ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ
ΕΡΓΑΣΤΗΡΙΟ 3

ΡΗΧΗ ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ

Η άσκηση αυτή εστιάζει στην ρηχή συντακτική ανάλυση. Ακολουθίες λέξεων οργανώνονται σε φράσεις (ονοματικές, ρηματικές, προθετικές κλπ), δημιουργώντας απλές, ρηχές συντακτικές δομές.

NP[Τη διανομή μερίσματος αξίας 90 δρχ] PP[ανά μετοχή] VP[αποφάσισαν] NP[οι μέτοχοι της Π.Γ. ΝΙΚΑΣ] PP[κατά τη χθεσινή πραγματοποιηθείσα τακτική γενική συνέλευσή τους .]

ΦΟΡΤΩΣΗ ΑΠΑΡΑΙΤΗΤΩΝ ΕΡΓΑΛΕΙΩΝ

1. Τρέξτε την Python μέσω του IDLE (Interactive Development Environment).
Start -> All Programs -> Python 2.5 -> IDLE.

2. Φορτώστε τα ακόλουθα εργαλεία του NLTK:

```
>>>import nltk, re, pprint
```

ΡΗΧΗ ΣΥΝΤΑΚΤΙΚΗ ΑΝΑΛΥΣΗ ΜΕ ΚΑΝΟΝΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

3. Πρότυπα φράσεων είναι συνήθως ακολουθίες μερών του λόγου που μπορούν να σχηματίζουν την φράση. Φτιάξτε **μια** κανονική έκφραση που να «ταιριάζει» στα ακόλουθα πρότυπα ονοματικών φράσεων:

another/DT sharp/JJ dive/NN
trade/NN figures/NNS
any/DT new/JJ policy/NN measures/NNS
earlier/JJR stages/NNS
Panamanian/JJ dictator/NN Manuel/NNP Noriega/NNP
his/PP work/NN
their/PP beautiful/JJ home/NN

Παρατήρηση: Για να καλύψετε όλα τα είδη επιθέτων ή ουσιαστικών μπορείτε να χρησιμοποιήσετε τον χαρακτήρα μπαλαντέρ ‘.’

Εφαρμόστε τον αναλυτή σας σε μια πρόταση ως εξής:

```
grammar = r"""
NP: {<DT><JJ><NN>}          # η κανονική σας έκφραση
"""
cp = nltk.RegexpParser(grammar)
tagged_tokens = [("Rapunzel", "NNP"), ("let", "VBD"), ("down", "RP"),
("her", "PP$"), ("golden", "JJ"), ("hair", "NN")]
>>> print cp.parse(tagged_tokens)
(S
(NP Rapunzel/NNP)
let/VBD
down/RP
(NP her/PP$ long/JJ golden/JJ hair/NN))
```

4. Έχει μεγάλη σημασία η σειρά που έχουν τα πρότυπα στην γραμματική. Έστω ότι έχετε την παρακάτω πρόταση:

```
tagged_tokens = [("The", "DT"), ("enchantress", "NN"),
("clutched", "VBD"), ("the", "DT"), ("beautiful", "JJ"), ("hair", "NN")]
```

Έστω πως έχετε δύο κανονικές εκφράσεις για πρότυπα ονοματικών φράσεων: την <DT><JJ><NN> και την <DT|NN>+. Φτιάχνετε δυο γραμματικές με τις δυο διαφορετικές διατάξεις των εκφράσεων:

```
cp1 = nltk.RegexpParser(r"""
NP: {<DT><JJ><NN>}          # Chunk det+adj+noun
    {<DT|NN>+}              # Chunk sequences of NN and DT
""")

cp2 = nltk.RegexpParser(r"""
NP: {<DT|NN>+}              # Chunk sequences of NN and DT
    {<DT><JJ><NN>}          # Chunk det+adj+noun
""")
```

