

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
«ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра  
«Системи автоматизованого проектування»

Звіт

До лабораторної роботи №11

З курсу: «Комп'ютерна лінгвістика»

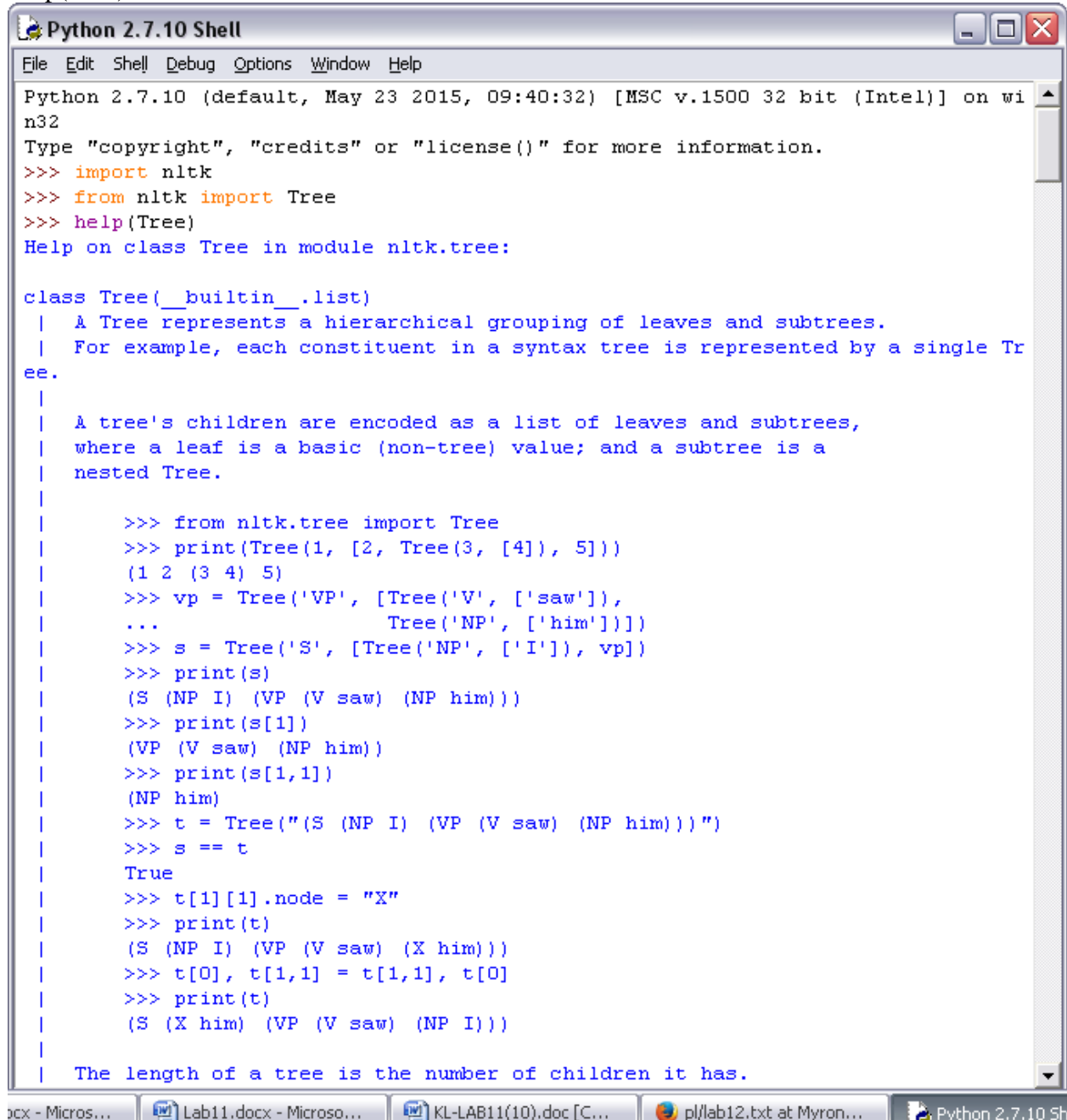
На тему: «Вивчення бібліотеки прикладних програм NLTK для опрацювання текстів  
природною мовою. Автоматичний синтаксичний аналіз (частина2)»

Виконала:  
ст. гр. ПРЛм-11  
Зварич О.І.  
Перевірів:  
викладач  
Дупак Б.П.

