you','they','he']
>>> sort_by_len(listout)
>>> listout
['we', 'he', 'you', 'they']
>>> |
```

Висновок: на цій лабораторній роботі я у своїх програмах використовувала структурне програмування мовою Python, повторила та закріпила знання отримані при виконанні попередніх лабораторних робіт, покращила загальні навички у програмуванні.