Εφαρμόζετε τις γραμματικές στην πρότασή σας, με ενεργοποιημένο το ίχνος (tracing). Το ίχνος δείχνει την διαδικασία ανάλυσης.

```
>>> print cp1.parse(tagged_tokens, trace=1)
# Input:
<DT> <NN> <VBD> <DT> <JJ> <NN>
# Chunk det+adj+noun:
<DT> <NN> <VBD> {<DT> <JJ> <NN>}
# Chunk sequences of NN and DT:
{<DT> <NN>} <VBD> {<DT> <JJ> <NN>}
```

```
(S
(NP The/DT enchantress/NN)
clutched/VBD
(NP the/DT beautiful/JJ hair/NN))

>>> print cp2.parse(tagged_tokens, trace=1)
# Input:
<DT> <NN> <VBD> <DT> <JJ> <NN>
# Chunk sequences of NN and DT:
{<DT> <NN>} <VBD> {<DT>} <JJ> {<NN>}
# Chunk det+adj+noun:
{<DT> <NN>} <VBD> {<DT>} <JJ> {<NN>}

(S
(NP The/DT enchantress/NN)
clutched/VBD
(NP the/DT)
beautiful/JJ
(NP hair/NN))
```

5. Αξιολόγηση Αναλυτή.
Έστω ότι έχω την πρόταση “the little cat sat on the mat”. Αρχικά ορίζεται η σωστή της ανάλυση με την συνάρτηση tagstr2tree():

```
>>> correct = nltk.chunk.tagstr2tree(
    "[ the/DT little/JJ cat/NN ] sat/VBD on/IN [ the/DT mat/NN ]")
```

Έστω ότι έχω την Γραμματική

```
>>> grammar = r"NP: {<PRP|DT|POS|JJ|CD|N.*>+}"
```

Φτιάχνω έναν αναλυτή κανονικών εκφράσεων

```
>>>cp = nltk.RegexpParser(grammar)
```

Εφαρμόζω τον αναλυτή στην πρόταση. Η συνάρτηση flatten() επιστρέφει το δέντρο της πρότασης μόνο με την ρίζα και τα φύλλα (τις λέξεις). Δηλαδή το correct.flatten() θα είναι το (S the/DT little/JJ cat/NN sat/VBD on/IN the/DT mat/NN).

```
>>> guess = cp.parse(correct.flatten())
```

Καλώ την συνάρτηση ChunkScore():

```
>>> chunkscore = nltk.chunk.ChunkScore()
```

Υπολογίζω την απόδοση του αναλυτή με βάση την σωστή ανάλυση

```
>>> chunkscore.score(correct, guess)
>>> print chunkscore

ChunkParse score:
IOB Accuracy: 100.0%
Precision: 100.0%
Recall: 100.0%
F-Measure: 100.0%
```

IOB Accuracy είναι το ποσοστό των λέξεων που έχουν επισημειωθεί σωστά ως προς το αν ανήκουν ή όχι σε κάποια φράση. *Precision* (ακρίβεια) είναι το ποσοστό των φράσεων που βρέθηκαν σωστά στο σύνολο των φράσεων που ανιχνεύθηκαν. *Recall* (ανάκληση) είναι το ποσοστό των φράσεων που ανιχνεύθηκαν στο σύνολο των πραγματικών φράσεων της πρότασης. Το μέτρο f (f-measure) είναι συνδυασμός των δύο:

μέτρο f = 2*ακρίβεια*ανάκληση/(ακρίβεια + ανάκληση)

Εργασία (προθεσμία 2 εβδομάδες):

- Φτιάξτε μια σειρά από κανονικές εκφράσεις για ονοματικές φράσεις στα Ελληνικά.
- Τρέξτε τις σε προτάσεις και δείτε την απόδοση του αναλυτή σας.
- Βελτιώστε την γραμματική σας και ξανατρέξτε την στις ίδιες και σε άλλες προτάσεις.
- Επαναλάβετε την διαδικασία μέχρι να βεβαιωθείτε ότι η γραμματική σας έχει ικανοποιητική απόδοση.