Львів-2015

## Варіант 4

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



```
Python 2.7.10 Shell
File Edit Shell Debug Options Window Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win
n32
Type "copyright", "credits" or "license()" for more information.
>>> import nltk
>>> from nltk import Tree
>>> help(Tree)
Help on class Tree in module nltk.tree:

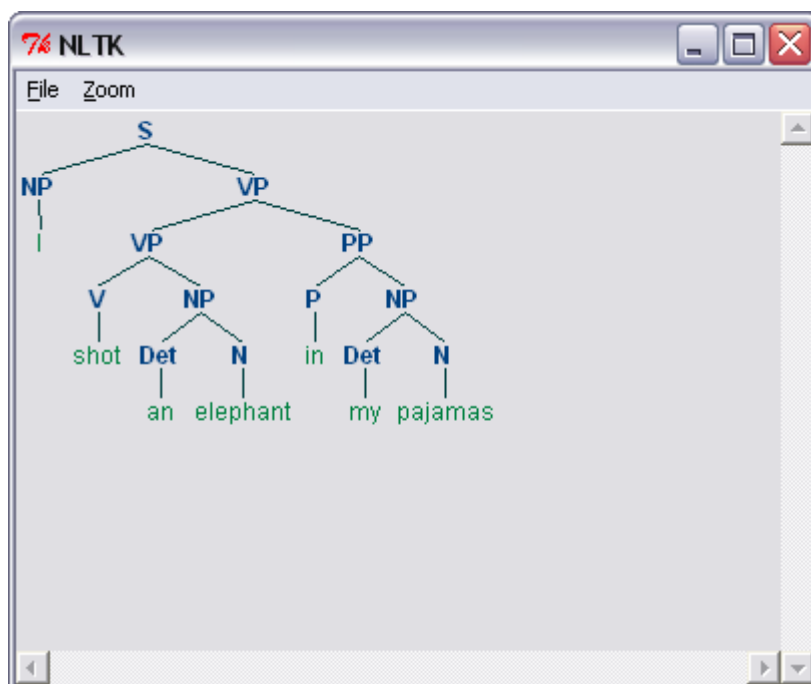
class Tree(_builtin_.list)
 |   A Tree represents a hierarchical grouping of leaves and subtrees.
 |   For example, each constituent in a syntax tree is represented by a single Tr
ee.
 |
 |   A tree's children are encoded as a list of leaves and subtrees,
 |   where a leaf is a basic (non-tree) value; 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.
```

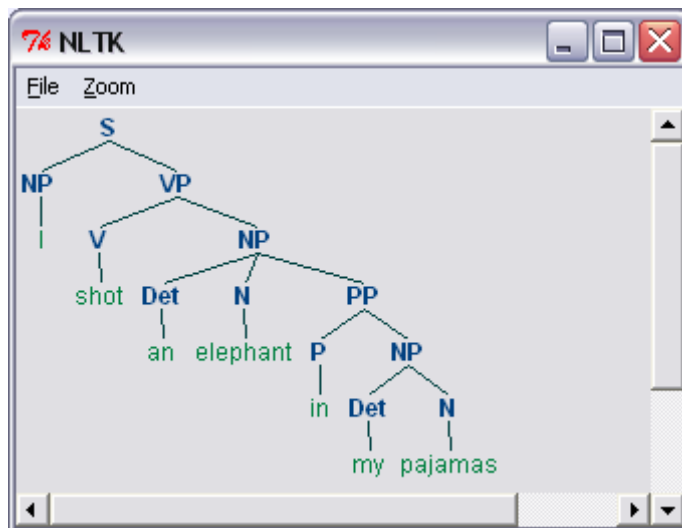
4. Перетворити всі дерева, які зустрічаються в методичних вказівка і зображені за допомогою дужок використовуючи nltk.Tree(). Використовувати draw() для побудови графічного зображення дерева.

```
lab11_4.py - E:/комплінгв/lab11_4.py (2.7.10)
File Edit Format Run Options Window Help

import nltk
from nltk import Tree
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 derevo
```

Ln: 2 Col: 0



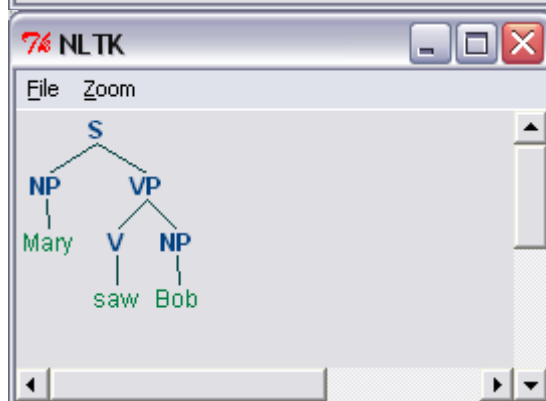


\*lab11\_4\_2.py - E:/комплінгв/lab11\_4\_2.py (2.7.10)\*

File Edit Format Run Options Window Help

```
import nltk
from nltk import Tree
grammar1 = nltk.parse_cfg("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
sent = "Mary saw Bob".split()
rd_parser = nltk.RecursiveDescentParser(grammar1)
for tree in rd_parser.nbest_parse(sent):
    tree.draw()
```

