МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра "Системи автоматизованого проектування"

Звіт

до лабораторної роботи №11

на тему: ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ СИНТАКСИЧНИЙ АНАЛІЗ (частина1).

з дисципліни "Комп'ютерна лінгвістика"

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

Львів 2015

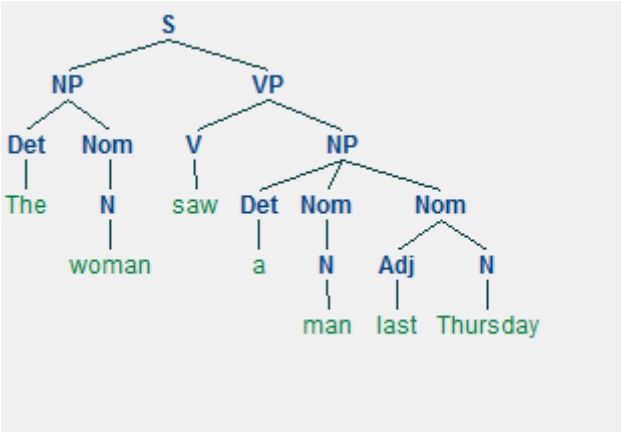
Мета роботи: вивчення основ програмування на мові Python. Ознайомлення з автоматичним синтаксичним аналізом в NLTK.

Тексти програм на мові *Python*.

Варіант – 3

5. Написати програму побудови дерев для речення The woman saw a man last Thursday.

```
>>> import nltk
>>> from nltk import grammar
>>> grammar = nltk.parse_cfg("""
S -> NP VP
NP -> Det Nom | Det Nom Nom
VP -> V NP
Nom -> Adj N | N
Det -> 'The'| 'a'
Adj -> 'last'
N -> 'woman' | 'man' | 'Thursday'
V -> 'saw'
""")
>>> sent = "The woman saw a man last Thursday".split()
>>> parser = nltk.ChartParser(grammar)
>>> trees = parser.nbest_parse(sent)
>>> for tree in trees:
        tree.draw()
```

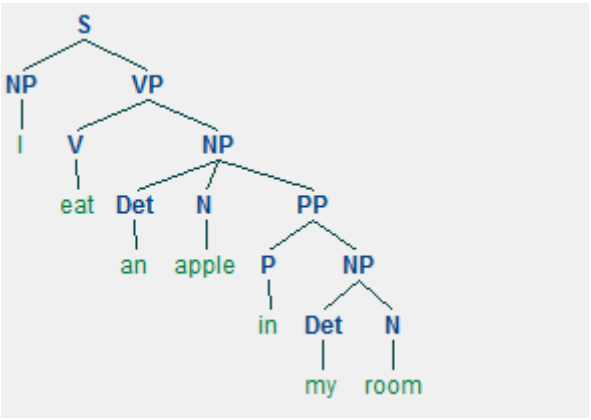


2. В класі Tree реалізовано різноманітні корисні методи. Переглянути файл допомоги Tree з документації та описати основні з цих методів (import Tree, help(Tree).

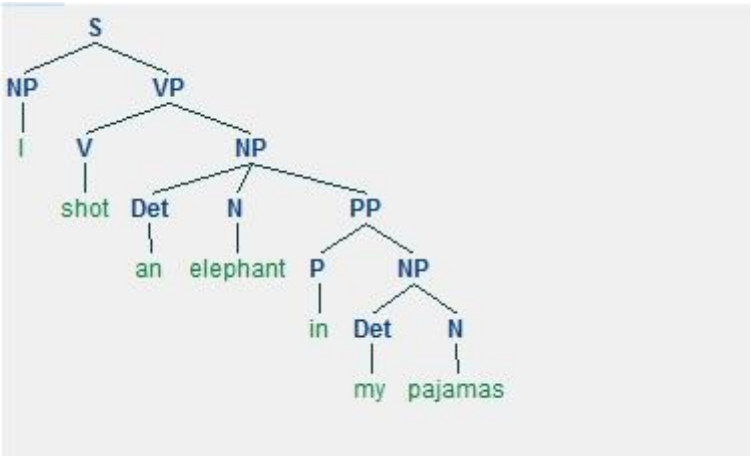
```
>>> import nltk
>>> from nltk import Tree
>>> help(Tree)
Help on class Tree in module nltk.tree:

class Tree(__builtin__.list)
 |  A Tree represents a hierarchical grouping of leaves and subtrees.
 |  For example, each constituent in a syntax tree is represented by a single Tree.
 |
 |  A tree's children are encoded as a list of leaves and subtrees,
 |  where a leaf is a basic (non-tree) value; and a subtree is a
 |  nested Tree.
 |
 |      >>> from nltk.tree import Tree
 |      >>> print Tree(1, [2, Tree(3, [4]), 5])
 |      (1 2 (3 4) 5)
 |      >>> vp = Tree('VP', [Tree('V', ['saw']),
 |      ...                   Tree('NP', ['him'])])
 |      >>> s = Tree('S', [Tree('NP', ['I']), vp])
 |      >>> print s
 |      (S (NP I) (VP (V saw) (NP him)))
 |      >>> print s[1]
 |      (VP (V saw) (NP him))
 |      >>> print s[1,1]
 |      (NP him)
 |      >>> t = Tree("(S (NP I) (VP (V saw) (NP him)))")
 |      >>> s == t
 |      True
 |      >>> t[1][1].node = "X"
 |      >>> print t
 |      (S (NP I) (VP (V saw) (X him)))
 |      >>> t[0], t[1,1] = t[1,1], t[0]
 |      >>> print t
 |      (S (X him) (VP (V saw) (NP I)))
 |
 |  The length of a tree is the number of children it has.
 |
 |      >>> len(t)
 |      2
```

4. Перетворити всі дерева , які зустрічаються в методичних вказівка і зображені за допомогою дужок використовуючи nltk.Tree() . Використовувати draw() для побудови графічного зображення дерева.