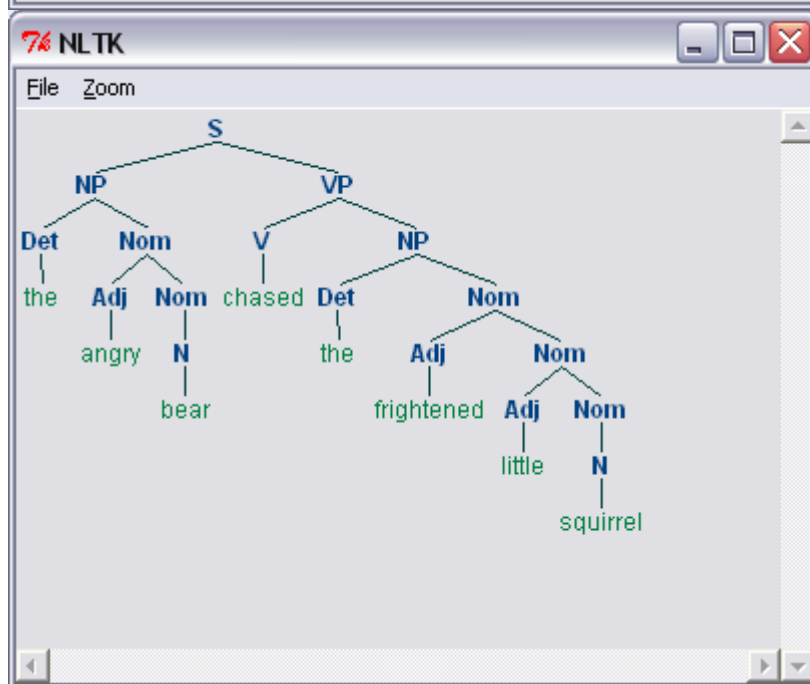
Ln: 18 Col: 0



```
lab11_3.py - E:/комплінгв/lab11_3.py (2.7.10)
File Edit Format Run Options Window Help

import nltk
from nltk import Tree
grammar2 = nltk.parse_cfg("""
S -> NP VP
NP -> Det Nom | PropN
Nom -> Adj Nom | N
VP -> V Adj | V NP | V S | V NP PP
PP -> P NP
PropN -> 'Buster' | 'Chatterer' | 'Joe'
Det -> 'the' | 'a'
N -> 'bear' | 'squirrel' | 'tree' | 'fish' | 'log'
Adj -> 'angry' | 'frightened' | 'little' | 'tall'
V -> 'chased' | 'saw' | 'said' | 'thought' | 'was' | 'put'
P -> 'on'
""")
sent = "the angry bear chased the frightened little squirrel ".split()
parser = nltk.ChartParser(grammar2)
trees = parser.nbest_parse(sent)
for tree in trees:
    tree.draw()
```

Ln: 19 Col: 18

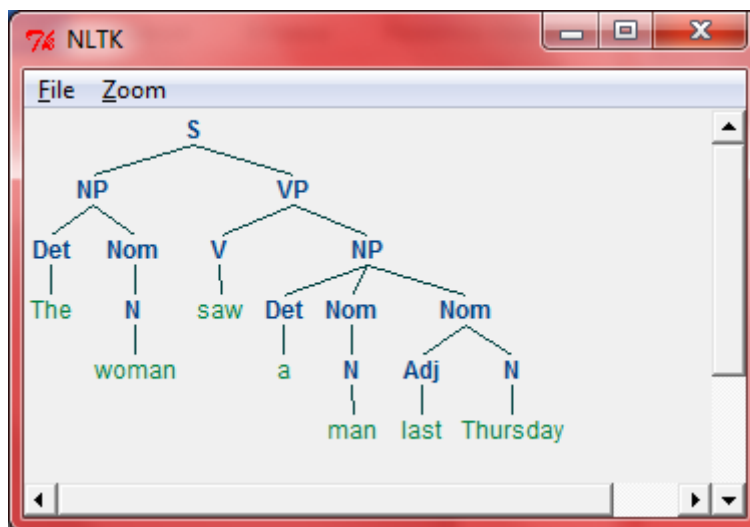


5. Написати програму побудови дерев для речення The woman saw a man last Thursday.

```

import nltk
from nltk import Tree
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()

```



8. Написати програму для пошуку відповіді на питання. Чи може grammar1 граматика використовуватися для опису речення довжиною більше ніж 20 слів?

```

import nltk
from nltk import Tree
grammar1 = nltk.parse_cfg("""
S -> NP VP
VP -> V NP | V NP PP
PP -> P NP
V -> "saw" | "ate" | "walked"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "man" | "dog" | "cat" | "telescope" | "park"
P -> "in" | "on" | "by" | "with"
""")
sent = "Mary saw Bob in the park the dog walked with a man John walked with tele
sr_parse = nltk.ShiftReduceParser(grammar1, trace=2)
print sr_parse.parse(sent)
sr_parse = nltk.ShiftReduceParser(grammar1)
print sr_parse.parse(sent)
for p in grammar1.productions():
    print p

```

>>>

Parsing 'Mary saw Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park'

[ \* Mary saw Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ 'Mary' \* saw Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ NP \* saw Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ NP 'saw' \* Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ NP V \* Bob in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ NP V 'Bob' \* in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ NP V NP \* in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ NP VP \* in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S \* in the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ S 'in' \* the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S P \* the park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ S P 'the' \* park the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S P Det \* park the dog walked with a man John walked with telescope Mary ate my cat in the park]

S [ S P Det 'park' \* the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S P Det N \* the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S P NP \* the dog walked with a man John walked with telescope Mary ate my cat in the park]

R [ S PP \* the dog walked with a man John walked with telescope Mary ate my cat in the park]

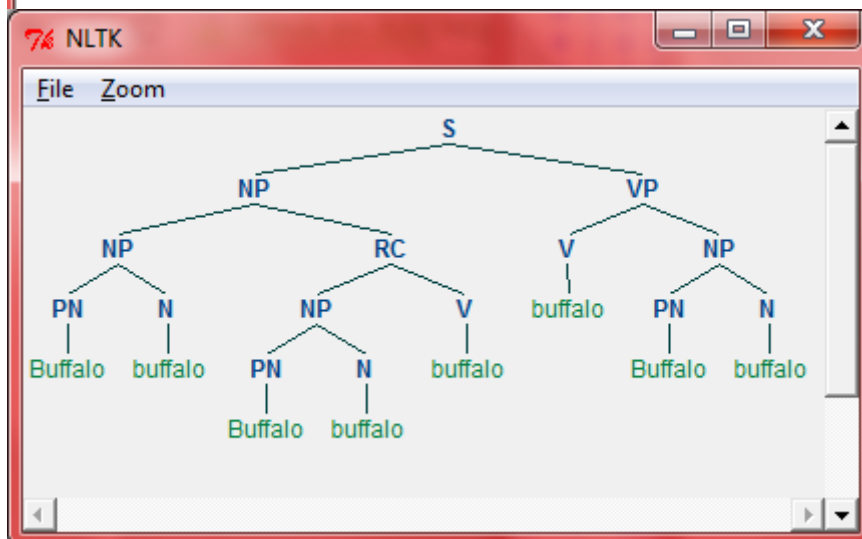
10 Здійснити аналіз послідовності слів: Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo. Оскільки, згідно 3

[http://en.wikipedia.org/wiki/Buffalo\\_buffalo\\_Buffalo\\_buffalo\\_buffalo\\_buffalo\\_Buffalo\\_buffalo](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?

```

import nltk
from nltk import Tree
Buffalo_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(Buffalo_grammar)
trees = parser.nbest_parse(sent)
for tree in trees:
    tree.draw()
r="Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo"
print r.lower()
a=r.lower()
b=a.split()
for tree in rd_parser.nbest_parse(b):
    print tree

```



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 nltk
import timeit
t=timeit.Timer(setup='from nltk import ChartParser')
Chart=t.timeit()
print "ChartParser: ", Chart
t=timeit.Timer(setup='from nltk import RecursiveDescentParser')
Recursive=t.timeit()
print "RecursiveDescentParser: ", Recursive
t=timeit.Timer(setup='from nltk import ShiftReduceParser')
Shift=t.timeit()
print "ShiftReduceParser: ", Shift

>>>
ChartParser: 0.0149878383371
RecursiveDescentParser: 0.0150485611265
ShiftReduceParser: 0.0151533293194
>>> |

```



13 Прочитати про "garden path" речення [http://en.wikipedia.org/wiki/Garden\\_path\\_sentence](http://en.wikipedia.org/wiki/Garden_path_sentence). Оцінити обчислювальну складність аналізу таких речень в порівнянні з труднощами аналізу таких речень людиною?

A garden path sentence is a grammatically correct sentence that starts in such a way that a reader's most likely interpretation will be incorrect; they are lured into an improper parse that turns out to be a dead end.

Garden path sentences mostly appear in analytic languages, where word order is heavily relied upon to establish the grammatical case and function in a sentence.

Garden path sentences are less common in spoken communication because the prosodic qualities of speech (such as the stress and the tone of voice) often serve to resolve ambiguities in the written text.

Висновок: на цій лабораторній роботі я ознайомила з автоматичним синтаксичним аналізатором NLTK, писала програми побудов дерев для речень, здійснювала аналіз послідовності слів, писала контекстно-вільну граматику на основі наведеного дерева, здійснювала модифікацію граматики в демонстраційній програмі синтаксичного аналізу згідно алгоритму рекурсивного спуску.