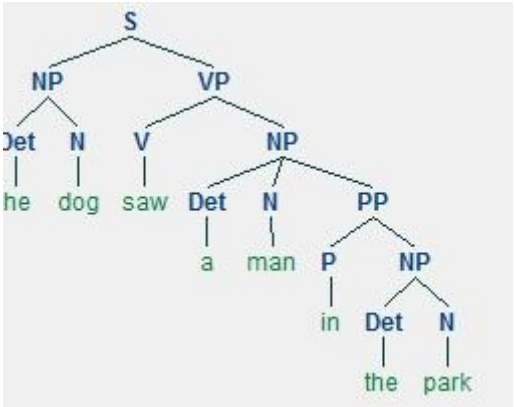
```
>>> import nltk
>>> groucho_grammar = nltk.parse_cfg("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'an' | 'my'
N -> 'apple' | 'room'
V -> 'eat'
P -> 'in'
""")
>>> sent = ['I','eat','an','apple','in','my','room']
>>> parser = nltk.ChartParser(groucho_grammar)
>>> trees = parser.nbest_parse(sent)
>>> for tree in trees:
        tree.draw()
```




```
groucho_grammar = nltk.parse_cfg("""
 S -> NP VP
 PP -> P NP
 NP -> Det N | Det N PP | 'I'
 VP -> V NP | VP PP
 Det -> 'an' | 'my'
 N -> 'elephant' | 'pajamas'
 V -> 'shot'
 P -> 'in'
 """)
sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
trees = parser.nbest_parse(sent)
for tree in trees:
    derevo=tree.draw()
print 'tree.draw'
print derevo
```

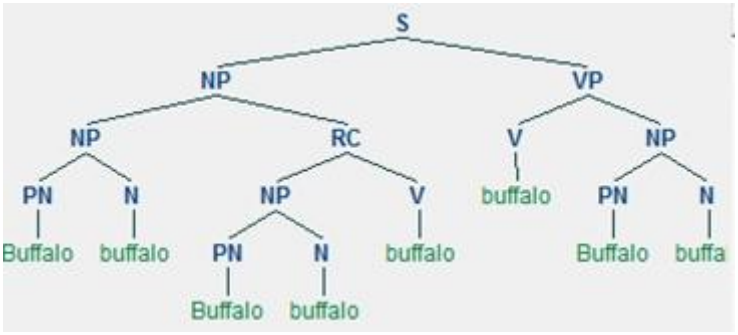


```
grammar1 = nltk.parse_cfg("""
  S -> NP VP
  NP -> Det N
  Det -> "the"
  N -> "dog"
  VP -> V NP
  V -> "saw"
  NP -> Det N PP
  Det -> "a"
  N -> "man"
  PP -> P NP
  P -> "in"
  NP -> Det N
  Det -> "the"
  N -> "park"
    """)
sent = "the dog saw a man in the park ".split()
parser = nltk.ChartParser(grammar1)
trees = parser.nbest_parse(sent)
for tree in trees:
    derevo3=tree.draw()
print 'tree.draw'
print derevo3
```



10. Здійснити аналіз послідовності слів: Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo. Оскільки, згідно з http://en.wikipedia.org/wiki/Buffalo_buffalo_Buffalo_buffalo_buffalo_buffalo_Buffalo_buffalo це граматично правильне речення, напишіть контексно-вільну граматику на основі дерева наведеного на цій сторінці з Інтернету. Здійсніть нормалізацію слів (lowercase), для моделювання ситуації коли слухач сприймає це речення на слух. Скільки дерев розбору може мати це дерево в такому випадку?

```
>>> grammar=nltk.parse_cfg("""
S -> NP VP
NP -> NP RC | PN N
VP -> V NP
RC -> NP V
PN -> "Buffalo"
N -> "buffalo"
V -> "buffalo"
""")
>>> sent = "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo".split()
>>> parser = nltk.ChartParser(grammar)
>>> trees = parser.nbest_parse(sent)
>>> for tree in trees:
        tree.draw()
```



12. Написати програму порівняння швидкодії всіх аналізаторів, які згадувалися в методичних. Використовувати timeit функцію для визначення часу синтаксичного аналізу одного і того самого речення різними аналізаторами.

```
import timeit
t = timeit.Timer(setup='from nltk import RecursiveDescentParser ')
a=t.timeit()
print 'RecursiveDescentParser: ', a

t = timeit.Timer(setup='from nltk import ShiftReduceParser')
b=t.timeit()
print 'ShiftReduceParser: ', b

t = timeit.Timer(setup='from nltk import ChartParser')
c=t.timeit()
print 'ChartParser: ', c

>>>
RecursiveDescentParser:  0.0433358666489
ShiftReduceParser:  0.0448515025132
ChartParser:  0.0426903065229
>>>
```

9. Використовуючи ручку і папір, здійснити всі кроки роботи аналізаторів рекурсивного спуску та переміщення згортання для довільного речення використовуючи просту контексно-вільну граматику.

```
>>> import nltk
>>> from nltk.parse import ShiftReduceParser
>>> grammar=nltk.parse_cfg("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'the'
N -> 'show'
V -> 'stole'
""")
>>> sp_parse=nltk.ShiftReduceParser(grammar)
>>> sent = "I stole the show".split()
>>> print sp_parse.parse(sent)
(S (NP I) (VP (V stole) (NP (Det the) (N show))))
>>>
```

Висновок: на цій лабораторній роботі я ознайомилася з автоматичним синтаксичним аналізом в NLTK.